UNIVERSITY OF TARTU

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Institute of Computer Science

Information Technology Curriculum

Silver Jürimäe

A Literature Survey of the Development Processes for Secure Software

Bachelor's Thesis (6 ECTS)

Supervisors: Dr. Raimundas Matulevičius

TARTU 2014

A Literature Survey of the Development Processes for Secure Software

Abstract

Secure software development processes are critical part of designing secure software. However, it is hard for the various stakeholders to make the decision about which software development process to choose without a comparison between them. Even further, after choosing the process, stakeholders have to decide which methods and techniques to use to fulfil activities required to develop secure software development processes. This is a problem, because there are a number of methods a stakeholder could use to fulfil these activities, but no explicit links between a method and development process.

In this thesis firstly we perform comparison of three secure system development approaches namely Microsoft Security Development Lifecycle, OWASP CLASP and Cigital's Security Touchpoints. In the next step we focus on step within these approaches, namely the security risk management and carry out an analytical survey to find out current methods for security risk management. We give a short overview and comparison between found methods, which potentially will help stakeholders to select their approach for designing secure software with the focus on security risk analysis. We also provide them with opportunity to perform all activities required in risk analysis phase of the development by giving them an aggregate view of risk management methods. This is essential, because risk analysis is a major part of developing secure software and combining different techniques can be used to discover and mitigate more risks in software under development.

Keywords

Security development processes, Security Development Lifecycle, OWASP CLASP, Cigital's Security Touchpoints, security risk management, Secure i*, SecReq, Secure Tropos, UMLsec, SQUARE, ISSRM domain model, Misuse cases

Uuring turvalise tarkvara arenguprotsesside kohta

Lühikokkuvõte

Turvalise tarkvara arendusprotsessidel on tähtis roll turvalise tarkvara kavandamisel, aga erinevate arendusprotsessidel vahel on rakse valikut teha ilma nendevahelise võrdluseta. Veel enam peale arendusprotsessi rakendamist tuleb valida meetodid, mida kasutada selle arendusprotsessi rakendamisel. Meetodite valikul tekib aga probleem, sest arendusprotsessides ei ole öeldud, milliseid meetodeid tuleks kasutada, et täita vajalikud tegevused turvalise tarkvara arendamiseks.

Selle töö raames me võrdleme kolme erinevat turvalise tarkvara arendusprotsessi: Microsoft Security Development Lifecycle, OWASP CLASP ja Cigital's Security Touchpoints. Järgmisena me keskendume valitud arendusprotsesside faasile, mis käsitleb turvariskide haldust ja viime läbi uuringu, et teada saada, mis on tänapäevased turvariski meetodid. Me anname nendest meetoditest lühikokkuvõtte ja võrdleme neid omavahel, mis loodetavasti lihtsustab nende vahel valimist. Me koostame veel leitud meetoditest ühise vaate, mis aitab kaasa kõigi arendusprotsesside poolt pakutud tegevuste täitmisele selle faasis. See on vajalik, sest riskihaldus mängib suurt rolli turvalise tarkvara arendamisel ja erinevate riskihaldus meetodite kombineerimist saab kasutada, et avastada rohkem riske loodavast tarkvarast ja hiljem neid riske korrektselt leevendada.

Võtmesõnad

Turvalise tarkvara arendusprotsessid, Security Development Lifecycle, OWASP CLASP, Cigital's Security Touchpoints, turvariskide haldamine, Secure i*, SecReq, Secure Tropos, UMLsec, SQUARE, ISSRM domain model, Misuse cases

Table of Contents

List of Tables and Figures

Chapter 1. Introduction

Security has a major role in developing software, but without guidelines it is hard to decide, which activities have to be implemented in order to develop secure software. A secure software process can be defined as the set of activities performed to develop, maintain, and deliver a secure software solution (Davis, 2006). These are a number of secure software processes available and although, they all have the same purpose, they are quite different in structure and activities, so it is hard to decide, which process is suitable for software under development. In this thesis we answer two research questions. The first question is: what are the differences between Security Development Lifecycle (Lipner & Howard, 2005), OWASP CLASP (Graham, 2006) and Cigital's Security Touchpoints (McGraw, 2006) and the second question is: what are the current practices and methods for security risk management?

The purpose of this thesis is to make the comparison between three development processes for secure software: Security Development Lifecycle, OWASP CLASP and Cigital's Security Touchpoints. Furthermore this thesis will provide a link between development processes and methods for security risk management. The link is made by performing a literature review to find out current methods and techniques chosen methods use for security risk management and by comparing the activities of the development processes to techniques of the chosen methods. Chosen methods are Secure i* (Elahi, *et al*., 2010), SecReq (Houmb, *et al*., 2009), Secure Tropos (Giorgini, *et al*., 2007), UMLsec (Jürjens, 2002), SQUARE (Suleiman & Svetinovic, 2012), ISSRM domain model (Alcalde, *et al*., 2009) and Misuse cases (Sindre & Opdahl, 2004). After reviewing the methods we categorise the information given by them to help the stakeholder use these methods successively in order to fulfil required activities.

The thesis is structured as follows. The first part introduces the background. In Chapter 2 we provide the summary and comparison tables to Security Development Lifecycle, OWASP CLASP and Cigital's Security Touchpoints. In Chapter 3 we will give the design for the systematic literature review and in Chapter 4 we will perform the systematic literature review to find current practices and methods for security risk management. Moreover we will provide criteria to compare found methods and find out in which development process can these methods be implemented. In Chapter 5 we will provide an aggregate view on risk analysis and requirements and therefore contribution of this thesis.

Chapter 2. Security Development Processes

There exist several approaches for developing secure software. In this chapter, we review three of these: Security Development Lifecycle (Lipner & Howard, 2005; Microsoft Developer Network, 2012), OWASP CLASP (Graham, 2006; OWASP 2, 2005; OWASP 1, 2012) and Cigital's Security Touchpoints (McGraw, 2006). These security development processes were chosen because they have comprehensive set of activities which cover a large part of the development process. The chapter concludes with their comparison in six different categories: education, project launch, risk analysis and requirements, architectural and detailed design, implementation and testing, release and deployment.

2.1 Security Development Lifecycle

Security Development Lifecycle (SDL) came out in 2002, as a result of Microsoft's commitment to improve the security of its operating system. Microsoft made the SDL to address the security issues they had to face in their products. SDL is a set of activities performed to develop and deliver a secure software solution. The SDL's activities are grouped in seven stages: training, requirements, design, implementation, verification, release and response. In this thesis we are merging the response phase with release phase due to the lack of response activities included in the security development processes reviewed in this thesis. Although SDL stages are security specific, they are very alike to the software development phases. Several activities continue throughout the SDL process, for instance threat modelling and education. Doing so the SDL process focuses mainly on remaking and improving on going results. SDL provides thorough description to which method should be used to carry out activities so the execution of an activity can be achieved.

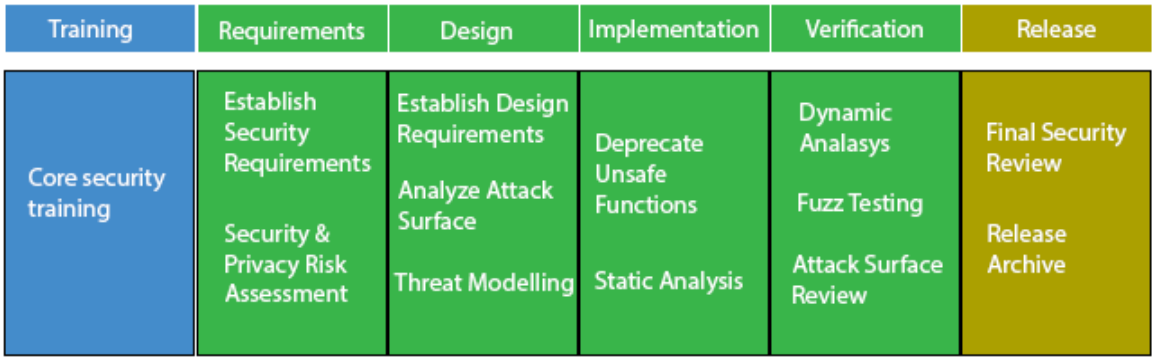

Figure 1 - Six phases of the traditional software development lifecycle (adapted from Microsoft, 2012)

Education is a major part of SDL. Every team member should have knowledge in software security in order to increase the awareness of the problem. Also mandatory advanced education is scheduled annually in order to keep up with the evolving field and new threats. SDL suggests instituting a measurement program to assess the effectiveness of knowledge received by training programs.

Security advisor is assigned to the project who serves as a point of contact, resource and guide as planning continues. This advisor helps the product team with security related issues and remains the team's point of contact from the beginning to the software release. Furthermore, security team is assembled for frequent interactions during software

development. SDL has devised a set of security metrics for product teams in order to monitor their success in implementing SDL.

SDL introduces a security risk assessment (SRA) as a mandatory exercise to identify functional aspects of the software that might require deep security review. SRA will determine which parts of the project will require threat modelling, which security design reviews and which penetration testing. SDL also recommends doing privacy requirements which measures the sensitivity of the data that software will process from a privacy point of view.

Architectural and detailed design is performed mainly by threat modelling. SDL focuses on the impact of the project on user privacy and minimization of attack surface. To minimize attack surface discarding unnecessary features and limiting privileges is suggested. SDL recommends STRIDE (STRIDE, 2007) to evoke threats. STRIDE stands for spoofing (impersonating something or someone else), tampering (modifying data or code), repudiation (claiming to have not performed an action), information disclosure (exposing information to someone not authorized to see it), denial of service (deny or degrade service to users), elevation of privileges (gain capabilities without proper authorization). It also provides all the resources and documents to carry out this technique. SDL also recommends a security expert to review the architecture of the system from security point of view.

SDL suggests applying coding and testing standards for implementation and testing. Coding standards help developers to avoid flaws that can lead to security vulnerabilities. Testing standards help to ensure that testing focuses on detecting potential security risks. Furthermore automated tools are suggested to detect minor errors. It also suggests conducting manual code reviews in order to supplement automated tools. SDL has heavy emphasis on fuzz testing tools, which unlike the static code-scanning tools must be built for each file format and because of this they are able to find errors missed by static analysis tools. The testing mainly covers only black box testing. SDL also describes security push to ensure that the final software meets the requirements and allow deeper review of any legacy code.

During the release phase, the software should be subject to a Final Security Review ("FSR"). The FSR is an independent review of the software conducted by the central security team for the organization. If FSR finds remaining vulnerabilities, the proper response would be to revisit the earlier phases and take other pointed actions to address root causes. SDL emphasizes evaluating reports of vulnerabilities after the release of the product as it helps to detect and eliminate further security weaknesses before they are discovered in the field.

2.2 OWASP CLASP

OWASP CLASP (CLASP), like SDL, is also a process for building secure software. It includes 24 activities and also supplementary resources, which can be fitted to the development process that is used. CLASP's activities are defined mainly from a theoretical angle and so the coverage of the activities is rather broad. CLASP is defined as a set of independent activities that have to be integrated in the development process. The choice of the activities and the order of execution are left open to make the development process more flexible. Furthermore, the execution density of these activities is specified to each activity, so the coordination of these activities is fairly difficult.

Two roadmaps (Legacy and Greenfield) have been made to give help on how to combine the activities into an ordered set.

CLASP defines the roles that are crucial for the security of the software product and appoints the activities to these roles, so the roles are used to help to structure the set of activities. Roles are responsible for the final outcome and the quality of the results of an activity. CLASP has a large set of security resources that support the implementation of the activities. For instance, it has a Vulnerability Lexicon that helps developers to avoid common coding errors in source code and Vulnerability Use Cases to portray conditions under which security services can become vulnerable in the software. 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.

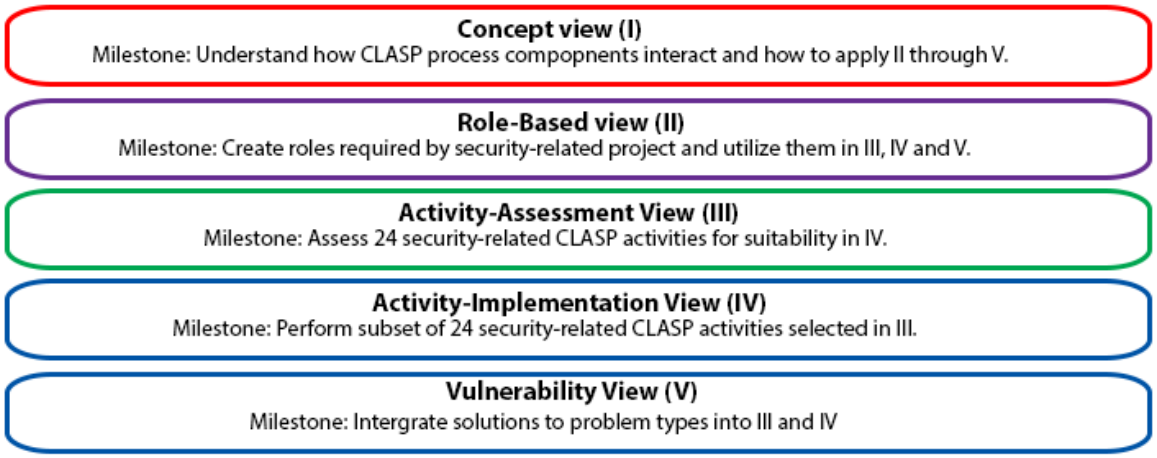

Figure 2 CLASP Views and their interactions (adapted from OWASP 1, 2012)

Education in CLASP is mandatory for all people involved in the project. Awareness programs are implemented, using external expert resources in order to help to ensure that activities promoting secure software will be implemented effectively.

CLASP emphasizes the construction of the security team and they recommend assigning a security officer to the project, which shares knowledge and reviews the project throughout the development process. Furthermore CLASP recommends the use of accountability to boost individual commitment and also has security metrics to assess the security of the product. CLASP emphasizes the importance of making corporate security policy to use as a base for security requirements and it provides templates to ease the making of this security policy.

CLASP recommends identifying data resources and linking them to system roles. Requirements are created by using both offense and defence by means of threat modelling and requirements specification. Threat modelling can be use case driven, during which attacks to use cases are performed and resource driven that concentrates on illegal use of resources. Functional security requirements are set to show how the basic security services are addressed for each resource for determining risk mitigation and resolving deficiencies and conflicts. CLASP also recommends identifying the attacker profile, so it would be simpler to specify where threats could originate.

CLASP supports threat modelling for architectural and detailed design. It includes assessing security posture of technology solutions to research and assess third party components that the project will depend on. CLASP is also devoted to minimize the attack surface by concentrating on restricting access. CLASP advises designers to apply security principles to design to harden and make software more resilient to attacks.

CLASP acknowledges importance of testing, but focuses more on the white box testing. It suggests automating security analysis and metrics by using dynamic or static

tools. CLASP deals with creation of necessary documentation to install and operate the software safely and suggests reviewing the specifications from the developer's perspective in order to spot any ambiguities. In verification phase CLASP suggests penetration testing to ensure that all issues have been caught.

CLASP recommends verifying security attributes of resources to confirm that software is meeting previously defined standards. It suggests code signing to provide the stakeholders with a way to validate the origin of the software. Following the release CLASP states that reported vulnerabilities should be addressed by updating software.

2.3 Cigital's Security Touchpoints

Cigital's Security Touchpoints (Touchpoints) provides a set of best practices that have been gathered over the years out of the extensive industrial experience. Best practices are grouped together into seven touchpoints. Touchpoints recognize the importance of risk management and tries to bridge the gap by elaborating a Risk Management Framework (RMF) that supports the Touchpoints activities. Touchpoints are a mix of destructive and constructive activities. Destructive activities are attacks, exploits, and breaking software. These kinds of things are represented by the black hat. Constructive activities are about design, defence, and functionality and these are represented by the white hat. In order to make it easier for companies, different touchpoints are in ranking: 1. Code review, 2. Architectural risk analysis, 3. Penetration testing, 4. Risk-based security tests, 5. Abuse cases, 6. Security requirements, 7. Security operations.

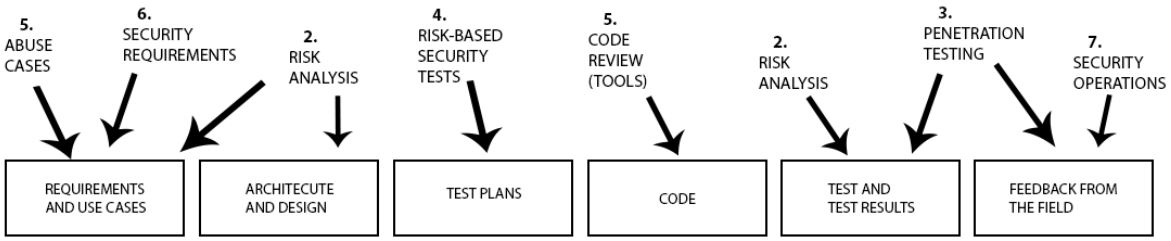

Figure 3 Software security best practices are applied to various software artefacts. (adapted from McGraw, 2006)

Touchpoints does not cover education before project launch. It is recognized that people should be trained about the particularities of the development environment, but there is no mandatory education to the personnel involved in building the secure software. A knowledge management framework is described to share software security knowledge among the project team.

Touchpoints describes an improvement program (McGraw 2006 p: 247-251) in order to adopt the best practices. This program assigns which part of the project will be done by whom, how the team will build and deploy it and also how they will continue to improve it over time. Improvement program also has a metric system put in place in order to demonstrate how well things are going from a security perspective. The improvement program will be tailored to the given business and technical situations.

Touchpoints advises abuse cases to be used in order to describe the system's behaviour under attack. Two critical activities of abuse cases are: creating an anti-requirements and creating an attack model. Anti-requirements are for describing what can go wrong and attack model is for describing how it can be achieved. Touchpoints also suggests creating a risk management framework (RMF) (McGraw 2006 p: 59) to identify

and keep track of risks over time as software project evolves. Extra security requirements are based on three sources: laws and regulations, commercial considerations and contractual obligations. Touchpoints also emphasize knowledge requirement as architectural risk analysis is knowledge intensive.

For architectural design the main focus is on threat modelling, but also risk analysis is introduced to identify risks in the system and mitigate them. Risk analysis consists of attack resistance analysis, ambiguity analysis and weakness analysis. Attack resistance analysis is meant to capture the checklist-like approach to risk analysis taken in Microsoft's STRIDE approach. Ambiguity analysis helps to uncover ambiguity and inconsistency and identify downstream. Weakness analysis is a sub process aimed at recognizing the impact of external software dependencies. Touchpoints also recommend a security expert to this phase.

Touchpoints emphasize the importance of testing by introducing risk-based security testing. Risk-based security testing is a mix of constructive and destructive activities that requires a black-and-white box approach. Testers must ground both the system's architectural reality and the attacker's mind-set. By identifying risks in the system and creating tests driven by those risks, a software security tester can properly focus on areas of code where an attack is likely to succeed. Touchpoints also suggests using automated tools as it is the best way to identify the most basic of implementation defects and it recommends penetration testing for a system in its final production environment. Touchpoints also acknowledges unit testing as an important part of security testing. Unit testing carries the benefit of breaking system security down into a number of discrete parts.

For release and deployment Touchpoints covers the importance of event-monitoring and event-logging as they will be effective during incident response operations.

2.4 Comparison

In this part we provide comparison between SDL, CLASP and Touchpoints in six different categories: education, project launch, risk analysis and requirements, architectural and detailed design, implementation and testing, release and deployment. Categories are implemented from SDL phases as CLASP and Touchpoints activities can be categorized similarly. Activities are SDL, CLASP and Touchpoints activities that each lifecycle recommends to fulfil in order to assure secure system.

2.4.1 Education

In Table 1 we compare three processes in education criteria. SDL and CLASP both emphasize education before project launch by instituting security awareness program and providing advanced education, but SDL goes one step further by measuring the knowledge gained from those activities. Touchpoints does not provide any activities for educating team members before the project launch.

Table 1 - Education

Activity	Description	SDL	CLASP	Touchpoints
Institute security awareness program	Ensure project members consider security to be an important project goal through training and accountability.	1	1	0
Provide advanced education	Members of the team that do not directly deal with security issues should be aware of the project's security practices.	1	1	0
Measure knowledge gained	Provided metrics are used to measure knowledge gained through training programs.	1	0	0
Sum		**3**	**2**	**0**

2.4.2 Project launch

In Table 2 we compare SDL, CLASP and Touchpoints in activities relating to project launch. All three processes recommend assembling a security team and monitoring implementation success. However, SDL and CLASP are different from Touchpoints by also recommending security advisor for the team. CLASP has the most activities regarding project launch as they also recommend instituting accountability and identifying global security policy. Touchpoints is unique by recommending improvement program.

Table 2 - Project launch comparison

Activity	Description	SDL	CLASP	Touchpoints
Assemble security team	Identification of the team that is responsible for tracking and managing security of the product	1	1	1
Appoint security advisor	Team member or external auditor will be appointed to be security advisor, who will review work of other team members	1	1	0
Monitor implementation success	A set of metrics is devised that product team can use to monitor their success in implementing the approach	1	1	1
Institute accountability	Team members will be accountable for performing activities to satisfactory level	0	1	0
Institute improvement program	A program which assigns which part of the project will be done by whom and how they will continue to improve it over time.	0	0	1
Identify global security policy	Provide a way to compare the security posture of different products across an organization.	0	1	0
Sum		**3**	**5**	**3**

2.4.3 Risk analysis and requirements

In Table 3 we compare the three processes in risk analysis and requirements criteria. SDL has the least activities in this stage of the project. It recommends threat modelling and specification of privacy requirements. Touchpoints and CLASP both suggest identifying attacker profile and usage of abuse cases and threat modelling. CLASP also advises identifying resources, trust boundaries, user roles and determining risk mitigation. Touchpoints, which has the most activities in this stage, suggests using anti-

requirements, attack model, risk management framework and also eliciting legal risks and knowledge requirement.

Tabel 3- Risk analysis and requirements comparison

Activity	Description	SDL	CLASP	Touchpoints
Identify resources and trust boundaries	Provide a structured foundation for understanding the security requirements of a system.	0	1	0
Identify user roles	Define user roles and the resources that the role can access.	0	1	0
Identify attacker profile	Identify potential groups that could be a threat as well as the gross resources one expects them to have.	0	1	1
Anti-requirements	Documenting the things that software should not do.	0	0	1
Abuse cases(misuse cases)	Use cases that are meant to detail common attempted abuses of the system.	0	1	1
Attack model	Given a set of requirements and a list of threats, cyclation through the list of known attacks is made and decided whether an attack applies to system under development	0	0	1
Threat modelling	Assess likely system risks by analysing the requirements and design.	1	1	1
Privacy requirements	Measures the sensitivity of the data that software will process from a privacy point of view.	1	0	0
Elicit legal and/or regulatory risk	Elicit and manage security from laws and regulations	0	0	1
Elicit knowledge requirement	Advanced knowledge is required before continuing to next phase of the development	0	0	1
Risk management framework	Risk management framework encompasses identifying, synthesizing, ranking, and keeping track of risks throughout software development.	0	0	1
Determine risk mitigation	Identify what risks could be considered, then identify solutions for addressing those risks.	0	1	0
Sum		**2**	**6**	**8**

2.4.4 Architectural and detailed design

In Table 4 we compare the three processes in architectural and detailed design activities. SDL and CLASP are more thorough than Touchpoints in this phase. They both suggest minimization of the attack surface, researching and assessing security posture of technology solutions and reviewing threat modelling. SDL and Touchpoints both recommend attack resistance analysis, but Touchpoints also recommends ambiguity analysis. CLASP is unique by recommending annotating class designs with security properties and applying security principles to design.

Table 4 - Architectural and detailed design comparison

Activity	Description	SDL	CLASP	Touchpoints
Minimization of attack surface	Specification of all entry points to a program in a structured way and minimization of those entry points	1	1	0
Research and assess security posture of technology solutions	Assess security risks in third-party components.	1	1	1
Annotate class designs with security properties	Elaborate security policies for individual data fields.	0	1	0
Review threat modelling	Assess likely system risks by analysing the requirements and design.	1	1	0
Perform attack resistance analysis	Identify general flaws using secure design literature and checklists	1	0	1
Apply security principles to design	Harden application design by applying security design principles.	0	1	0
Perform ambiguity analysis	The ambiguity analysis takes advantage of the multiple points of view afforded by multiple analysts to create a critical analysis technique.	0	0	1
Create data flow diagrams	Used to graphically represent a system	1	0	0
Sum		**5**	**5**	**3**

2.4.5 Implementation and testing

In Table 5 we compare SDL, CLASP and Touchpoints in implementation and testing criteria. SDL, which has the most activities in this phase, is unique by recommending coding and testing standards, fuzz testing and security push. It is similar to CLASP and Touchpoints by suggesting usage of automated tools and penetration testing. CLASP which has the least activities suggests integrating security analysis into source management process. Touchpoints focuses mainly on testing as it recommends risk-based security testing and unit testing.

Table 5 – Implementation and testing comparison

Activity	Description	SDL	CLASP	Touchpoints
Apply coding and testing standards		1	0	0
Implement automated tools		1	1	1
Perform penetration testing	Method of evaluating the security of a computer system or network by simulating an attack from malicious outsiders and malicious insiders	1	1	1
Perform fuzz testing	Software testing technique that involves providing invalid, unexpected, or random data to the inputs of a computer program.	1	0	0
Integrate security analysis into source management process	Automate implementation-level security analysis and metrics collection.	0	1	0
Perform risk-based security testing	Covers functionality testing and emulates the steps that an attacker will take when breaking a target system.	0	0	1
Perform security push	Team-wide focus on threat model updates, code review, testing, and documentation scrub.	1	0	0
Unit testing	Method by which individual units of source code are tested to determine if they are fit for use.	0	0	1
Sum		**5**	**3**	**4**

2.4.6 Release and deployment

In Table 6 we compare the three processes in activities relating to release and deployment. SDL and CLASP have the most activities in this stage as they both suggest conducting independent review of the software and updating it regularly. CLASP also recommends code signing and SDL suggests evaluating reports of vulnerabilities. Touchpoints, which has the least activities, suggests event-monitoring and event-logging after the release.

Table 6 – Release and deployment

Activity	Description	SDL	CLASP	Touchpoints
Conduct independent review of software	Independent review of the software conducted by the security team.	1	1	0
Perform code signing	Provide the stakeholder with a way to validate the origin and integrity of the software.	0	1	0
Evaluate reports of vulnerabilities		1	0	0
Update software		1	1	0
Perform event-monitoring	Process of collecting, analysing, and signalling event occurrences to subscribers.	0	0	1
Perform event-logging	Provides system administrators with information useful for diagnostics and auditing	0	0	1
Sum		**3**	**3**	**2**

2.5 Summary

Security development models play an important role in developing a secure system and as we can see they all focus on different stages in development process. SDL has the most activities in education, design and implementation. CLASP concentrates mainly on project launch and risk analysis and Touchpoints emphasizes the importance of risk analysis and security requirements. Choosing the process depends on what development stage is the most important from stakeholder perspective. We selected risk analysis phase for further analysis, because in this phase consequences of different threats are assessed and the activities carried out in this phase give stakeholders the way to take appropriate response to mitigate the risks in their software making it in our opinion the most important phase of software development.

Chapter 3. Risk Analysis and Requirements: Survey Design

In this chapter we have conducted a systematic literature review (SLR). The SLR was carried out by effectuating the following activities: defining research question, source selection, studies selection process and information extraction.

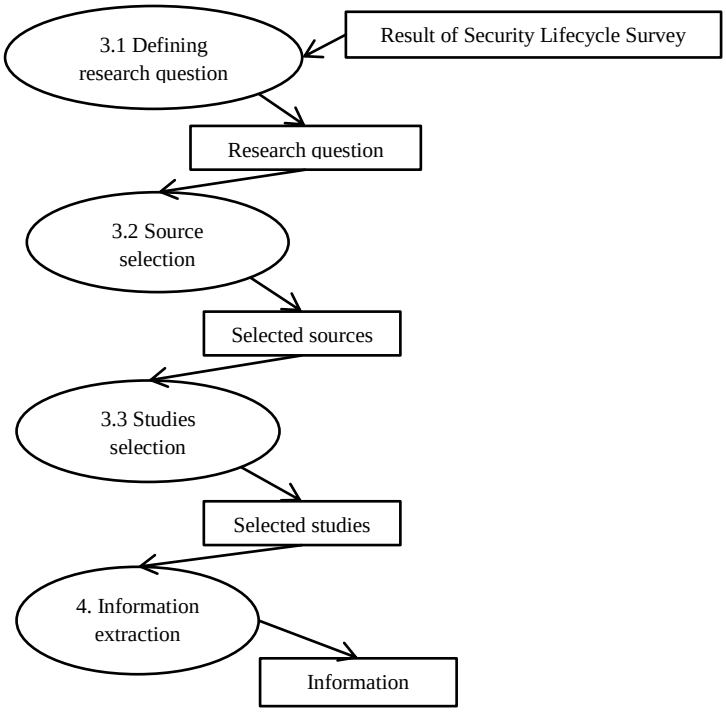

Figure 4 – Design of systematic literature review

3.1 Research question

We defined the following research question: "What are the current practices and methods for security risk management?" After carrying out this SLR we expect to find out, which activities current security risk management methods cover in Table 3 and also provide a link between found methods and security development processes.

3.2 Source selection

We picked sources which are of the recognized quality within the research community and possibly can contain answers for our research question. These sources are:

- Requirements Engineering Journal (REJ)
- Computers & Security – Journal (COSE)
- Information Security Technical Report (ISTR)
- International Conference on Advanced Information Systems Engineering (CAISE)
- International Conference on Availability, Reliability and Security (ARES)
- European Conference on Information Systems (ECIS)
- Information Security Journal (ISJ)
- International Journal of Secure Software Engineering (IJSSE)

In the selected sources, we experimented with various search string criteria. That which eventually retrieved the highest number of useful results was:
(security risk management) AND (methods OR study OR review OR practices).

3.3 Information extraction

Having defined the source selection, we implemented procedures to identify those studies that provided direct evidence to the research question. First we implemented criteria that studies have to be published in last 4 years to be current practices and methods for security risk management. Older studies may still be relevant at the present time, but as we had limited time and manpower we decided to focus on the studies published in last 4 years. After that we found initial studies by reading the title, abstract and introduction. Studies which were not related to the research question were put aside.

Next we reviewed the studies that had been selected and found out if they contain activities in Table 3. Out of those studies we selected two methods Secure i*(Elahi, *et al*., 2010) and SecReq (Houmb, *et al*., 2009).

After finding only two methods we wanted to expand our literature review and to expand it, we went through the references of our found methods. The outcome was the selection of five other methods for security risk management. Those methods were Secure Tropos (Giorgini, *et al*., 2007; Mouratidis, *et al*., 2007), UMLsec (Jürjens, 2002), SQUARE (Suleiman & Svetinovic, 2012; Stehney & Mead, 2005), ISSRM domain model (Mayer, *et al*., 2006; Alcalde, *et al*., 2009; Mayer, *et al*., 2008) and Misuse cases (Sindre & Goguen, 2004).

Table 7 - Summary of the studies selected.

Sources	REJ	COSE	ISTR	CAISE	ARES	ECIS	ISJ	IJSSE
Total results	23	11	21	5	12	36	40	23
Results selected	3	0	1	0	2	0	0	3
Selected studies	(Guerses, *et al*., 2011), (Elahi, *et al*., 2010), (Houmb, *et al*., 2009)		(Jirasek, 2012)		(Beckers, 2012), (Jakoubi, 2010)			(Islam, *et al*., 2013), (Nhlabatsi, *et al*., 2010), (Khan, 2012)

3.4 Threats to validity

The main threat to validity is that whether we have failed to find all the relevant studies, although we have selected a wide range of conferences and journals, there may still exist relevant papers that we have not included. This may be caused by faulty search string criteria or limited source selection. Another threat may come from different interpretation of the methods selected. As selected methods' activities may not accord exactly to Table 3 definitions, it may result in some studies, which interpret the accordance of the methods to Table 3 differently from our study.

3.5 Summary

To carry out the systematic literature review, we defined our research question: “What are the current practices and methods for security risk management?” After that we picked sources that are of recognised quality and selected 7 different methods out of the studies that we found. These 7 methods are Secure i*, SecReq, Secure Tropos, UMLsec, SQUARE, ISSRM domain model and Misuse cases. In the next chapter we will extract information from found methods to compare them to each other and provide an aggregate view of those seven security risk management methods.

Chapter 4. Risk analysis and Requirements: Result Analysis

In this part we have created Table 8 from Table 3 to see which activities from security development processes each method covers. We have also composed reviews of selected methods and description how each method covers its activities.

Table 8 – Comparison of methods for security risk management

Activity	Definition	Secure i*	SecReq	Secure Tropos	UMLsec	SQUARE	ISSRM domain model	Misuse cases	Sum
Identify resources and trust boundaries	Provide a structured foundation for understanding the security requirements of a system.	1	1	1	1	1	1	1	**7**
Identify user roles	Define user roles and the resources that the role can access.	1	1	1	1	1	0	0	**5**
Identify attacker profile	Identify potential groups that could be a threat and define their skillset and motivation for the attack as well as the gross resources one expects them to have.	1	0	0	0	0	1	1	**3**
Anti-requirements	Documenting the things that software should not do.	0	0	0	0	0	0	0	**0**
Abuse cases(misuse cases)	Use cases that are meant to detail common attempted abuses of the system.	0	0	0	0	0	0	1	**1**
Attack model	Model that shows goals and methods that attacker may use.	1	0	0	0	1	1	1	**3**
Threat modelling	Assess likely system risks by analysing the requirements and design.	1	0	1	1	1	1	1	**5**
Privacy requirements	Measures the sensitivity of the data that software will process from a privacy point of view.	0	0	1	0	0	0	0	**1**
Determine risk mitigation	Identify what risks could be considered, and then identify solutions for addressing those risks.	1	0	1	0	1	1	1	**5**
Sum		**6**	**2**	**5**	**2**	**5**	**5**	**6**	

4.1 Secure i*

The i* framework provides the basic setting for representing vulnerabilities that are brought by actions and assets and propagating them through the decomposition and dependency links to other elements of model (Elahi, *et al*., 2010). The modelling process consists of five views: requirements view, vulnerabilities view, attackers template view, attackers' profile view and countermeasures view. Identification of resources and user roles is done by requirements view that shows stakeholders and actors with their goals, the tasks to achieve those goals, required resources and the dependencies among them. Threat modelling is done by vulnerabilities view that extends the requirements view by adding vulnerabilities that tasks and resources bring to the system and what impact these vulnerabilities have to the system. Attack model is in the attackers' template view that represents how an attacker can exploit the vulnerabilities. Attacker profile is identified in

attackers' profile view. The attackers' profile view captures the attacker's goals, skills and behaviour. Risk mitigation is done in countermeasure view that shows the security solutions adopted by actors to protect the system as well as their impacts on attacks and vulnerabilities.

4.2 SecReq

SecReq is a security requirements elicitation and tracing method built on the CC standard, The Heuristic Requirements Assistant (HeRA) tool, and UMLsec. The elicitation part consists of five steps that take a developer through a series of refinement steps starting from system objectives and functional requirements and ending with specific security requirements at an early stage (Houmb, *et al*., 2009). The SecReq method consists of six steps. In first step we must identify resources and trust boundaries by specifying security objectives from system objectives and functional requirements. These requirements are refined from security objectives. In step two we need to identify user roles by distinguishing users or groups of end-users so they are properly authenticated to the system. In step 3 we refine security objectives to sub security objectives. Sub security objectives are a refinement of security objectives and are a detailed description of the relevant part of the secure environment for end-users of the system specified by the security objective (Houmb, *et al*., 2009). Step 4 takes the result from Step 3 and refines the sub security-objectives into security requirements. Step 5 takes the result from Step 4 and refines it to requirements that are specific, measurable, achievable, realisable and traceable. Throughout Steps 1–5 the HeRA tool observes requirements inputs and raises warning and hints when security-related input is detected. In step 6 we capture the results of step 5 and integrate them into UML diagrams by using UMLsec stereotypes.

4.3 Secure Tropos

Tropos is a software development methodology tailored to describe both the organisational environment of a system and the system itself. Secure Tropos extends the original Tropos methodology with some new concepts: a security constraint, secure entities, ownership, provisioning, trust of permission, trust of execution, delegation of permission, delegation of execution, secure trust of permission, secure delegation of permission (Giorgini, *et al*., 2007). Secure Tropos starts with identifying user roles, resources and trust boundaries is done by modelling stakeholders and actors with their goals, producing an actor diagram and extending the actor diagram with trust and ownership relationships. Next we identify privacy requirements by modelling the security constraints to identify secure capabilities for each actor. Threat modelling is done by security reference modelling. Security reference modelling involves identification of security needs, threats and vulnerabilities and also possible solutions to the security problems.

4.4 UMLsec

The Unified Modeling Language (UML) is the industry-standard in object-oriented modelling. It offers an unprecedented opportunity for high-quality critical systems development that is feasible in an industrial context (Jürjens, 2002). UMLsec is an extension for Unified Modeling Language that allows to express security relevant information within the diagrams in a system specification (Jürjens, 2002). UMLsec's security requirements are encapsulated in UML stereotypes, tags in the UMLsec profile

and constraints. UMLsec identifies resources and trust boundaries using statecharts, sequence diagrams and class diagrams. Statecharts give the object behaviour, while class diagrams define the static structure of the system and sequence diagrams ensure correctness of security-critical interactions between objects (Jürjens, 2002). UMLsec defines user roles by defining actors using activity diagrams and showing their rights to access a protected resource. Threat modelling is done using threat scenarios in deployment diagrams.

4.5 SQUARE method

The security quality requirements engineering (SQUARE) method is a security requirements engineering method developed by Nancy Mead. SQUARE consists of nine steps: agree on definitions, identify security goals, develop artefacts to support security requirements definitions, perform risk assessment, select requirements elicitation technique, elicit the security requirements, categorize the security requirements, prioritize the security requirements, and inspect the security requirements. The steps include identifying suitable techniques to systematically perform each step (Mead, *et. al.*, 2005).

SQUARE specifies five artefacts: system architecture diagrams, use cases, use-case diagrams, attack trees and security template. Architecture diagram identifies resources and trust boundaries. Resources are defined by security goals, which can be derived from business application goals or potential threats to assets. and user roles are demonstrated in use cases and use-case diagrams. In SQUARE threat modelling is done by the security template. The security template is a modified version of the Software Engineering Institute's security template. The template specifies: source - specifies the weakness, threat or vulnerability point, stimulus - specifies the first action triggering the event that reveals the security threat, artefact - specifies the data or system services that attackers want to attack, specifies the status of the environment before an attack, action - specifies the actions that attackers plan to perform by exercising specific vulnerability, consequence - specifies the results or the effects of an attack. Attack model is described by attack trees that capture the security weakness points in the system and show us the goals and methods that attacker may use. Risk mitigation is done by using National Institute of Standards and Technology risk assessment method (Stoneburner, *et al.*, 2002). This method has five steps: threats identification, vulnerabilities identification, likelihood analysis, impact analysis and risk determination.

4.6 ISSRM domain model

The objective of ISSRM is to protect assets of an organisation, from all harm to IS security which could arise accidentally or deliberately, by using a risk management approach. Its domain model aims at presenting the different concepts involved and their mutual relationships. ISSRM core concepts are organised in three categories: asset-related concepts, risk-related concepts and risk-treatment related concepts (Alcalde, *et al.*, 2009).

In first category we must identify resources and trust boundaries by defining which assets are important to protect and what are their security needs. In second category we must identify attacker profile by describing threat agents and their potential attacks to an asset. Attack model is formed by describing the vulnerabilities that an attacker exploits and the effect that the attack will have on an asset. In third category threat modelling is done by analysing the security requirements and linking them to found risks. Furthermore we determine risk mitigation by describing how to treat the identified risks.

4.7 Eliciting security requirements with misuse cases

Misuse Cases is described as a sequence of actions, including variants that a system or other entity can perform, interacting with misusers of the entity and causing harm to some stakeholder if the sequence is allowed to complete (Sindre & Opdahl, 2004).

Eliciting security requirements with misuse cases consists of five steps. In first step identification of resources is done by identifying critical assets in the system. In second step security goals are added to each asset identified in first step. In third step attacker profile is identified by identifying misusers that may harm the system or its environment and also attack model is provided by describing attackers' goals and methods. Threat modelling is done in fourth step, where risks are identified and analysed. In fifth step risk mitigation is done by defining countermeasures. Misuse cases compliment identifying security threats, which can be described as misuse cases and misusers and also security requirements can be described by misuse cases.

4.8 Comparison

In this part we provide comparison between an extended i* meta-model, SecReq, Secure Tropos, UMLsec, SQUARE, ISSRM domain model and Misuse cases. It is clear that all of these methods provide a structured foundation for understanding the security requirements of a system. All of these methods, except ISRRM domain model and Misuse cases, identify user roles. Attacker profile is identified in Secure i*, ISSRM domain model and Misuse cases. Anti-requirements are not included in any of these methods. Abuse cases are used in method Misuse cases. All of these methods, except SecReq, provide threat modelling, but Secure i*, SQUARE, ISSRM and Misuse case provide us also with attack models. Secure Tropos is the only method that measures the sensitivity of the data that software will process from a privacy point of view. ISSRM and Secure Tropos also acknowledge that data's confidentiality and integrity are important, but no actual measurement is given. UMLsec and SecReq are the two methods that do not provide solutions for addressing security risks of a system.

4.9 Security development models and risk management methods

In Table 3 we can see that in risk analysis and requirements stage SDL consists of threat modelling and privacy requirements. In Table 8 the only method that covers these activities is Secure Tropos. Secure i*, UMLsec, SQUARE, ISSRM domain model and Misuse Cases can also be used in SDL as they cover threat modelling. CLASP consists of seven activities that are also in Table 8. These activities are identify resources and trust boundaries, identify user roles, identify attacker profile, abuse cases, threat modelling and determining risk mitigation. There is no method in Table 8 that covers all of these activities. Misuse Cases covers all other activities requested for CLASP, except identify user roles. Secure i* covers everything except abuse cases. Both Secure Tropos and SQUARE cover four activities required in CLASP. Those activities are identify resources and trust boundaries, identify user roles, threat modelling and determining risk mitigation. ISSRM domain model also covers four activities, but instead of identifying user roles, attacker profile is needed. UMLsec consists of three activities that are also requested for CLASP: identify resources and trust boundaries, identify user roles and threat modelling. SecReq cover the least activities for CLASP as it identifies resources, trust boundaries and

user roles. Touchpoints consists of four activities that are also in Table 8: identify attacker profile, anti-requirements, abuse cases and attack model. None of the methods we have chosen cover anti-requirements. However, all other Touchpoints activities can be covered with Misuse cases, which combined with anti-requirements cover all Touchpoints activities. Identification of attacker profile and an attack model is also provided in Secure i* and ISSRM domain model. SQUARE can also be used as it provides an attack model. Anti-requirements are used in the work of van Lamsweerde (2004), where he introduces anti-models and anti-goals to document the things that software should not do.

4.10 Summary

Security risk management methods are important part of development process for secure software, however the choice between the methods can be rather difficult. Difficulty comes from the structure of the methods. Even if the methods execute the same activity, they may do so by using different artefacts, definitions and means to do so. For example in Secure i* attack model_is in the attackers' template view that represents how an attacker can exploit the vulnerabilities and in SQUARE attack model is described by attack trees that capture the security weakness points in the system and show the goals and methods that attacker may use. Stakeholders are the ones who have to choose which technique is the best for system under development and Table 8 can only serve as a guideline to their selection process.

Chapter 5. Aggregate view on the Risk Analysis and Requirements

Completing our research question in chapter 4 gave us the understanding how found practices and methods can be used successively in order to fulfil the activities required in risk analysis phase. In this chapter we have created Table 9 to show which type of information is given from each security risk management method for completing the activities listed in Table 8. Additionally we explain in this chapter how the methods give the type of information.

We have divided information types into three groups: conceptual definition, application guidelines and analysis techniques. Conceptual definition defines the meaning of terms used in risk analysis and requirements activity. Application guidelines give rules how to accomplish these activities and finally, analysis techniques give us a way of carrying out a particular activity.

Identifying resources and trust boundaries. To identify resources and their trusted boundaries, one can use the ISSRM domain model, where the conceptual base for *assets and their security criteria is defined*. This can guide the combined application of SecReq and SQUARE. For instance resources and trust boundaries in SecReq are *defined as security objectives,* which are *derived from system objectives and functional requirements*. In SQUARE resources are considered for the *security goals*. They are *elicited from business application goals and through consideration of protected threats*. Analysis techniques for this activity include Secure i*, Secure Tropos, Misuse Cases and UMLsec. Resources and trust boundaries are identified in Secure i* by modelling required *resources and goals*. Similarly it is done in Secure Tropos. Misuse Cases treat resources as *critical assets in the system*. UMLsec suggests means to define *stereotypes together with tags* in order to give the object behaviour and interactions between objects.

Identifying user roles. Application guidelines to identify user roles are given in SQUARE, which demonstrates *how to use modelling techniques (e.g. use cases) to identify actors and processes.* Analysis techniques for identifying user roles include Secure i*, Secure Tropos and UMLsec. In Secure i* user roles are identified *modelling stakeholders and actors with their goals and the tasks to achieve those goals*. Similarly it is done in Secure Tropos, but they extend it with *trust and ownership relations*. UMLsec defines *the actors using activity diagram* and shows their *rights to access a protected resource*.

Identifying attacker profile. ISSRM domain model gives the application guidelines for identifying attacker profile by *guiding the definition of threat agent*. Secure i* and UMLsec provide the analysis techniques for this activity. In Secure i* attacker profile is identified by defining *the actor that can exploit the vulnerabilities to have a negative impact towards the system*. In Misuse Cases the attacker is defined as *misuser that wants to misuse the system* under consideration.

Abuse cases (misuse cases). Analysis technique for abuse cases is given by Misuse Cases that *identifies security threats and security requirements,* which then can be *described by misuse cases*.

Attack model. Application guidelines for attack model are given by ISSRM domain model and Misuse Cases. The ISSRM domain model describes *vulnerabilities of*

the system that an attacker exploits and the effects that an attack has on an asset. In Misuse Cases *attackers' goals and methods* are described to provide an attack model.
Secure i* and SQUARE provide the analysis techniques for composing an attack model. In Secure i* attack model is depicted as *attackers' template view* that represents how *an attacker can exploit the vulnerabilities in the system*. SQUARE describes *attack trees* that *capture the security weakness points in the system and show us the goals and methods that attacker may use*.

Threat modelling. Secure i*, SQUARE, ISSRM domain model and Misuse Cases give the application guideline for threat modelling. In Secure i* threat modelling is done by describing *vulnerabilities that tasks and resources bring* to the system and also describing *the impact that these vulnerabilities have* to the system. In SQUARE threat modelling is done by *specifying the weakness in the system, threat that the weakness brings, the action triggering the attack, the data or system service that attacker wants to attack, the status before the attack, the attackers plan and consequences of the attack*. Misuse Cases says that to do threat modelling, it is necessary to *identify and analyse found risks*. ISSRM domain model defines *the security requirements of the system and links them to found risks*. Analysis techniques for threat modelling are provided by Secure Tropos and UMLsec. Secure Tropos does security reference modelling that consists of *identifying security needs, threats and vulnerabilities*. UMLsec covers threat modelling by *threats scenarios in deployment diagrams*.

Privacy requirements. Privacy requirements are carried out only in Secure Tropos, which gives the analysis technique for it. Privacy requirements are covered by *modelling the security constraint to identify secure capabilities for the actors*.

Determine risk mitigation. Guidelines for determining risk mitigation are given by the ISRRM domain model and Misuse Cases. ISSRM domain model insists on *description how to treat the identified risks*. In Misuse cases *countermeasures are defined to found risks*. Analysis techniques are given by Secure i*, Secure Tropos and SQUARE. Secure i* determines risk mitigation by showing *the security solutions adopted by actors to protect the system* and also *their impact on attacks and vulnerabilities*. In Secure Tropos risk are mitigated by *security reference modelling*, where *possible solutions to security problems* are shown and in SQUARE threats and vulnerabilities are identified using *National Institute of Standards and Technology risk assessment method* (Stoneburner, *et al*., 2002). It includes *threat's likelihood analysis, impact analysis and risk determination*.

Table 9 - Aggregate view on the Risk Analysis and Requirements

	Identify resources and trust boundaries	Identify user roles	Identify attacker profile	Abuse cases(misuse cases)	Attack model	Threat modelling	Privacy requirements	Determine risk mitigation
Conceptual definition	ISSRM Domain model							
Application guidelines	SecReq, SQUARE	SQUARE	ISSRM domain model		ISSRM domain model, Misuse Cases	Secure i*, SQUARE, ISSRM domain model, Misuse Cases		ISSRM domain model ,Misuse Cases
Analysis techniques	Secure i*, Secure Tropos, Misuse Cases, UMLsec	Secure i*, Secure Tropos, UMLsec	Secure i*, Misuse Cases	Misuse Cases	Secure i*, SQUARE	Secure Tropos, UMLsec	Secure Tropos	Secure i*, Secure Tropos, SQUARE

Chapter 6. Related Work

Comparison of risk management methods has also been done by Fabian *et al*., (2009). They presented a conceptual framework for security requirements engineering that established a common vocabulary and made interrelations between different concepts used in security engineering. Using the presented framework they compared different risk management methods. SQUARE, UMLsec, Secure Tropos and Secure i* are methods that are provided in their comparison as well as ours. They divided their methods into six different approaches: multilateral approach, UML-based approaches, goal-oriented approaches, problem frame-based approaches, risk analysis-based approaches and common criteria-based approaches. Similar division can be seen in our aggregate view Table 9, except we do not assign a method into one category, but rather appoint a method into a category for each activity. Another comparison has been done by Kalloniatis *et al*.,(2004). They compare requirements engineering methods under the scope of helping eGovernment application development. Secure i* and Tropos are methods that are covered in their comparison as well as ours. They conclude with the need for a combination of methods, which would cover all aspects of security requirements modeling, which we trying to achieve by giving stakeholders an aggregate view of our chosen methods. SecReq (Houmb, *et al*., 2009) is also a combination of risk management methods. They use Heuristics, Common Criteria, and UMLsec to provide one complete risk management method. Difference comes from the quantity of methods and the criteria being followed. While in Houmb *et al*.,(2004) work they formulate their own criteria that methods have to fulfil, we follow the criteria given by security development models. Despite their existing comparisons of different risk management methods, we did not find any related works that connect risk management methods to security development processes.

Chapter 7. Conclusion

Security development models like SDL (Lipner & Howard, 2005), CLASP (Graham, 2006) & Touchpoints (McGraw, 2006) are made to improve the security of software products by recommending series of security activities and although they serve the same purpose, the activities they recommend vary greatly depending on the phase of development. In this thesis we gave the answer to the following research question: what are the differences between SDL, CLASP & Touchpoints? Current security risk management methods are also composed to improve the security of the software, but their coverage of the development is much smaller as they usually cover only one phase of the development. To see which activities risk management methods cover, we answered the question: what are the current practices and methods for security risk management? After finding out current security risk management methods, we developed an aggregate view on risk analysis and requirements to find the similarities between found methods, so it would be easier to combine them in providing a secure software system.

Our literature review was limited to one phase of security development processes, because of limited time and manpower. In future, other phases of security development processes (e.g. architectural and detailed design) can be covered similarly with a literature review. That would give us a complete overview of methods that are needed to complete these security development processes. Also aggregate view on risk analysis and requirements can be completed by finding new methods that would give conceptual definitions or application guidelines to activities, where they are missing.

References

Alcalde, B., Dubois, E., Mauw, S., Mayer, N., & Radomirovic, S. (2009). Towards a Decision Model Based on Trust and Security Risk Management. In Proc. Seventh Australasian Information Security Conference (AISC 2009), Wellington, New Zealand.

Beckers, K. (2012. Using Security Requirements Engineering Approaches to Support ISO 27001 Information Security Management Systems Development and Documentation. *Availability, Reliability and Security (ARES), 2012 Seventh International Conference on* (pp 242-248)

Davis, N. (2006). Secure Software Development Life Cycle Processes, Retrieved December 13, 2011, from https://buildsecurityin.us-cert.gov/bsi/articles/knowledge/sdlc/326-BSI.html

Elahi, G., Yu, E., &, Zannone, N. (2010). A Vulnerability-Centric Requirements Engineering Framework: Analyzing Security Attacks, Countermeasures, and Requirements Based on Vulnerabilities. *Requirements Engineering*: Vol. 15, Issue 1 (pp 41-62).

Fabian, B., Gürses, S., Heisel, H., Santen, T., & Schmidt, H. (2009). A comparison of security requirements engineering methods. *Requirements Engineering*: Vol 15, Issue 1, (pp 7-40)

Giorgini, P., & Mouratidis, H. (2007). Secure Tropos: A Security-oriented Extension of the Tropos Methodology, Haralambos Mouratidis, Paolo Giorgini.

Giorgini, P., Mouratidis, H., & Zannone, N. (2007). Modelling Security and Trust with Secure Tropos. *Integrating Security and Software Engineering: Advances and Future Visions :* (pp 160-189).

Graham, D. (2006). Introduction to the CLASP Process. Retrieved December 13, 2012, from https://buildsecurityin.us-cert.gov/bsi/articles/best-practices/requirements/548-BSI.html

Guerses, S., Seguran, M., & Zannone, N. (2011). Requirements engineering within a large-scale security-oriented research project: lessons learned. *Requirements Engineering*: Vol. 18, Issue 1 (pp 43-66).

Houmb, S. H., Islam, S., Knauss, E., Jürjens, J., & Schneider, K. (2009). Eliciting security requirements and tracing them to design:an integration of Common Criteria, heuristics, and UMLsec. *Requirements Engineering*: Vol. 15, Issue 1 (pp 63-93).

Islam, S., Mouratidis, H., Kalloniatis, C., Hudic, A., & Zechner, L. (2013). Model Based Process to Support Security and Privacy Requirements Engineering. *International Journal of Secure Software Engineering:* Vol. 3, Issue 3 (pp 1-22).

Jakoubi, S. (2010). A Formal Approach Towards Risk-Aware Service Level Analysis and Planning. *Availability, Reliability, and Security, 2010. ARES '10 International Conference on* (pp 180-187)

Jirasek, V. (2012). Practical application of information security models. *Information Security Technical Report:* Volume 17, Issues 1–2 (pp 1-8).

Jürjens, J. (2002). UMLsec: Extending UML for Secure Systems Development. *Lecture Notes in Computer Science*: Vol. 2460 (pp 412-425).

Kalloniatis, C., Kavakli, E., & Gritzalis, S.(2004). Security Requirements Engineering for e-Government Applications: Analysis of Current Frameworks. *Electronic Government Lecture Notes in Computer Science*: Vol. 3183 (pp 66-71).

Khan, K. M. (2012). Software Security Engineering: Design and Applications. *International Journal of Secure Software Engineering:* Vol. 3, Issue 1(pp 62-63).

Lipner, S. & Howard, M. (2005).The Trustworthy Computing Security Development Lifecycle. Retrieved December 13, 2012, from http://msdn.microsoft.com/en-us/library/ms995349.aspx

Mayer, N., Dubois, E., Matulevičius, R., & Heymans, P. (2008). Towards a Measurement Framework for Security Risk Management.

Mayer, N., Heymans, P., & Matulevičius, R. (2006). Design of a Modelling Language for Information System Security Risk Management.

McGraw, G. (2006). Software Security: Building Security In. Addison-Wesley.

Microsoft Developer Network. (2012) . Retrieved December 13, 2012, from http://msdn.microsoft.com/en-us/library/windows/desktop/cc307748.aspx

Microsoft. (2012). Retrieved December 13, 2012, from http://www.microsoft.com/security/sdl/default.aspx

Nhlabatsi, A., Nuseibeh, B., & Yu, Y. (2010). Security Requirements Engineering for Evolving Software Systems: A Survey. *International Journal of Secure Software Engineering:* Vol. 1, Issue 1 (pp 54-73).

OWASP 1. (2012). Retrieved December 13, 2012, from https://www.owasp.org/index.php/CLASP_Concepts

OWASP 2. (2005). The CLASP Application Security Process. Retrieved December 13, 2012, from https://buildsecurityin.us-cert.gov/bsi/100/version/1/part/4/data/CLASP_ApplicationSecurityProcess.pdf?branch=main&language=default

Sindre, G., & Opdahl, A., L. (2004). Eliciting security requirements with misuse cases. *Requirements Engineering*: Vol 10, Issue 1, (pp 34-44)

Stehney, T., & Mead, N., R. (2005). Security Quality Requirements Engineering (SQUARE) Methodology.

Stoneburner, G., Goguen, A., & Feringa, A. (2002). Risk Management Guide for Information Technology Systems.

Stride Chart. (2007). Retrieved December 13, 2012, from http://blogs.msdn.com/b/sdl/archive/2007/09/11/stride-chart.aspx

Suleiman, H., & Svetinovic, D. (2012). Evaluating the effectiveness of the security quality requirements engineering (SQUARE) method: a case study using smart grid advanced metering infrastructure. *Requirements Engineering*, April 2012.

Tøndel, I. A., & Jaatun, M. G., & Meland, P. H. (2008). Security Requirements for the Rest of Us. *IEEE Software,* 20-27.

van Lamsweerde. A. (2004). Elaborating Security Requirements by Construction of Intentional Anti-Models. In Proceedings of the 26th International Conference on Software Engineering (ISCE ’04). IEEE Computer Society, Washington, DC, USA, 148-157.”

Non-exclusive licence to reproduce thesis and make thesis public

I, Silver Jürimäe

(date of birth: 04.07.1991),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

A Literature Survey of the Development Processes for Secure Software,

supervised **by Dr. Raimundas Matulevičius,**

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **14.05.2014**