jamk.fi

Test performance and security measurements in software development

Mikko Sara

Master's Thesis
June 2019

School of Technology, Communication and Transport
Information and Communication Technology
Degree Programme in Cyber Security

jamk.fi

Description

Author(s)	Type of publication	Date
Sara, Mikko	Master's thesis	June 2019
		Language of publication: English
	Number of pages 91	Permission for web publication: Yes

Title of publication

Test performance and security measurements in software development

Degree programme

Information and Communication Technology, Master's programme in Cyber Security

Supervisor(s)

Saarisilta, Juha; Saharinen, Karo

Assigned by

Finnish Defence Forces Logistics Command, Tikkanen, Raimo

Abstract

The main goal of the thesis was to study cyber security implementation in different software development frameworks with two different research methods. The first method was a survey aimed at industry supplemented by the second method, literary research answering the research question. The thesis was assigned by the Finnish Defense Forces Logistics Command, which utilized the results of the work in the software procurements bidding competitions.

The research project proceeded by first discovering the sources, followed by a literature research based on the sources. The content of the survey aimed at industry was designed against a certain ICT architecture to support the findings of the literature research. The survey was conducted anonymously for domestic and foreign industries.

The aim of the study was to build a situational awareness of how the industry sees secure software development and find alternative models for the secure software development framework. The research achieved the goals set by the Finnish Defence Forces, and the research clearly showed the facts received from the industry and the facts produced by the literature research.

In conclusion, there are different methods for performing safe software development; however, there is no standardized model or standard, which would be applicable to every software development framework and every company's quality system and processes.

Keywords/tags

Software development, security tools, security testing, secure code development

Miscellaneous

Research permit AO20582

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Sara, Mikko	Opinnäytetyö, ylempi AMK	Kesäkuu 2019
		Julkaisun kieli: Englanti
	Sivumäärä 91	Verkkojulkaisulupa myönnetty: Kyllä

Työn nimi

Test performance and security measurements in software development

Tutkinto-ohjelma

Information and Communication Technology, Master's programme in Cyber Security

Työn ohjaaja(t)

Juha Saarisilta; Karo Saharinen

Toimeksiantaja(t)

Puolustusvoimien logistiikkalaitos, Raimo Tikkanen

Tiivistelmä

Opinnäytetyön päätavoitteena oli tutkia tietoturvan toteutumista eri ohjelmistokehitysmenetelmissä kahdella eri tutkimusmenetelmällä. Teollisuudelle suunnatulla kyselytutkimuksella tavoiteltiin vastausta tutkimuskysymykseen, jota täydennettiin kirjallisuustutkimuksella. Opinnäytetyön toimeksiantajana toimi Puolustusvoimien logistiikkalaitos, joka hyödynsi työn tuloksia ohjelmistohankintojen kilpailutuksissa.

Työn toteutus eteni eri lähteiden kartoittamisella, jonka jälkeen toteutettiin kirjallisuustutkimus perustuen lähteisiin. Teollisuudelle suunnatun kyselytutkimuksen sisältö suunniteltiin tietynlaista ICT -arkkitehtuuria vastaan sekä myös tukemaan kirjallisuustutkimuksessa tehtyjä havaintoja. Kyselytutkimus suunnattiin anonyymina sekä kotimaiselle että ulkomaalaiselle teollisuudelle.

Tutkimuksen tavoitteena oli rakentaa tilannekuva siitä, miten teollisuus näkee ohjelmistokehityksen tietoturvan sekä kartoittaa vaihtoehtoisia malleja turvalliseen ohjelmistokehitykseen. Tutkimuksilla saavutettiin työn toimeksiantajan asettamat tavoitteet, joista oli selkeästi nähtävissä sekä teollisuuden näkemykset että kirjallisuustutkimuksen tuottamat faktat.

Johtopäätöksenä voitiin todeta, että ohjelmistojen turvalliseen kehittämiseen on olemassa erilaisia menetelmiä, mutta sellaista vakioitua mallia tai standardia aiheeseen ei ole olemassa, mikä olisi sovellettavissa jokaiseen ohjelmistokehitysmenetelmään, projektiin sekä eri yrityksissä toimiviin laatujärjestelmiin että prosesseihin.

Avainsanat

Ohjelmistokehitys, tietoturvatestaus, ohjelmiston turvallinen kehittäminen

Miscellaneous

Tutkimuslupa AO20582

Contents

Figures

Tables

Acronyms

API	Application programming interface
APP	Application
ASVS	Application Security Verification Standard
CD	Compact disc
CERT	Computer Emergency Response Team
CSRF	Cross-Site Request Forgery
CWE	Common Weakness Enumeration
DLL	Dynamic Link Library
GQM	Goal, Question and Metric
HMI	Human Machine Interface
HTML	Hypertext Markup Language
HTTPS	Hypertext Transfer Protocol Secure
HQL	Hibernate Query Language
ICT	Information and Communication Technology
IDS	Intrusion Detection System
InfoSec	Information Security
IP	Internet Protocol
ISMS	Information security management system
IT	Information Technology
KATAKRI	National Security Audit Criteria
MS	Microsoft
NATO	North Atlantic Treaty Organization
NCSC	National Cyber Security Centre

Nmap	Network Mapper
OpenVAS	Open Vulnerability Assessment System
OQL	Object Query Language
OWASP	Open Web Application Security Project
PHP	Hypertext Preprocessor
QA	Quality Assurance
SDLC	Software Development Life Cycle
SQL	Structured Query Language
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UML	Unified Modeling Language
URL	Uniform Resource Locator
USB	Universal Serial Bus
VAHTI	Government Information Security Management Team
VoIP	Voice Over Internet Protocol

1 Introduction

1.1 About cyber security

The definition of cyber security varies depending on the forum where this well known and almost daily used term is used. Cyber security means a wider area of security than data security and if compromised, losing the cyber security might cause damage to the critical functions of the society. Globally, there can be several purposes to build protective mechanisms in cyber space. Usually cyber security is perceived as a network-, application- and information security supplemented by information (InfoSec) security, disaster recovery and business continuity management.When considering cyber threats of a different kind, threats can be divided into cyber terror, cyber attack and cyber crime subtopics (Cisco Systems 2018).

Cyber terrors usually have an effect on the IP based electronic systems to cause chaos, panic or fear. A cyber attack often involves politically motivated information gathering or causes service of denial or other outage or lack of targeted ICT/IT system´s service level. Cyber crime usually includes single nefarious actors or criminal groups targeting systems for economic benefits. Crimes are usually committed with extort programs by stealing data and with a demand for ransoms. (Kaspersky 2018.)

A successful cyber security thinking can be planed with several layers of protection among the computers, networks, software, or data that needs to be kept safe. In an organization, staff, organization main and support processes and technology must support each other to create an effective defense entity for cyber attacks. End users are needed to fully understand and comply with all cyber security principles defined by organization such as following the password and e-mail policy without forgetting backing up data on a daily basis. Organizations need to have a framework and backup plans how to address the problems with cyber attacks in case of an attack success or attack failure. One mature state of art type of guidelines can cover everything in the cyber domain and can explain how people can notice cyber attacks and how to protect big or small systems. Guidelines can give comprehensive answers

howto detect and immediately respond to cyber threats, and how perform to recover from attacks where attackers achieve their goals. (Cisco Systems 2018.)

In today's digitalized world where data handling and networks are considered as a backbone of the society, everyone can get benefit out of advanced cyber defense programs and systems. People on the earth rely on a critical IP infrastructure. Banks, power plants, defence systems, hospitals and basically all industry functions one way or another are exchanging digitally information needed to run the society . Securing these assets' data and infrastructure is essential to keep the society functioning safe. (Cisco Systems 2018.)

> *According to Statista: The statistic presents the recorded number of data breaches and records exposed in the United States between 2005 and the first half of 2018. In the last measured period, the number of data breaches in the United States amounted to 668 with over 22 million records exposed.* (Statista 2018.)

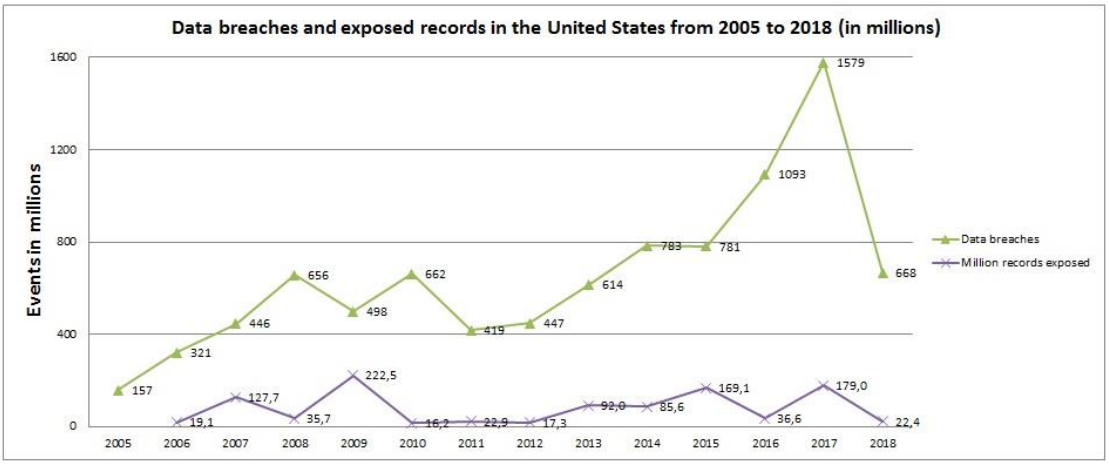


Figure 1. An annual number of the data breach and exposed recordings.

As displayed in Figure 1, it is easy to build an image how many data breach events have occurred just inside the United States of America and how seriously cyber threats need to be taken into consideration in the increasingly digitalized world. After some years, especially when the Internet of Things has spread globally statistics might look very different than they look right now (Center for strategic & international studies 2018).

1.2 Cyber threats to commercial enterprises

There can be two different kinds of kind cyber threats which may be targeted against commercial enterprise actors. For industry actors, losing data usually means losing money but also industrial espionage can be one area of interest for attackers. This basically means stealing research and development data or information about what can be used to defeat the rival in a bidding competition. Attacking against a rival's web service can be beneficial to some instance, for example during a big sport event where the whole world's focus is on some online service in a specific time of the year. (Center for strategic & international studies 2018.)

Another common attack method causing immediate threat to commercial actors is the usage of ransomware. Ransomware is a type of malicious software designed to extort money or other assets by hijacking targeted files or the whole computer system until the ransoms are paid or other demands are fulfilled. It should be kept in mind that paying the ransom or obeying demands does not necessary guarantee that the stolen data or assets will be recovered to the owner or the system will be unlocked. The attacker might put forward additional requirements and repeat this process, and the locked data stays locked forever. This is one good example of why cyber defense investments are money well spent if a company's data is considered business critical. (Cisco Systems 2018.)

1.3 Cyber threats in international military domain

Cyber threats in the military domain have caused discussion for many years. Many military branches and government agencies have built capabilities for defending military networks, operational data- and weapon systems. NATO is also faced with a development of a very complex threat environment with many kinds of electronic threat vectors in the cyber domain. In military doctrines, cyber-attacks have been a part of the hybrid and modern warfare already for some time. NATO relies on strong and dynamic cyber defenses to fulfil NATO´s core missions of collective defense, cri-

sis management and cooperative security. NATO, like many other military branches needs to be well prepared to defend its networks, operations, missions and weapon systems against the growing modern cyber threats and attacks it faces. (North Atlantic Treaty Organization 2018.)

The number of the increasing cyber-attacks is shown as an increasing need for economic resources to build cyber defense and security systems. Figure 2 shows how much money just the Government of The United States has proposed to spend during the years 2017 – 2019. (Statista 2018).

> *According to Statista: In FY 2019, a 15 billion U.S. dollar budget for cyber security spending was proposed, representing a 4.1 percent increase from the previous year. These federal resources for cyber security are set to support a broad-based cyber security strategy for securing the government, enhancing the security of critical infrastructure and important technologies. (Statista 2018.)*

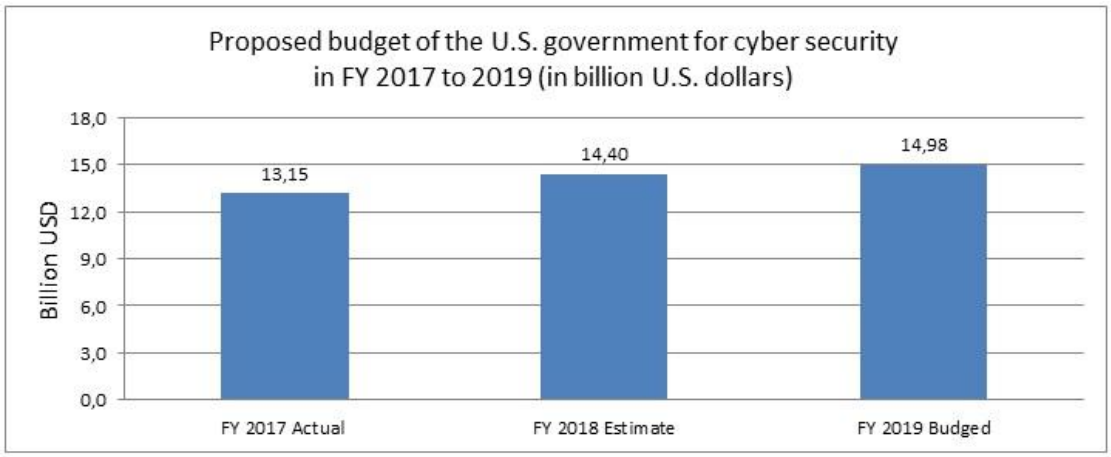


Figure 2. Proposed budget of the USG for cyber security.

One of the scariest threat scenarios is that a cyber-attack can cause a situation where a nuclear weapon system control is lost to a hostile attacker. The biggest issue and maybe also the biggest threat vector in the subject is that nuclear weapon systems were developed way before the time of the internet's arrival and modern computer technology. This issue gives hostile actors an unfortunate change to exploit vulnerabilities against the command and control systems of nuclear weapons. In the worst case scenario, there can be extremely serious consequences if nuclear governments

lose control of command and control of the weapon systems to a party who intends to take hostile actions. (The Guardian 2018.)

1.4 Cyber security in Finland

The Finnish Government has released Finland`s national Cyber security Strategy Government Resolution which defines national level vision for the desired levels of cyber security. Because Finland is a small country, the vision needs to be synchronized with Finland`s resources and the knowledge effectively available. Figure 3 displays the vision of Finland's Cyber Strategy and how the various areas are bound to each other. (Implementation Programme for Finland's Cyber Security Strategy for 2017–2020 2018; Turvallisuuskomitea 2013, 3.)

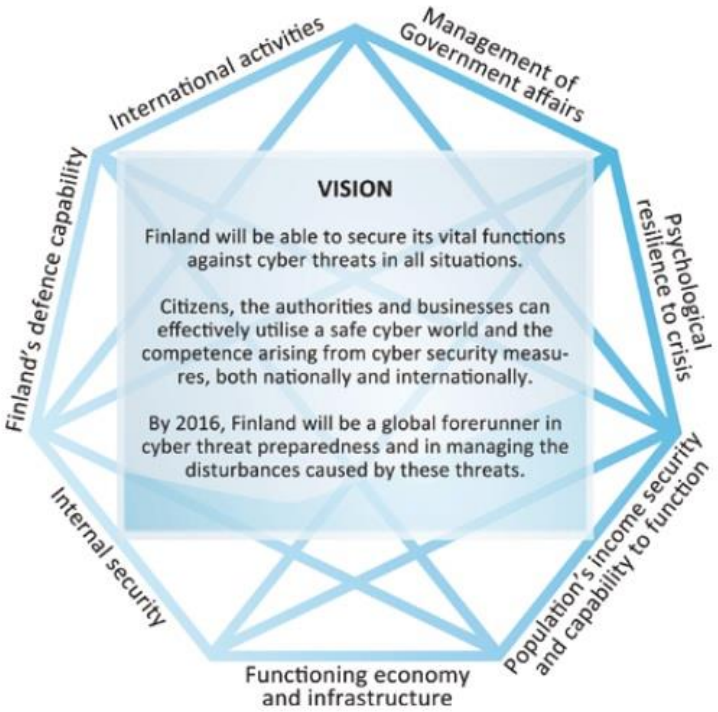


Figure 3. Vision for cyber strategy.

After the government resolution level of the cyber strategy was issued describing cyber security strategy with the big vision and policy settings needed, it was noted that an implementation program was also needed for the deployment of the desired

vision. The implementation programme for the years 2017 – 2020 was launched and it addresses natively across the Finnish Government agencies and industry. (Implementation Programme for Finland's Cyber Security Strategy for 2017–2020 2018, 4 and 7.)

One of the strategic guidelines of cyber security preparedness defined in the Implementation Programme for Finland's Cyber Security for 2017-2020 2018, 7) is:

> *The Finnish Defence Forces will create a comprehensive cyber defence capability for their statutory tasks.*

The Finnish authorities have taken many government level actions to fulfill the Government Resolution to Cyber security Strategy. Some of these actions and the most famous ones are Finnish national audition criteria (KATAKRI) published by the Finnish ministry of Defence, and VAHTI instructions published by The Finnish Ministry of Finance. In addition to VAHTI and KATAKRI, The Finnish Communication Regulatory Authority agency has published guidelines and instructions for the subject, and the agency also acts as a national Cyber Security Centre of Finland (NCSC-FI). (Finnish National Security Authority. National Security Auditing Criteria 2015.)

KATAKRI publication defines the security measurements and criteria at an organizational level but according to the thesis goals, the subdivision "I" defining communications security-, systems-, data and operations security criteria can be looked into more deeply. The first version of KATAKRI was published in 2009 as it was considered one part of Finnish Government's programme for internal security. (Finnish National Security Authority. National Security Auditing Criteria 2018 2 and 29.)

VAHTI instructions, which have been written and maintained mainly for the government agencies are also applicable for commercial enterprises for cyber protection planning. VAHTI instructions defines security requirements and common used tools for risk management, impact assessment, data encryption guidelines and other general level information security instructions and guidelines (Ministry Of Finance. VAHTI-instructions 2016).

As The Finnish Communication Regulatory Authority agency is responsible for acting as a Cyber Security Centre of Finland (NCSC-FI), the agency is also responsible for

CERT-FI duties including the following main tasks and objectives (Finnish Transport and Communications Agency. CERT-FI 2018):

- Solving information security violations and cyber threats against IP network
- Gathering information on cyber incidents and violations
- Broadcasting and sharing information on information security matters and detections
- Ensuring that public communications networks and communications services function safely and properly in Finland
- All safeguard functions that are vital to Finland`s society.

The agency is also responsible for the official regulations such as telecoms operator`s rights and obligations in Finland, electronic identification and trust services and corporate subscribers' rights and obligations in Finland. (Finnish Transport and Communications Agency. CERT-FI 2018.)

Because of the global rising of cyber threat level and definitions captured from the Implementation Programme of Finland's Cyber Security Strategy, The Finnish Defence Forces have also started significantly to raise their cyber security capability and readiness in the subject. The goal is to achieve an operational state where The Finnish Defence Forces are able to allocate resources to cyber readiness and surveillance and protection of operational data networks (Puolustusvoimat kohottaa valmiuttaan kyberuhkia vastaan 2016).

1.5 The Finnish Defence Forces Logistics Command

The main mission of the Finnish Defence Forces Logistisc Command is to ensure that the Defence Forces have effective capabilities in operational use and ensure the deployment of operations in all circumstances during peace and war time. The agency takes care of the performance usage of the Finnish troops and usage of the operational command and control systems used for the home land defence. System

maintenance, lifecycle management, research and development are also important tasks given to the agency. (Puolustusvoimat 2018).

The Logistics Command started to operate in year 2015 and it has 2,200 employees operating in 39 different areas around Finland. Logistics Command's annual budget is approximately 1,400 million euro. Approximately 500 million euro will be used for material acquisitions annually. Under Finnish Defence Forces organization is also operating Finnish Defence Forces C5 Agency which is working in the cyber defence domain. (Puolustusvoimat 2018.)

The main duties of the Finnish Defence Forces are regulated and defined by the Finnish law. The same law defines four major tasks that the Defence Forces are defined to perform. The same law also regulates operating mandate, organization structure, administration and decision making hierarchy and staff. Defence Forces main duties according to the law are (Laki puolustusvoimista 28.6.2017/427 2 §):

- Defending home land
- Supporting other authoritys
- Obeying European Union agreement accordance to the article 222 or according to the article 42 paragraph 7 for international missions or assistance
- Participating in international military crisis manangement.

Top level guidance for planning Defence Forces activities is related to Finland's Security and Defence Policy which is published by the Finnish Prime Minister's Office. This publication sets the foundation for guiding Finland's policy and interests as well as goals according to the subject. The publication analyses changes and trends in the global security environment and discusses defence capability development and builds a framework to secure the critical functions of Finnish society. Finland's security policy cover the most important goal of Finnish security and defence policy. (Prime Minister's Office 2012.)

2 Examples of software development frameworks

2.1 Scrum

The framework of Scrum software development is built on top of three cornerstones: inspection, adaption and transparency. Inspection points and phases are the most beneficial when performed by inspectors at certain points of the work. Scrum users must frequently and constantly inspect Scrum artifacts and progress toward each Sprint's Goal to detect the unwanted variances of the code. Scrum actors' inspection cycle and phases are not to cause disturbance to the work itself. The Scrum framework is displayed in Figure 4. (The Scrum Guide 2017, 3 and 5.)

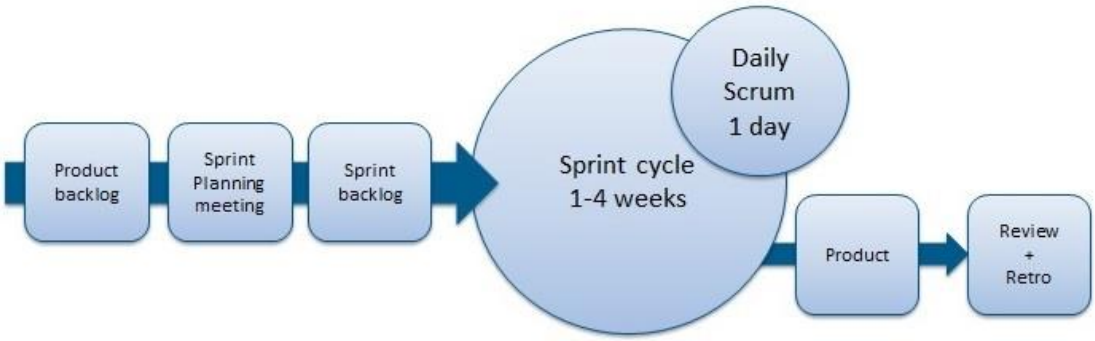


Figure 4. Scrum framework.

If a scrum member detects that one or more aspects diverge from the acceptable limits, and that the end product will likely be unacceptable, the process or the requirements under work need to be changed. Changes must be made as soon as possible to minimize further damage to the development. (The Scrum Guide 2017, 3 and 5.)

Transparency means that all significant aspects of the Scrum must be visible to Scrum members for a successful outcome. Transparency requires all aspects to be defined by a common understandable standard so observers and Scrum members can share common understanding of what is being seen. (The Scrum Guide 2017, 3 and 5.)

Scrum guides describe four formal events, requirement inspections and adaptions. The events are: Sprint planning, Daily Scrum, Sprint Review and Sprint Retrospective (The Scrum Guide 2017, 3 and 5).

2.2 Extreme programming

Extreme programming was first introduced in 1996 and it is one framework in agile software development framework family. One of the biggest benefits of extreme programming is that the framework emphasizes customer needs and the fulfillment of the requirements set on the software under development. The framework is based on simple rules, which are planning, managing, designing, coding and testing. These rules are bound closely to the customer participation via user stories and test scenarios. (Extreme Programming 2013.)

Customers enjoy being participants in the software process with the development team as developers actively contribute to the experience level, and managers concentrate on communication and relationships. With the customer's constant interaction, unproductive activities are scaled down, and costs and frustration of project staff is reduced. As described in Figure 5, Programmers get feedback by testing their software starting straight from the beginning of the project. (Extreme Programming 2013.)

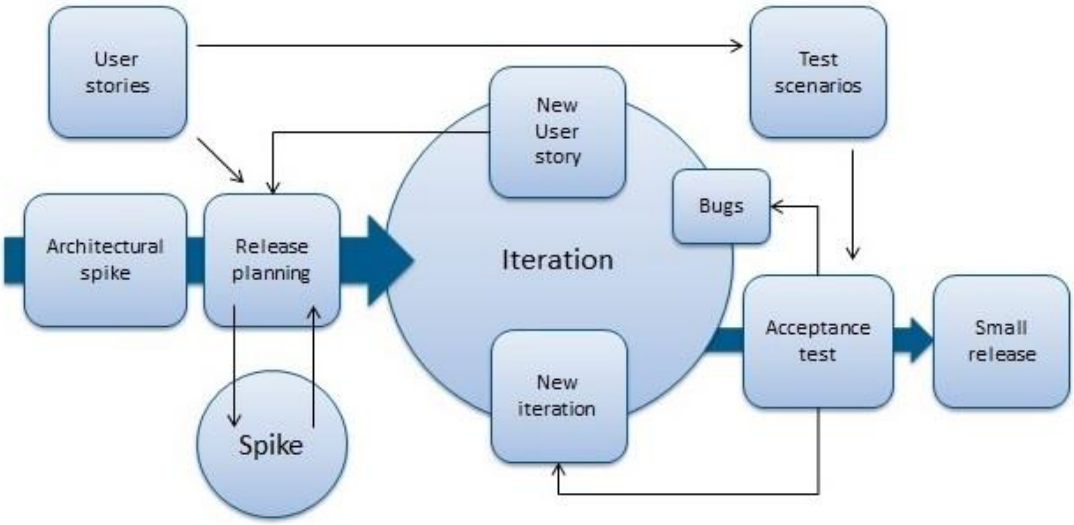


Figure 5. Extreme programming project.

The idea is that developers deliver the software release to the customer as early as possible and implement requirements and user stories as suggested. Every small success deepens the programmers' respect for every team member. With these principles, extreme programmers are able to respond to changing requirements, technology and customer stories and requirements (Extreme Programming 2013).

2.3 Waterfall model

The waterfall development model in software development life cycle is also a long known model for the developing of not only software but also systems and more traditional products. The waterfall model can be used in almost every project with its advantages and disadvantages depending on what goals are set on the project (Tutorialspoint 2018).

Some major advantages using the waterfall model in software development are: it is simple to use and understand, the framework is easy to manage as the definitions of steps are clearly defined, the steps are executed one at the time so there is no need for multitasking, and the model is applicable especially, when the requirements stay

static, as the tasks are easy to organize and milestones are easy to understand. (Tutorialspoint 2018).

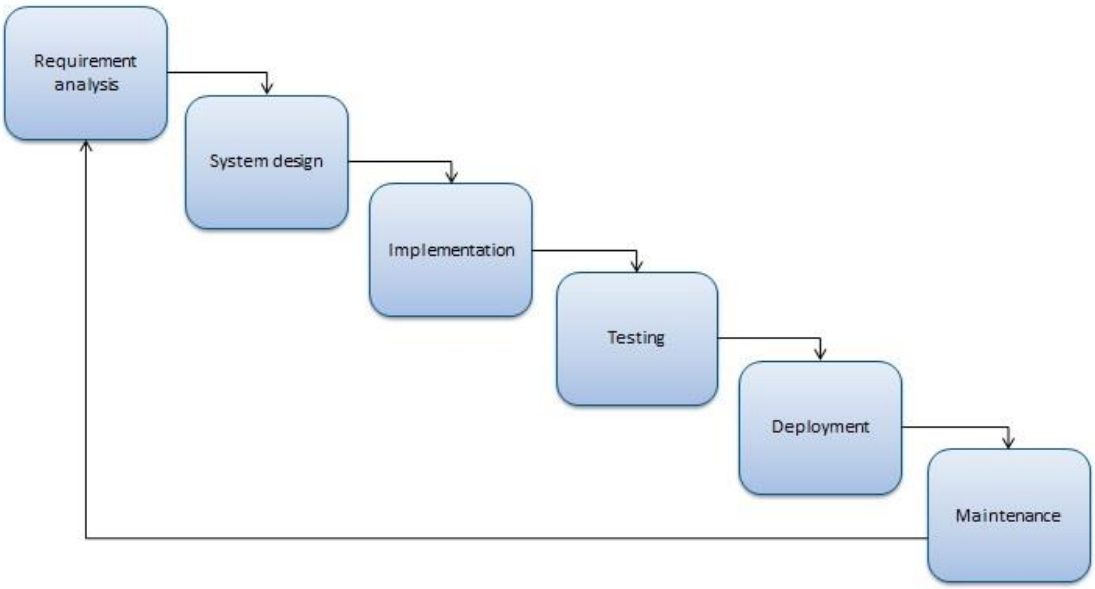


Figure 6. Waterfall model.

Some disadvantages of the waterfall model to software development are: multiple revisions and end product reflection is quite low; once the product is ready and it is tested, it's hard to go back and fix possible problems. Some other features of this model are that no working end product is released before all steps have been completed and the requirement changes are hard to manage. If the requirement definitions are failed in the beginning at the project, there is a high risk for failure for the end product performance. Figure 6 displays a basic waterfall model with commonly used milestones and steps. (Tutorialspoint 2018.)

3 Secure code development example tools

3.1 Wireshark

Wireshark network traffic sniffer is widely used and a globally and commonly known network traffic and protocol analyzer tool. Wireshark has many features sets for deep inspection, live traffic capture and offline analysis, VoIP analysis and a wide range of features for importing data and export analysis and reports. During software development security measurement, the Wireshark tool can be used investigate what TCP- or UDP port ranges a specific software module uses or if the IP network traffic between the software modules is executed in a way that security requirements are met. (Wireshark 2018.)

3.2 Nessus

Nessus was developed to discover vulnerabilities in software. Another feature of Nessus is to discover configuration and compliance assessments. The goal of Nessus is to prevent hostile attacks on IP network by identifying the vulnerabilities from software modules attached to the network. Nessus also discovers configuration issues and mismatches that hackers might use to penetrate IP network or intercept network traffic. Some of the key functions of Nessus are high-speed asset discovery feature, configuration auditing and target profiling feature and malware detection functionality. (Nessus 2018.)

3.3 Nmap

Network Mapper, or Nmap is a tool for network exploration and security auditing of IP networks. Nmap is based on open source code and it is free to use. The advantage of Nmap is to rapidly scan large networks, single hosts or software components.

Nmap can discover what hosts (IP addresses) are available on the scanned IP network and what services and operating systems the found hosts are running. Nmap is not only a good tool for security measurements but it is also handy for routine tasks such as network host inventory and asset management, managing assets and services provided per asset. (Nmap 2018.)

3.4 Threat modelling

Threat analysis means a comprehensive approach and deep analysis of threats that might occur against a software (assets) under development or deployment. The goal of threat analysis is to give input to the requirement set on the code to secure confidentiality, integrity and availability of the software. During the threat analysis process it should be listed who or what might be potential attacker(s), what kind of threats a specific code is exposed to, or third party modules and libraries, what are other assets, who are stakeholders and what are the odds and risk level that some threat event occurs. (Aura 2017, 4.)

Threat modelling and analysis can be done more than one way, and the right method is often related to the domain where software is developed and the environment where software is used after it is deployed into the production. During threat modeling it is also advised to discover counter measurements against each threat if a threat is escalated for one reason or another. One way to execute this task is to use mind mapper, MS PowerPoint© or UML modeling tools for modelling data flows and dependences between the software modules and stakeholders. (Aura 2017, 4 and 9 and 14.)

One way to introduce threat modeling philosophy is via an example from a more commonly known environment, e.g. the theft of a car as displayed in Figure 7.

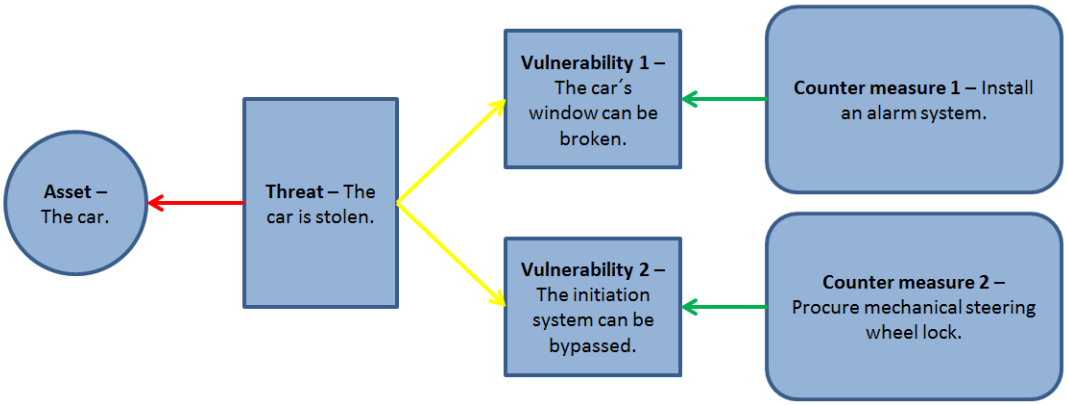


Figure 7. Threat modelling example.

3.5 Coding standards

Some corporations have developed their own coding standards for secure software development as security of the software does not happen by itself. Secure software development requires a security assurance during the software development lifecycle across the organization and the development team. The main goal is to achieve the level of the conformity with the objectives, policies, and principles with a process transparency applied which meets the desired level of the code security. Building these standards applied to the code development has been driven within the innovations, well known coding standards, training and the use of the automated testing tools and analysis like code reviewing. (Oracle 2018; Microsoft 2014.)

The secure coding standards have been studied by various research institutes and researchers around the world. The institutes and researchers have published articles where the focus has been on the importance of the secure software development. One way to approach secure software development is to build a knowledge base on what contributes to the set of security measures and different technologies for the coders who are non-experts in secure software development. One knowledge base covers the artifacts of the common tasks around the subject such as identifying threats, the lists of attacks and countermeasures, and an analysis on how a countermeasure is used in a common way. (IEEE 39th ACSAC 2015.)

Rules for the developing secure code have also been introduced by the computer emergency response team where the secure coding standards have been coordinated and maintained with the researchers and specialists. More than 1,900 instances are developing rules and guidelines for the set of the coding languages. The latest publications introduce the rules and the root factors for secure, reliable and safe code development. The guidelines also introduce how vulnerabilities can be exploited and what can be the follow ups for the security breach. Some of these standards and secure coding books are free for download and currently cover coding languages C, C++ and Java. (Software Engineering Institute 2016.)

Even though there is no common widely used framework applicable to every software project used for the secure software development standard, some other ICT standards provide baselines for taking security measures into account when embedding the secure software framework and processes. The security requirements shall be written against the real threats keeping in mind that the usability of the software shall also be achieved. In some cases, the security requirements may forbid the third party software component communication with the core functions of the software. This means that certain assistive technologies within the software development shall be allowed to communicate in accordance with the level of the security set on the core code (SFS-EN 301549:2018:en 2012).

4 Theory summary

4.1 Cornerstones of Theory

According to the research questions, the main cornerstones recognized from the theory part are mostly used in agile software frameworks and more widely traditionally used in the waterfall model. In addition, some secure software development models introduced in Chapter 3.5 are recognized as cornerstones when summarizing the theory. The theory part also introduces some examples of the security testing capabilities, which also partially answer to the proposed sub-research questions. The cornerstones also contain some secure software development coding models invented by corporations. These models include some mechanisms which provide a secure code checking, code reviews and models for secure software development. Risk management and threat modelling are also seen as cornerstones of the theory and these points were also raised up during the literature research.

4.2 Linking theory to the research

In the first instance, the literature research results are analyzed against the research questions but also the cornerstones raised up in Chapter 4.1 are linked to the analysis of the research results. As risk management and threat modelling were spotted as one of the cornerstones of the theory part, these factors are also linked to the survey research questions.

Multiple-choice questions in the research survey part are linked to the observations done within the literature research and the theory introduced according to the security testing capability. The theory chapter includes points which were clearly seen as the main point around the research questions. Some security testing capabilities and coding standards were introduced because the hypothesis is that similar kinds of tools or tool sets are used for checking the software security.

5 Research methods

5.1 Main research questions

The main research question is “how should software code be developed in a secure way?” During the research work, it was defined in order to get answers to the following sub questions derived from the main question of the research:

1) What are the best and most beneficial points in software development project where software code shall be security checked?
2) How big software entities would usually be wise to expose to security check?
3) What are appropriate entities or phases in the framework where security checks should be focused on to ensure that the final product of the development process fulfills the requirements set on the product (e.g. modules, interfaces or HMI) according to the framework used?
4) What would the best capabilities be that can be used for software security checking during software development process?

The main questions and the scope of the thesis are defined above; yet, a secondary goal and a so-called “side product” of the thesis is to get a vision by using the results of the thesis where further research would be beneficial to focus on as the threat environment is developing every day.

5.2 Research principles applied

5.2.1 Dividing the research

The research is divided into two parts. The first part is carried out as a literature research based on different sources around the research design and the second part is conducted using a questionnaire survey addressed to the domestic and foreign industry with a possibility for a free feedback section. Selecting and scoping the re-

search and analyzing methods implemented into the research it is important that the methods selected are in line with the main research questions and serve the intended purpose and the goal of the research as well as possible.

5.2.2 Research methods and ethical questions

When defining research methods, it is important that all necessary science factors are met in practice including the known practices of science research which guarantees the credibility and reliability of the research. During the material collection process, it is defined how valuable the collected material and information is in further research and if the materiel can be used for another research of the same subject or can the material be benchmarked against another research focusing on the same subject? When collecting and analyzing research data, it is considered that the results and material are valuable enough to store the data for further analysis. According to the main research question, these factors are critical to take into consideration during the definition of research methods and practices. (Kuula 2015.)

Literature research mainly relies on the information gathered on the well-known and trusted official web pages found on the internet and electronically published publications, standards and guidelines found in the university libraries and official standardization domains. In the nature of this, it can be concluded that ethics of the usage of the internet for research fulfills the legal and research ethics. During information gathering it is evaluated in advance that the Finnish law protecting the personnel data and copyright law of Finland sets no limitations to the usage of the sources found in online services and in internet domains. It is also evaluated in advance that there is no limitations from the European Union regulations when collecting research data. (Kuula 2015.)

According to the Finnish laws and based on the nature of ethical research questions using internet sources in research, it can be concluded that laws and ethical questions are fulfilled because of the following points. (Kuula 2015.):

- There is no need to register into a specific online service for information gathering. Standards and e-books deviate from this definition because the documents are available from the official service provided by the University. According to this definition, it can be concluded that there is no limitation for information used within the research

- Storage time and information availability might vary between sources used but the main scope is that sources used are well-known official domains. According to this, it can be considered that the information used is trusted and meets the quality requirements set on the research. This same statement applies to the users and maintainers of the online services and internet domains.

The basic principle for form data handling and securing anonymity of a quantitative questionnaire and free feedback is that no information is collected that needs to be separated from the big data entity. This means that there is no need for removing a specific data or build decode system for data given within the answers. Quantitative data from a questionnaire research is collected and stored in the way that the data stays anonymous and the definitions from Finnish laws protecting personnel data are met. Company names and respondents' personal or other identification data is not collected. If for some reason the gathered data is considered sensitive according to the Finnish laws (if given in the open feedback section), that particular data is removed to respect the respondents' answers and the company privacy. (Kuula 2015.)

In addition, after meeting the research methods and having defined the ethical questions, during the research execution well-known good practices for science are also applied. This means honesty and common accuracy for data handling; the recordings' results and the results analysis and displays followed a common sense according to the definitions of the ethical board of the research. Gathering information according to the commonly used ethical principles and analyzing and evaluating data gathered during research is also conducted in accordance with the criteria for research based on science. Data used from the research published by other researchers will be re-

spected in the appropriate way that research results can have the scientific value that belongs to them. (Kuula 2015.)

5.2.3 Literature research

Literature research gathers data from different sources and data is analyzed using qualitative research methods. During process when the results are analyzed, discretionary sampling of the results is applied. When scoping the sources for the research, it is defined that only a limited amount of trusted well- known sources is used which also fulfills the goal of the research. Theses and research publications are considered well trusted sources as the sources have been checked and accepted by science institutions and universities. Publications published by industry working around the cyber domain are also considered well-known trusted sources as it is assumed that well-known companies and their representatives have deep knowledge and long experience from the cyber domain area. In addition, sources have also been selected within the consideration what the goal of the research is and what the main research questions are. (Rautio 2007; KvantiMOTV 2007.)

The qualitative analysis is characterized by inductive reasoning, which aims to make generalizations and conclusions on the basis of facts displayed in the material used. The aim is to analyze the material in a complex and detailed way, displaying on relevant themes according to the research goal. Research does not aim at statistical broadening. The methods for analyzing literature research include discussion analysis where results are benchmarked between the sources and conclusions done based on the sources used. (Rautio 2007; KvantiMOTV 2007.)

5.2.4 Survey research and open feedback

The goal of this applied research method is to understand the phenomenon of the subject researched. During the research, no predictions or hypotheses are set on the expected results. Usually, this research method is used when a large group of people is exposed to the research, and quantitative research cannot give answers to the question “why”. According to these specifications, the quantitative research method is the most applicable to getting answers to the main question of the research. The theory behind the questionnaire research is presented and the results are benchmarked towards the theory and the results of the literature research, however, the goal of the questionnaire research is not to develop the theory itself. (KvantiMOTV 2008; KvantiMOTV 2010.)

Questionnaire results are organized in order starting from lower and going to the highest amount of answers (KvantiMOTV 2008; KvantiMOTV 2010).

The questionnaire survey is completed with Webropol survey tools, which will provide analysis tools for the gathered data. The tools are used to analyze the results of the questionnaire, and the following results and factors are calculated and displayed as follows:

1) Dependencies to analyze is how answers correlate with each other and how statistically significant the results are
2) Quantities of the answers for an analysis how statistically significant a quantity or answer is. Generally, it can be stated that an answer is statistically significant when more than ten answers have been received
3) Average of the answers and confidence interval of the quantity of the answers. A large confidence interval indicates that the answers are not statistically significant
4) The median value of the answers. This indicates the importance of a specific answer
5) Standard deviation. A huge deviation value indicates that the defendants have disagreed on the opinions between the answers.

The results gathered from the open feedback section are analyzed using the same analysis methods as the ones conducted in the literature research.

Because there is no hypothesis set on the questionnaire research, there is no need for operationalization as the measured subjects are more objective than subjective. This defines that the meters constructed for the research are suitable and in valid comparison with the audience of the respondent cohort. Therefore, the meters set on the research are considered reliable enough according to the main research questions. When dividing research meters into stability and consistency factors, it can be concluded that factors are in line with the goal of the research as the timeline for the survey is limited. A limited time line downscales the risk that respondents might change their minds about the answers provided, and survey technical execution also supports this philosophy. (KvantiMOTV 2007; KvantiMOTV 2008.)

When defining the questionnaire for which there is an electronic form, the content of the form is kept quite short and the proposed questions have been carefully considered against the research questions. The proposed questions are logically progressed from the top level topic down to more substance level questions. Common requirements for the clarity of the form are applied, and the questionnaire is simple and easy to understand. The options for answering are set to control respondents to choose “either or” type of answers according to the main questions and the goal of the research. Building trust between the researcher and the respondents is important, and the motivating of the respondents is covered with a cover letter of the research. (KvantiMOTV 2010; Valli & Aarnos 2018.)

6 Literature research

6.1 Framework, processes and testing

When defining the security measurement phases in the software development frameworks, it should be kept in mind that the goals for the actual tools used for hands on work are to ensure, protect and identify the risks before they are realized. In order to achieve this goal this goal, it should be kept in mind that the software security is a process that will identify planning errors, execution errors and operational errors. Automatic tools are used for finding the input errors; however, errors occurred in different use cases of the software need to be found manually. Operative errors can be easily found with the vulnerability scanners and the testing can be automatized without any manual interaction. Figure 8 displays the process and the phases where the security measurements will be taken into use during the software development process in the agile framework. (Kirmanen 2016.)

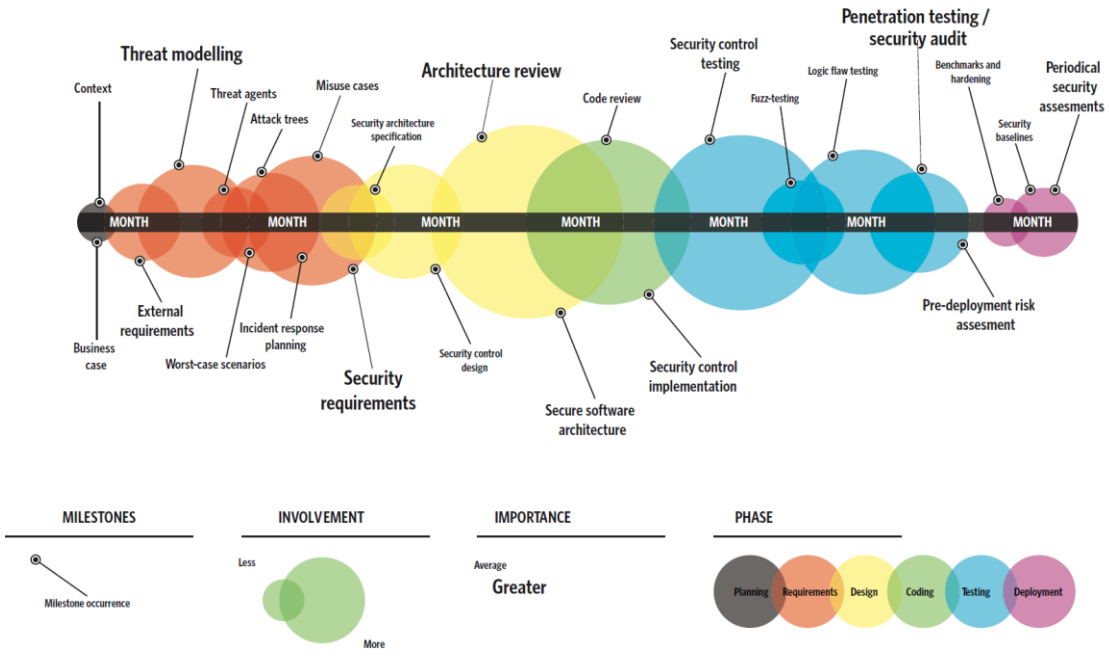


Figure 8. Security testing process.

As displayed in Figure 8, security testing is performed from the administrative point of view and from the technical point of view before the deployment phase. The technical security measurements are fuzzing-, logic flaw- and penetration testing. It shall also be noticed that even though the research focuses on phases and tool features used during the software development process, threat modeling, worst-case scenarios, incident response planning, security requirements and architecture specification with the security control design are implemented before the usage of the security tools. This means that the administrative security planning needs to be well focused before the actual hands on security testing. The process displayed in Figure 8 does not depend on a particular software development framework, so it is applicable equally to the agile frameworks such as Scrum and the waterfall models. (Kirmanen 2016.)

During the security testing planning, it shall be kept in mind that there can be certain factors in the testing from the security point of view. This means that the hands on test phases must be changed or shifted to focus on software security architecture, overall threat analysis and the use cases. The software should be checked during the development with functional testing as far as it can be done in a realistic and cost effective way within the right phases of the development process. Treating the software entity on the deployment phase as a “black-box” and testing just the external interfaces of the components does not ensure the security of the extracted build of the software entity. (Råman 2006, 30.)

When scoping the software modules and the entities for the security testing, one method in order to achieve cost effective and reasonable entities is to start with the baseline build in accordance with the common understanding for the project’s goal and get the results and error the lists as well as redefine testing entities and repeat the test process. After receiving the error list or the security flaw findings, the test personnel can build an error list, prioritize the errors from critical to primary and secondary. After this, through the risk evaluation, mitigation strategy can be done against the error list. Another approach to scoping the software module entities is to start with the list of top 25 the most dangerous software errors based on the 2011 CWE/SANS publication and benchmark the list against the software security architecture, threat analysis and the security requirements (Mini & Feng 2016, 63).

Secure software development work should be started from the enterprise-wide strategy defined program, which gives the top-level frame for the security culture according to the subject. One shall also keep in mind that implementing security during the software development does not cover the whole security after the software deployment. This gives a good reason for implementing the security also in maintenance and continuity management during the life cycle of the software. The factors mentioned are one starting point when scoping the software entities which need to be security checked during the development. After the requirement and design phase when the threat modeling and risk assessments have been done and the development implementation phase is starting, the proper security testing tools need to be implemented. In the verification phase fuzz testing and the final security review in the release phase should be performed. In the agile software development lifecycle, security can be implemented and categorized into three different phases with the subtopics, which can be presented against the research main question and in accordance with Table 1. (De Win 2016.)

Table 1. SDLC task breakdown

Phase	**Function**
Every sprint	Database access. Security fix. Mitigation. Threat model update. Credentials encryption. Internal security review.
Bucket	Security verification (Fuzzing and Penetration testing). Design review (Privacy review and threat model). Planning (Update buglist).
Onetime requirements	Baseline for threat model. Security response plan.

	Define security Officer. Define and use latest compilers.

As the SDLC combines cornerstones such as the risk management, tools, knowledge, processes and people with the training implemented into the phases, it can be used effectively when scoping the software module entities that need to be security checked during the development process (De Win 2016).

Another way to approach the research question is to applying the compatibility analysis of the software security matrix that defines the security factors implemented for the software development phases. The matrix is a very handy tool in the planning phase and when defining what security measurement and capabilities are needed during the process. When scoping the software module entities for the security testing, it shall be kept in mind that approximately 38 % of the critical vulnerabilities are caused by failed design of the software. In the implementation phase, approximately 47 % of the critical the errors are caused. According to this analysis, the security testing is vital during these phases. The most common absence of the security and security breach founding areas are analyzed to the five different categories presented in Table 2. (Koistinen 2013, 87 – 91.)

Table 2. Security breach discoveries

Area	**Percent (design phase)**	**Percent (implementation phase)**
Input validation	0 %	100 %
Session handling	32.6 %	14 %
Authentication	37.8 %	51.4 %
Access control	88.9 %	0 %
System protection	24 %	0 %

To find the security breach in practice, vulnerability and penetration testing and fuzzing with the static code analysis are performed according to the good practice methods. The methods can vary as the commercial market provides a wide range of testing tools. Table 3 displays the security testing measurements needed to be executed during the design and the planning phase. (Koistinen 2013, 87 – 91; ScienseSoft 2018.)

Table 3. Security testing areas during development

Area	**Function**
Vulnerability testing	Used for previously software Weaknesses (software and system). Usually function is performed with the automated scanners.
Penetration testing	Method requires more expertise from personnel executing tests. Goal is to find weaknesses and find possible vulnerabilities.
Fuzzing	Application robustness is tested by sending high amounts of traffic with no meaning to the application. This method can reveal unknown errors from the software.
Static code analysis	Method is used to reveal programming errors that are hard to find during the usage of the software. Code analysis is done to the code which is not compiled (source code).

During the secure software development process planning such as the SDLC, it is important to focus on those assets that support the company's business requirements and those that can be easily automated. The business needs can be linked to the

company's ISMS what can reflect on the SDLC processes. When defining the scope for the security testing during the software development, one approach via risk management is to use the Octave Allegro process displayed in Figure 9. (Keiski 2013, 60 – 61.)

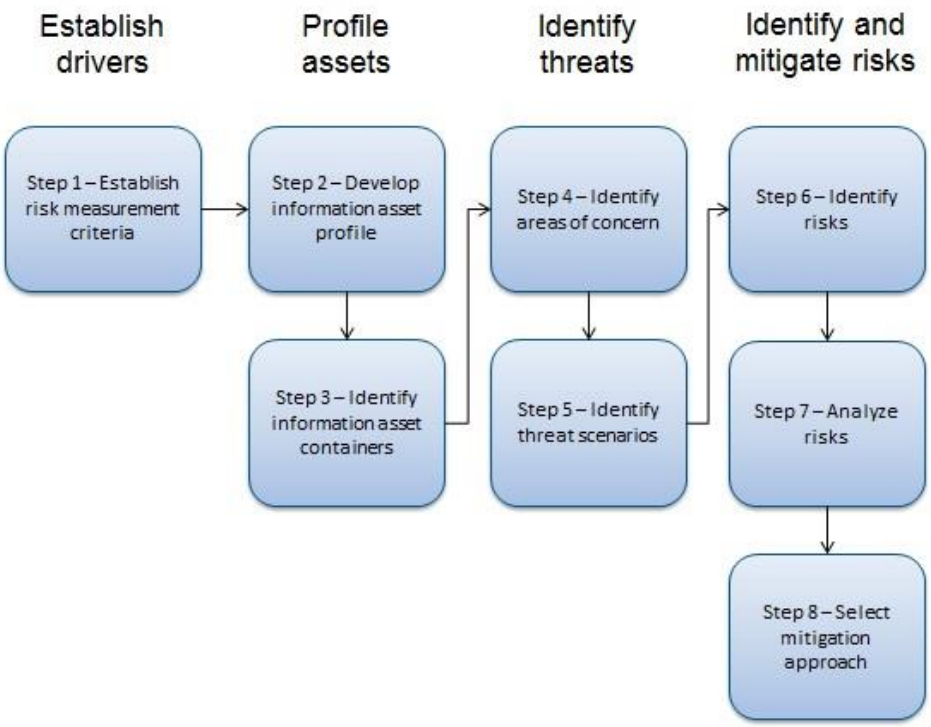


Figure 9. Octave Allegro process.

When defining security testing tools needed for the each phase in the development process, the tools need to be defined accordance to the common used tools baseline. Development team shall define the tools and propose the approval for the usage of the toolset from the third party instance. Using the tools with no latest versions shall be avoided. Some of the commonly used interfaces and functions are not considered as safe in the current threat environment. All interfaces and functions shall be analyzed, and if they or some of them are flagged as a dangerous function, this function should be excluded from the implementation (Karppinen 2014, 28 - 31).

Dynamic analysis during the software development is strongly recommended simultaneously with the other security through the development process. Fuzz testing is

one option to perform a dynamic analysis as this test method can find even the architectural level design errors during the development. Fuzz testing focuses directly on the source code and running tests shall be planned for the regular basis execution. Fuzzer source code analyzers can be divided into four different classes: file format parser, network protocol parser, APIs and miscellaneous parsers. (Karppinen 2014, 30.)

After the secure code development has been performed successfully, the final step is to execute the final security review. During this evaluation it is reviewed if the development team followed the SDLC process correctly and if the final product is ready for publication (Karppinen 2014, 32).

If WebSocket technology is taking a part of the software entity or the project under development, it shall be kept in mind that WebSocket needs to be security tested and security measurements shall be implemented during the development process just like every other software module needs to. It is the programmer's responsibility to do this and take care of the security testing according to the SDLC process implemented. Basic security measurements needed in accordance with the SDLC are penetration testing and a possible code review. Other security areas to focus on WebSocket are authentication, authorization, cross-domain requesting, traffic encryption, input validation from the client and server side and the resource exhaustion. Popular tools for performing WebSocket security test are proxy tools Burp Suite and OWASP Zed Attack Proxy. (Kuosmanen 2016, 26 – 35 and 38 – 53.)

The most common used attacks against the web applications in the cyber domain are SQL injection, session hijacking and cross site scripting. During the web application coding process it shall be kept in mind that where the information is coming and where it should be send. Usually the lack of security in web programming is caused by the inexperienced programmer who ignores especially vulnerabilities of the PHP language (Yurdakul 2013, 13 – 15 and 19).

After the software has been developed according to e.g. the SDLC process, it is important to set up the security metrics which will show objectively how the security measures were implemented. The most commonly used and best known approach to define the metrics is to use the GQM (Goal, Question and Metric) thinking. The met-

rics shall be implemented from the initial phase of the development all the way to the end of the process. The baseline and the number of the different phases are presented in Table 4. (Sibbigui 2017, 11 – 12.)

Table 4. Security metrics definition phases

Phase	**Questions for metric**
Pre-Requirement Phase	Is security required in the system? Is security possible for the system? Number of possible security requirements in each phase of Software Development Process? Total Number of security requirement in Software Development Process?
Requirement Gathering and Analysis Phase	Number of priority security requirements? Number of least priority security requirements? Total number of security requirements? Ratio of security requirements?
Software Design Phase	Number of design decisions related to security? Ratio of design decisions?
Coding Phase	Stall ratio?
Implementation Phase	Number of errors found in the system? Number of implementation errors associated to security of the system? Ratio of implementation errors that have impact on security? Number of exceptions implemented to handle failure related to security of the system?

	Number of omitted exceptions for handling execution failures related to security? Ratio of the number of omitted exceptions?
Testing Phase	Total number of security test cases of the system? Number of security test cases of the system that fails? Ratio of security test cases? Ratio of security test cases that fails?
Maintenance Phase	Ratio of software changes due to security consideration? Ratio of patches issued to address security vulnerabilities of the system?

According to the commonly recognized best practices, the code development implementation phase for securing the software development is to code in accordance with the common standards, code reviews, security unit testing and managing the defects. When executing unit testing, it is recommended that the testing focuses on those interfaces or modules that are going to handle the data feeds received from another user or system. Security testing shall cover the data formats, encrypted traffic and the mechanisms used for the user authentication. During the security testing, the focus should be on the data feeds received, managing errors within the process and regular usage of the static security measurement tools. (Touko 2015, 13.)

The code review also known as peer review is considered a successful strategy for the software quality or the security assurance. The review can be done either manually or with automated tools where developers or review team check the code by reviewing and reading parts of the source code. This function is accomplished after the implementation or as an interruption of the implementation phase of the development. If done by the developer team, at least one of the team members shall not be the code's author. Codes can be reviewed using the online services provided such

as Git, where developers are able to collaboratively review the code. Offline tools for the code analysis can review a large amount of code where developers are systematically performing the checks for the source code for the known vulnerabilities and the programming errors which can cause a security threat (Software testing help 2018).

Some commonly known best practices for code reviews done by humans are (Smartbear 2018) as follows:

1) Maximum amount of code for one review is less than 400 lines of code
2) Speed of the code analysis is 500 lines of code / 1 hour
3) Maximum time for the one review session is less than 1 hour at the time
4) Goals for the review session shall be defined, and the metrics feedback shall be captured
5) Checklist shall be defined prior to the review
6) Processes for fixing the discovered errors shall be defined
7) Positive code review culture should be generated which promotes the advantages of the review session
8) One should not practice too hard and heavy review methods.

When implementing the security measurement into a software development framework, the commonly known challenge is the absence of the standard method in order to perform the implementation from the initial phase of the development all the way to the end of the process in accordance with a specific framework used. One way to approach this challenge is the SCRIPT process which consists of three different phases:

1) Elicitation
2) Modelling
3) Implementation.

Each of these phases contains several iterations according to the subject. In the elicitation phase, the goal is to be achieved by a security specialist who uses meta models describing the basic assumptions and the security models. This leads to the security models, and the profiles for the deployment. Modeling and implementation

phases produce the UML models for use cases and finally version of the code, which ensures in addition to the so called payload code that the final code is secure (Homanen 2017, 13 - 15).

Machine learning can be used as a metric to identify the unsecure code based on the code signature recognition system. The RIPPER, Multi-Naive Eyes and Naïve Bayes methods have been tested for recognition of unsafe samples in the code during the development process. The RIPPER method is based on catching the code in accordance with the DLL function called Naïve Eyes method to recognize characters from the source code. The results for the tested methods achieved the highest value with 97.11 % accuracy. (Tuovinen 2018, 20 – 22.)

Microsoft have implement and defined the secure software development process, which consists of three different elements: best practices, process improvements, and metrics. The process is also known Secure Development Lifecycle. Microsoft have defined comprehensively secure by design, secure by default, secure in communication and communications to help developers to determine where security is needed. This thinking is also known as the SD3+C guiding principles. Table 5 displays the main subtopics for the secure by design, secure by default, secure in deployment, communications, privacy by design, privacy by default, privacy in deployment and communications. (Microsoft 2012, 7 – 11.)

Table 5. SD3+C topics

Area	**Topic**
Secure by Design	Secure architecture, design, and structure.
	Threat modeling and mitigation.
	Elimination of vulnerabilities.
	Improvements in security.
Secure by Default	Least privilege.
	Defense in depth.
	Conservative default settings.

	Avoidance of risky default changes.
	Less commonly used services off by default.
Secure in Deployment	Deployment guides.
	Analysis and management tools.
	Patch deployment tools.
Communications	Security response.
	Community engagement.
Privacy by Design	Provide notice and consent.
	Enable user policy and control.
	Minimize data collection and sensitivity.
	Protect the storage and transfer of data.
Privacy by Default	Ship with conservative default settings.
Privacy in Deployment	Publish deployment guides.
Communications	Publish author-appropriate privacy disclosures.
	Promote transparency.
	Establish a privacy response team.

After implementing topics displayed in Table 5, the SDLC process can be described in Figure 10.

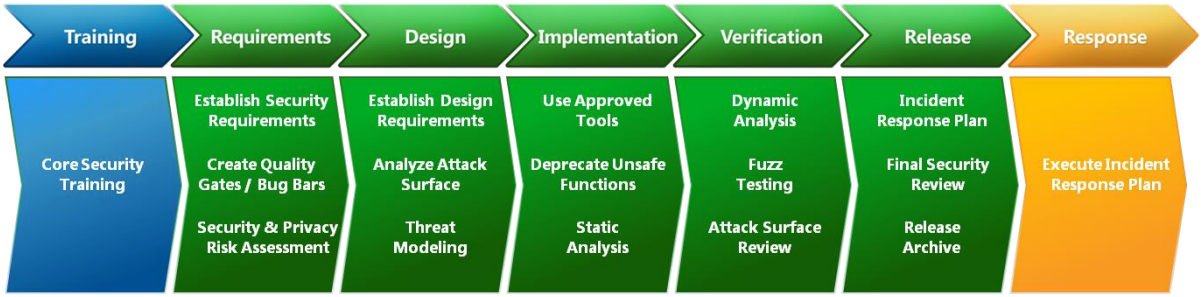


Figure 10. SDLC process.

The goal of the OWASP project is to define and build standard SDLC methodology for software developers coding the Web and APP software. The main goal for the OWASP project is to achieve the highest level of security for SDLC. OWASP is free and open for the users and it includes a risk based approach. The OWASP ASVS provides technical controls and a list of the security requirements that software developers can implement. The OWASP ASVS provides mechanisms for the technical security controls and the technical security controls for the environment, where the Web environment threats are taking place. (OWASP 2015, 9.)

The OWASP ASVS have three different levels of implementation verification. Level one is for all software, level two is for protecting the sensitive data and level three is for the critical applications data with the highest classification of data. Implementing the OWASP ASVS in practice means that the coders need to generate a secure coding checklist that reflects the product requirements set on the project. The ASVS can also be tailored for the customer's user stories or use cases which will increase the level of security for the final product. The OWASP ASVS detailed security verification requirements are displayed in Table 6. (OWASP 2015, 23.)

Table 6. Detailed Verification Requirements

ID	**Requirement**
V1.	Architecture, design and threat modelling.
V2.	Authentication.
V3.	Session management.
V4.	Access control.
V5.	Malicious input handling.
V7.	Cryptography at rest.
V8.	Error handling and logging.
V9.	Data protection.
V10.	Communications.
V11.	HTTP security configuration.
V13.	Malicious controls.

V15.	Business logic.
V16.	File and resources.
V17.	Mobile
V18.	Web services (NEW for 3.0)
V19.	Configuration (NEW for 3.0)

The top-level requirements displayed in Table 6 are divided into sub-requirements in three levels defined by the OWASP ASVS. Developers can easily use the matrix for the security checks and the level of the requirement maturity. Table 7 describes a sample of the OWASP ASVS matrix for maturity traceability (OWASP 2015, 24).

Table 7. Requirements traceability matrix

ID #	**Description / requirement**	**Level 1**	**Level 2**	**Level 3**	**Since (OWASP release)**
1.1	Verify that all application components are identified and are known to be needed.	x	x	x	1.0
1.2	Verify that all components, such as libraries, modules, and external systems, that are not part of the application but that the application relies on to operate are identified.			x	1.0

When coding the client server applications in the Web domain, secure coding practices are strongly suggested to be implemented into the development framework and to the testing phases. The security shall be verified early and often during the development process. Table 8 displays the security tips as an example according to the best secure coding practices that can be implemented. (Veracode 2018.)

Table 8. Security tips for Web coding

Topic	**Tips**
Queries parameters for protect SQL injection	Be cautious about allowing user input into object queries (OQL/HQL) or other advanced queries supported by the framework. Defend against SQL injection using proper database management system configuration.
Data encoding and characterization translation for making injection methods threats ineffective using rendering	Monitor how dynamic webpage development occurs, and consider how JavaScript and HTML populate user input, along with the risks of untrusted sources.
Input validation for ensuring that all data is valid and can be trusted	Assume that all incoming data is untrusted. Develop whitelists for checking syntax. For example, regular expressions are a great way to implement whitelist validation, as they offer a way to check whether data matches a specific pattern.
Identity and authentication controls implementation for avoiding security breaches and building strong authentication methods	Establish timeout and inactivity periods for every session. Use monitoring and analytics to spot suspicious IP addresses and machine IDs.
Access control implementation	Consider denying all access for features that haven't been configured for access control. Consider checking if the user has access to a feature in code, as opposed to checking the user's role.
Data protection according to domestic	Do not make confidential or sensitive

and international regulation limitations applied	data accessible in memory, or allow it to be written into temporary storage locations or log files that an attacker can view. Use transport layer security (TLS) to encrypt data in transit.
Logging and IDS implementation	Keep various audit and transaction logs separate for both security and auditing purposes. Log at an optimal level. Too much or too little logging heightens risk.
Frameworks and libraries leverage. Respecting existing secure approved mothed and libraries	Use web application security frameworks, including Spring Security, Apache Shiro, Django Security, and Flask security. Regularly check for security flaws, and keep frameworks and libraries up to date.
Exception and error handling and monitoring to avoid catastrophic failures	Conduct careful code reviews and use negative testing, including exploratory testing and pen testing, fuzzing, and fault injection, to identify problems in error handling. Confirm that error messages sent to users are not susceptible to critical data leaks, and that exceptions are logged in a way that delivers enough information for QA, forensics, or incident response teams to understand the problem.

The Finnish Government Treasury has published instructions for the secure development guide for software application development. This guideline defines that security shall be implemented in every phase of the development life cycle displayed in Figure 11. (VAHTI 1/2013).

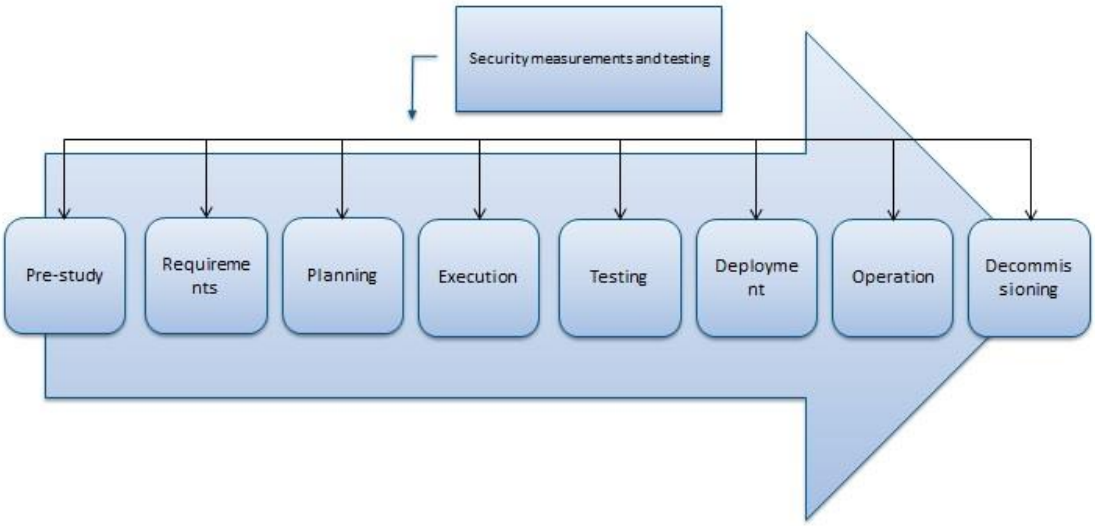


Figure 11. Security measures and testing.

The guideline also defines that if agile frameworks are implemented, the product owner defines implementation and misuse cases of the security and which metrics shall be implemented into three agile frameworks increments. When defining the test cases for the security testing, it shall be kept in mind that at least the test cases shall be led from several different sources instead of just one. Topics that are needed at least for the implemented test case definition work are displayed in Table 9. (VAHTI 1/2013, 18 – 19 and 57.)

Table 9. Test cases for security

Source	**Test case**
Requirements for the code	Case #1
Functional requirements for the code	Case #2
Discoveries and vulnerabilities found earlier (if available)	Case #3
Misuse cases (if been done)	Case #4
Common known problems for technology used (e.g. OWASP Top 10.)	Case #5

To achieve more security awareness and transparency during the Scrum usage, the primary objective is to find a working model for the security testing. The main questions deal with questions such as which requirements shall be chosen for the security testing and how to formulate those requirements into user stories as well as the use and misuse cases. The key factors for success are proceeding bit by bit and there should be a possibility to adjust the security testing on the fly. Learning from the previous projects and colleagues, and taking advantage of existing materials are also key factors for a successful secure code development (Puhakainen, & Sääskilahti 2012).

Adding the security measurements into the Scrum framework requires four steps implemented into the agile framework:

1) Introduce coding processes and practices to the development team
2) Introduction of development phases and operations in practice
3) Introduce threat modeling
4) Introduce risk analysis and backlog priority. (Adding Security to Agile's Scrum 2017.)

6.2 Security testing tools

The OWASP Zap is a well-known tool for discovering Web application vulnerabilities. In addition to the automated testing processes, the OWASP Zap can also be used for the pen testing manually for the Web services. The main features and capabilities of the OWASP Zap are displayed in Table 10. (Owasp Zap 2018.)

Table 10. Main capabilities of OWASP Zap

Function	**Capability**
Intercepting proxy	Intercepts requests and responses between client (browser) and server. Messages can be read or changed.
Active and passive scanner	Passive scanner examines requests and responses and can detect certain kinds of problems. Active scanner provides a wide range of attacks.
Spider	Spider tool automatically discovers new resources (URLs).
Report generation	Capability to generate reports of findings and provide additional information about them.
Brute force	Finding files, even links that are not provided in the actual file. This is based on OWASP DirBuster tool.
Fuzzing	Capability for fuzzing parameters. This function can find vulnerabilities what other tools cannot find.
Extensibility	Zap tool extensions are provided via a separate development program. Extensions can be used to investigate some

	specific feature or area of interest of targeted host.

In addition to the main features of the OWASP Zap, there are additional features available. They are displayed in Table 11 (Owasp Zap 2018).

Table 11. OWASP Zap additional features

Additional function	**Capability**
Auto tagging	Tags messages in Zap to show operator which page has hidden certain fields.
Port scanner	Discovers open ports from the scanned host.
Parameter analyzer	Handles all parameters what application uses and summarize them.
Smart card support	If a tested application is located in a smart card, Zap can read data from the card.
Session comparison	Compares sessions if application under investigation supports multiple roles.
Invoke external apps	Can be done over network, i.e. URL that is scoped to cover the area of interest.
API + Headless mode	Can be used for automated testing.
Dynamic SSL certificates	Can intercept HTTPS traffic and tell browser about a new certificate which can be trusted.
Anti CSRF token handling	Zap can regenerate tokens if applications are using these.

The penetration (or pen test) testing tools can be used to perform authorized simulated attack against to a computer system or host which have an IP address to evaluate and test the security of the system or the host. Global markets provide a wide

range of different types of tools for the pen testing. According to the research goal, the tools for the pen testing and vulnerability scanning are displayed in Table 12. (Walker 2017; Wordfence 2017.)

Table 12. Pen testing and vulnerability scanning

Tool	**Capability and features**
Metasploit	This tool is the most advanced and popular on the market, and it is also the world's most used software for this purpose at the moment. The tool runs a payload code that performs operations on a target machine.
Wireshark	Reference to chapter 3.1. In addition, Wireshark is the world's foremost network protocol analyzer and therefore widely used across the world.
Nessus	Reference to chapter 3.2. In addition, Nessus is the most widely deployed Vulnerability Scanner in the World. Nessus has over 60,000 plugins and its key features are local capability and also client server based capability for performing security checks.
Nmap (Network mapper)	Reference to chapter 3.3. In addition, Nmap can discover what systems are running behind the firewall and which hosts are vulnerable. Nmap can discover services never known to exist. Nmap is widely used by security specialists.
Acunetix	This tool is used primarily as a web vulnerability scanner for web applications.

	This tool provides SQL injection and cross-site scripting testing capability.
w3af	This tool is used to test web applications against attacks such as HTTP requests and malicious payloads.
Monitis	This tool is a cloud based monitoring tool that measures the performance of Web server. It also includes vulnerability scanning feature.
Sqlmap	Open source penetration testing tool. Sqlmap can automate processes of detecting and exploiting SQL injection flaws and taking over database servers. This tool supports a wide range of different databases. Some databases that Sqlmap supports are: MySQL, Oracle and PostgreSQL.
Burb Suite	Reference to chapter 6.1 (WebSocket).
Open VAS	This tool is a framework of several services and tools offering a comprehensive and powerful vulnerability scanning and vulnerability management solution.

Linux KALI, based on the Debian Linux distribution package is designed for the forensic investigation, and the analysis and penetration testing. KALI Linux comes with an extremely wide portfolio of testing tools, and KALI is maintained and financed by the Offensive Security Ltd. KALI can be booted up from a live CD or from USB mass media. KALI is also one platform for the Metasploit framework tools. Table 13 introduces the main capabilities that KALI can support for security testing. Each capability includes around 20 – 70 different tools / binaries which can be used within the technical topic under investigation. (KALI tools 2018.)

Table 13. KALI Linux capabilities

KALI Linux capabilities (main functionality)	**Number of capabilities / main functionality**
Information Gathering	69.
Vulnerability Analysis	29.
Wireless Attacks	53.
Web Applications	44.
Exploitation Tools	21.
Stress Testing	14.
Forensics Tools	23.
Sniffing & Spoofing	32.
Password Attacks	41.
Maintaining Access	18.
Reverse Engineering	11.
Reporting Tools	10.
Hardware Hacking	6.

Some instances have benchmarked the most common and popular tools used by hackers and security professionals. Table 14 introduces tools with the description of their capabilities (Binary Tech 2017).

Table 14. Popular tools for hacking

Tool	**Capability and features**
Hydra	Brute force tool for hacking more than 30 protocols.
Cain & Able	Hacking and password recovery software comes with multiple functionalities for man-in-the middle type of activities. Tool can sniff network and crack encrypted passwords.

7 Survey research

7.1 Multiple-choice questions

The questionnaire question set is built against the main research question and the cornerstones found in the literature during the research to sample the results and provide a wider space for the analysis of the results. The results received from the industry are benchmarked against the results of the literature research. The questionnaire with multiple-choice questions and the form are presented in Appendix 1.

7.2 Free feedback

In the free feedback section, the respondents have the opportunity to provide the answer to the proposed questions freely using their own words. The free feedback questions are set against the research question and the cornerstones found in the literature research. Free feedback questions and the Questionnaire form are presented in Appendix 1.

8 Research results and analysis

8.1 Literature research

8.1.1 Answers to the research questions

The answers to the research question number 1 “What are the best and beneficial points in the software development project where software code shall be security checked?” are discussed below.

It cannot be said straight forward what the best and beneficial points for the security testing are as this definition depends on what programming language and framework are used and what the planned code’s achitechture is. The checkpoints shall be selected in accordance with these limitations. However, it can be generally said that there are some laws of subordination that apply and answers the research question though:

- Software code checks should be implemented from the initial stage of the software development all the way to the production phase. Check metrics can be planning, review or technical measurements

- Security measurements and tools shall be planned and implemented according to the used framework, used programming language, security requirements set for the code, and the threat analysis done prior to the start of the project

- Benefical points for the security testing can be built against the framework, e.g. the SDLC where the software team defines the code check points themselves against the factors mentioned above and the SDLC framework rules which need to be obeyed

- The basic philosofhy for the beneficial code check points can be found in the VAHTI guide. It does not matter what framework is applied as far as the security checks are done each time before moving to the next phase of the lifecycle

- In the agile environment, the security checks shall be excecuted after every sprint when the incremential release of the software is done as the sprints are considered as a short lifecycle inside the e.g. Scrum framework

- In the waterfall model, the benefcial checkpoints are implementation, testing and deployment phases; however, it must be not forgotten that if the requirement and the system design security factors are ignored, there is no use for the testing as the results can be quite predictable

- The code review is considered as one type of the security check and this shall be applied during the security requirements in plan, build, test and review process phases e.g. in the Scrum framework

- In the extreme programming or similar agile framework, the security checks shall be executed after every iteration, architectural spike and small release if the end product of the lifecycle produces reasonable functionalities and the entities for checking. The user stories also applied case by case if a new story has effect on the security factors and if the threat analysis is changed

- The development team shall make the final decision on how many sprints shall be skipped before the security check as the code structure and architechture may vary during the development project

- The development team shall have deep knowledge of the technology used. This sets up training and knowledge requeirements on the team members.

The knowledge of the development team is a key factor when coding secure software as a team; the final factor is who defines the beneficial check points

- The beneficial points for the security testing can also be looked as a selection of the well-known third party up-to-date libraries and the third party modules. This downscales the risk of the security breach.

The answers to the research question number 2 “How big software entities would usually be wise to expose to security check?” are discussed below.

It cannot to be said straightforward how big software entities would it usually be wise to expose to security check as this definition depends on the planned code’s achitechture including third party code modules. Code entities shall be selected in a way that if some of the security issue has managed to pass, it can be recapped with less of delays and minimized extra work for the project.

It can be estimated that e.g. in Scrum, the average amount of the code implemented into the sprint is around 1 – 12 user stories / sprint. Another approach is to scope 1 – 1.5 user stories / one developer / one sprint. These are the definitions how the code is generally manufactured in the agile framework domain. During the development project, the development should be limited up to 1.5 X where X is the head count of the development team members. This defines the amount of the code to be exposed to the security checks if e.g.the SDLC is implemented. If considering the security measures within the project, it is important that the product owner does not push the coders and the Scrum master to execute more user stories than mentioned as the error rate of coding might increase. This can lead to the lack of security or bypassing the security check for higher productivity.

If the Waterfall modell is implemented, this basically means that the code is designed, implemented, tested and deployed. This philosophy means that the testing is done for the whole code rather than to small pieces of it. This is one reason why the waterfall model in the secure software development domain does not necessarily serve the intended purpose of the current security requirements and constantly changing cyber threats.

The answers to the research question number 3 “What are appropriate entities or phases in the framework where the security checks should be focused at to ensure that the final product of the development process fulfills the requirements set on the product (i.e. modules, interfaces or HMI) according to the framework used?” are discussed below.

The answers to this question partially overlap with the answers provided to the Questions 1 and 2. The answer to the question what modules, interfaces or HMIs shall be checked are:

- Security checks for the SQL database injection shall be focused on as the SQL injection is the most common type of attacks used against to the web applications

- The general assumption is that all incoming data is untrusted. This sets the rule that all interfaces which are connected to the external data source, shall be security checked according to the requirements what technology is used within a specific interface implementation

- When coding the HMIs, as many as possible user stories shall be implemented as, if the security testing is done on the requlars basis, exotic user actions causing e.g. a buffer overflow can get caught. This function is related to the penetration testing done for the database and the modules behing the HMI

- Accoring to the interface security, it shall be kept in mind that testing just the external interfaces of the components does not ensure the security of the extracted build of the software entity. All interfaces and functions shall be analyzed and if they or some of them are flagged as a dangerous function, this function should be excluded from the implementation and replaced with a more secure solution if it cannot be fixed

- When executing the unit testing, it is to recommend that the testing focuses on those interfaces or modules that are going to handle the data feeds received from other users or systems.

The answers to the research question number 4 “What would be the best capabilities that can be used for the software security checking during the software development process?” are discussed further below.

Markets provide several kinds of capabilities for the security testing in the cyber domain. For measuring the software code security, the tools are basically divided into four different categories introduced in Table 15. It is important to notice that the capabilities are not just the technical tools as processes, and other methods are also considered as one metric and answer the research question.

Table 15. Answer to the research question 4

Technical measurement	For the interface testing, there are several different tool manufacturers who provide the same capabilities for the technical measurement as other manufacturers. During the research it was easy to spot that the most common used tools were Cain & Able, KALI Linux capabilities and Hydra. Nessus, Wireshark and Nmap were also introduced several times. The same tool set can be used for the SQL injection and vulnerability testing.
Code review	Code compilers provide functionalities for the code review; however, it is also possible to use external viewers such as Codacy, CodeFactor.io, Collaborator, DeepScan or Gamma. The code review is considered as one metric for measuring the code security even though a review might not necessarily included in the cyber testing capabilities. Reviewing the code aims to find the mistakes overlooked in the initial phase of development and improve the overall quality and security of the software. In the SDLC verification phase, fuzz testing is usually included in the security code review when the code is released to the production.

Framework	The secure software development has always been case sensitive and there are some dependencies that might set up the limitations to the security factor execution. This is the main reason why there is no standard model for secure code development. Nevertheless, there are many good frameworks and approaches such as the secure SDLC which defines the basic functions for each phase of the lifecycle. However, what the SDLC does not define are the standard tools and the capabilities needed for the security testing. Developers need to think about the tool set implementation by themselves in accordance with the project and the capabilities provided by their own organization. Microsoft© S-SDLC or the frameworks related to Microsoft© S-SDLC were one of the commonly used frameworks according to the research. Scrum framework was also introduced many times during the research; however, the Scrum does not provide security by default as it has been developed without secure coding practices.
Threat modeling	One of the tools answering the research question is the threat modelling which is the 100 % administrative metric. The threat modelling can be performed in several different ways, and the threat modeler shall select the appropriate method in order to model the threats. One approach to this is to implement the risk management together with the Octave Allegro process. The VAHTI guide provides baselines for the threat modeling and the threat analysis.

8.1.2 Scoping the software module entities

Identified design and implementation phase risks can be mitigated by increasing the level of the expertise and the knowledge for planners and programmers. This thinking supports directly the SDLC cornerstones and reduces the failure of the scope of the software binaries exposed to security testing. When scoping the software module entities exposed to the security checks, the list displayed in Table 16 can be implemented during the work. (De Win 2016.)

Table 16. List of key factors for success

Identification	Factors to be implemented
1	Protect right factors, relevancy and weakest links. Build abuse use cases prior to the project.
2	Execute clear and simple clarity. Put effort to the security requirements.
3	Focus on the code quality and execute the risk analysis.
4	Involve all Stakeholders and Interact with the users regularly during the development. Execute external code reviews.
5	Implement right technology and processes. Implement risk-based security testing.
6	Plan fail safe options.
7	Define defense in the depth.
8	Define the principles of appropriate privileges right. Execute penetration tests.
9	Think well what third party modules are needed to implement.
10	Define clearly how to manipulate the sensitive data.
11	Expose the code to the attacks in the early phases of the development.
12	Think outside the box.
13	Be humble and listen.

14	Put maximum effort to the code reviews.

Another way to find appropriate software and module entities for security testing is to look at the user stories, test cases and test scenarios, and link the requirements set on the project for each test case or user story. After this, it can be easily scoped what software entities are linked to the group of the requirements per each user story. After this work is done, the software modules for security testing can be scoped quite easily. It should be kept in mind though that the list of the metrics displayed in Table 16 might change the definition of the software modules scoped for security testing via user stories if all metrics are applied during the development.

8.2 Survey research

Five multiple choice survey questions were proposed according to the Attachment 1. The survey was aimed at the domestic and foreign industry and it was 100 % anonymous. Because of the mechanism how the survey was sent to the target audience via certain point of contacts and to protect the surveys anonymity, the results are displayed as percentage distribution of the responses.

8.2.1 Anwers to the research question

Table 17 displays the multiple choice questions proposed. Figures 12 – 16 display answers to the survey questions numbers 1 – 5. Domestic and foreign industry is displayed in same chart. By displaying the results in the same chart enables benchmarking the results at the same time.

Table 17. Survey questions

Question number	Question
1	Should your country's national government level cyber strategy be noticed in software development?
2	Should currently effective cyber threat analysis be conducted before starting a new software coding project?
3	How should risk management be taken into account during the software development (choose one or multiple please)?
4	What should data security checks focus on during the software development (choose one or multiple please)?
5	What would be the best phase to perform security check to a software code (choose one or multiple please)?

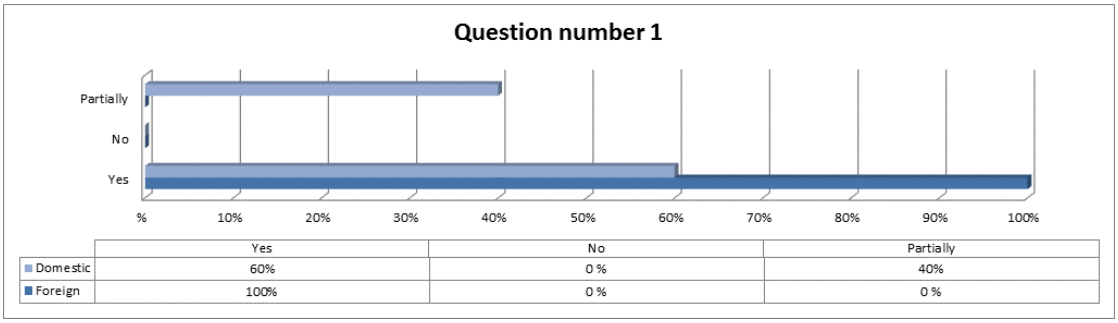


Figure 12. Answers to the question 1.

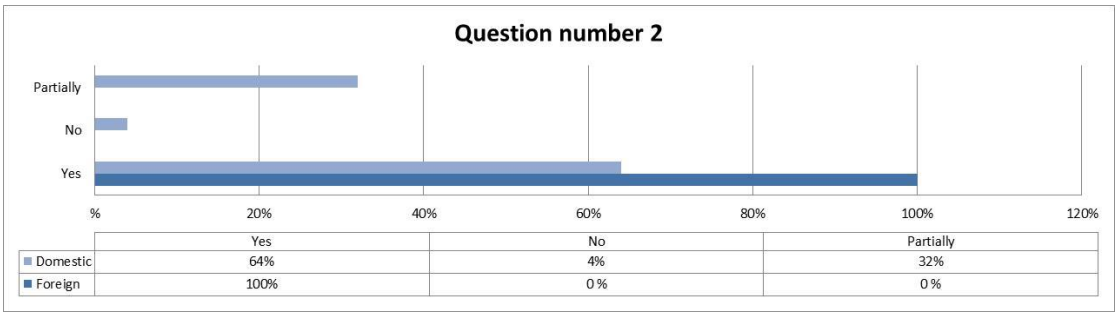
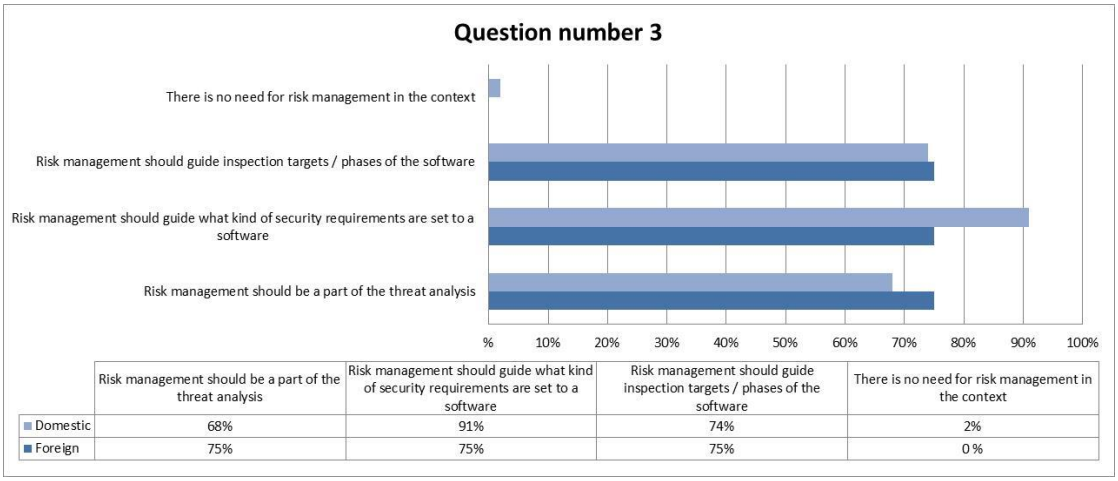
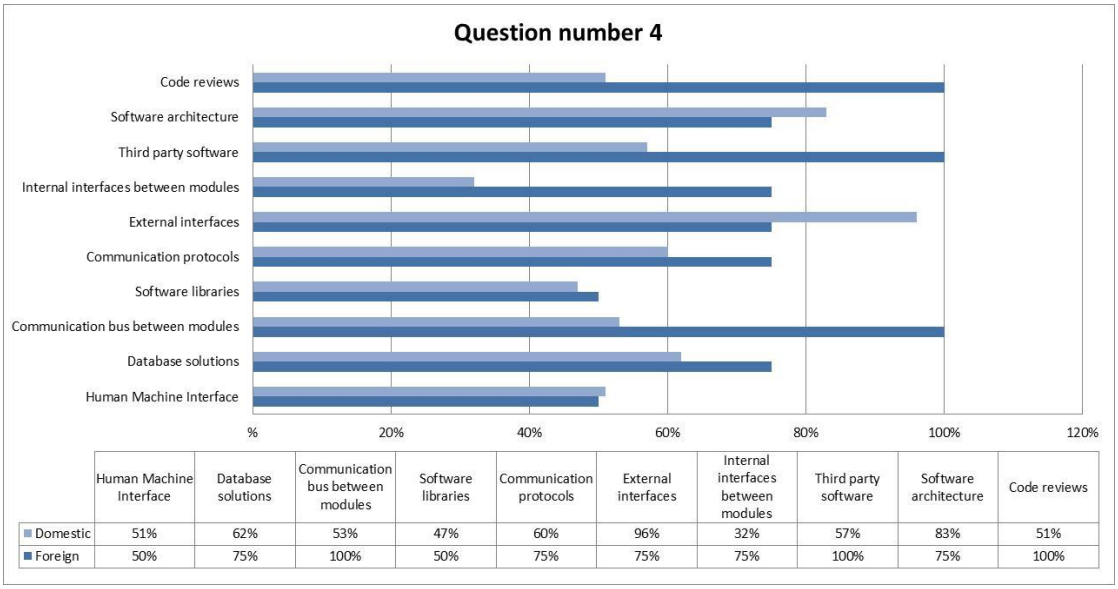


Figure 13. Answers to the question 2.



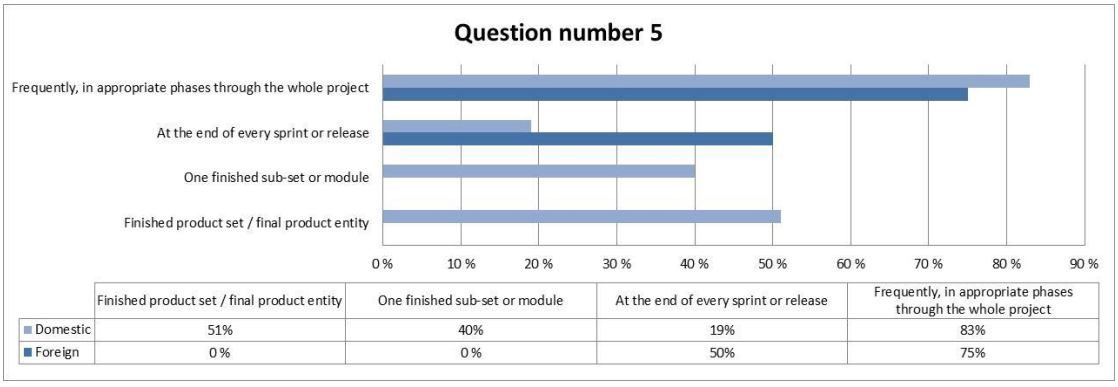
	Risk management should be a part of the threat analysis	Risk management should guide what kind of security requirements are set to a software	Risk management should guide inspection targets / phases of the software	There is no need for risk management in the context
Domestic	68%	91%	74%	2%
Foreign	75%	75%	75%	0 %

Figure 14. Answers to the question 3.



	Human Machine Interface	Database solutions	Communication bus between modules	Software libraries	Communication protocols	External interfaces	Internal interfaces between modules	Third party software	Software architecture	Code reviews
Domestic	51%	62%	53%	47%	60%	96%	32%	57%	83%	51%
Foreign	50%	75%	100%	50%	75%	75%	75%	100%	75%	100%

Figure 15. Answers to the question 4.



	Finished product set / final product entity	One finished sub-set or module	At the end of every sprint or release	Frequently, in appropriate phases through the whole project
Domestic	51%	40%	19%	83%
Foreign	0 %	0 %	50%	75%

Figure 16. Answers to the question 5.

8.2.2 Open feedback

According to the qualitative research methods introduced in chapter 5.2.3, open feedback data was also analyzed according to this research method. After the data transcript was done, Table 18 displays discretionary sampled results of the open feedback which were chronologically repetitive across the open feedback received from survey recipients.

Table 18. Open feedback

Observation number	**Observation**
O_1	Developers should be aware of the security requirements at the practical level.
O_2	Security testing must be automated for continuous integration.
O_3	The focus should be on better management of 3rd party additions and software shall be focused on.
O_4	A software development process needs to be developed to provide a secure development model that covers the entire software lifecycle. Information security should be taken into account throughout the whole life cycle of the project.
O_5	One of the best ways is to use developers who are used to dealing with security as part of their work.
0_6	Data security and security related requirements (e.g. deepened and refined by risk and threat analysis) must be handled in practically the same way as any other non-functional security requirement.
O_7	In a development project software libraries should rely on popular, actively maintained open source libraries.
O_8	The focus should be on training educators and designers in security matters, and training should be increased.
O_9	Automatic quality & security tools shall be implemented to identify security threats and holes in source code and 3rd party libraries. These should be utilized during the sprints in continuous integration and continuous inspection of the development environments.
O_10	A decent architectural design shall be done prior to the project that takes information security into account.
O_11	Improving security with implementing peer-review practices in to a project.

8.3 Assessment of the research results

8.3.1 Literature research

According to the results of the literature research, it can be concluded that the answers to the research questions were reached in an appropriate way. The sources used for the research were considered qualitatively qualified and well known facts around the subject were found. The selected sources can be classified as high quality sources with the appropriate contents and. Sources were considered up to date and impartial as only well-known qualitatively qualified sources were used. The benchmarking of the results is displayed in Table 21, which also supports the assessment of the quality of the results as NCSC-FI is considered the highest authority in the cyber domain in Finland. This definition determines the NCSC-Fi sources as highly reliable.

The goal was to benchmark the research results against to the research conducted around the same subject. During benchmarking and analysis, it was clearly seen that the literature research gave the same type of results as the ones discovered from another publications dealing with the same topic. This assessment confirms the quality of the research results and sources used in the research.

When benchmarking the results against the common standards and guidelines for the cyber security, it can be concluded that the results support the definition of common standards and guidelines regarding the secure software development where vulnerabilities are aimed to be discovered during the software development process (ISO/IEC 27032:2012).

Global industry producing the secure software code is aware of the adapting of the common standards and guidelines around the cyber security to a software development framework. During the assessment of the validity of the research results against the common standards and guidelines, it was clearly seen that the basic philosophy was met where the threats to the security and organizational level security policy supported the secure software development. The facts being displayed in accordance with the research results can be found in and linked to the standards, e.g. steps needed to be taken to prevent a vulnerabilities arising in the software devel-

opment. The threats shall be eliminated within the active steps and after that the remaining threats or factors shall be minimized and monitored. Additionally, incorrect design choices, incorrect specifications and security requirements as well as the lack of the vulnerability introduction were clearly seen from the standards, which reflect directly to the threat analysis and therefore confirm the quality of the research results (ISO/IEC 15408-3:2008).

8.3.2 Survey research

When analyzing the results of the survey research, analyses were done with the capability provided by the Webpropol survey services. Based on a theory, if a quantity value / one answer is more than ten answers, it can be said that the result can be considered being statistically valid. In this case, the amount of each answer fulfills this requirement if not stated other vice later.

Average value can be used to display the weight to which the answers are pinpointing; however, when analyzing a small amount of answers, calculating average values can become untrustworthy. Average confidence interval value indicates the average value variance with 95 % accuracy. A big variance value indicates that the result cannot be statically trusted.

Table 19 displays domestic statistic values of the survey extracted from the Webpropol service.

Table 19. Assessment of the domestic survey answers

Question number	Average	Average confidence interval	Median	Standard deviation
1	1,808511	1,52 – 2,09	1	0,992106
2	1,680851	1,41 – 1,95	1	0,934982
3	2,045045	1,9 – 2,19	2	0,802127
4	5,636691	5,3 – 5,97	6	2,834996
5	2,692308	2,43 – 2,95	3	1,270978

Table 20 displays foreign statistic values of the survey extracted from the Webpropol service.

Table 20. Assessment of the foreign survey answers

Question number	Average	Average confidence interval	Median	Standard deviation
1	1	1 – 1	1	0
2	1	1 – 1	1	0
3	2	1,43 – 2,57	2	0,866025
4	5,83871	4,81 – 6,86	6	2,910862
5	3,6	3,12 – 4,08	4	0,547723

When analyzing the confidence interval of the answers, it can be calculated that according to Table 19, in domestic answers the interval maximum difference between smallest and biggest values is less than 0,67. This leads to the conclusion that the variance between the answers is very small and answers can be considered trustworthy. According to Table 20, it can be calculated that in foreign answers the interval maximum difference between smallest and biggest values is less than 2,05. This leads to the conclusion that the variance between the answers is bigger than in the domestic answers and therefore, the answers are not that valid than domestic answers.

If the median values are compared between Table 19 and Table 20, it can be clearly seen that there is a difference of just one number, which leads to the conclusion that based on median values, the answers can be considered as equally valid.

If the standard deviation value is big or bigger, this indicates that the respondents have disagreed with the answers. If comparing the smallest and the biggest deviation values in Table 19, it can be seen that in domestic answers the deviation values are between 0,802127 and 2,834996. If comparing the same values in Table 20, it can be seen that in foreign answers the deviation values are between 0 and 2,910862. This leads to the conclusion that the answers follow a certain baseline and therefore, the results can be considered quite trusted and fact based.

When assessing and comparing the answers between foreign and domestic industry, Figure 17 displays the percentage distribution between the questions 1 and 2.

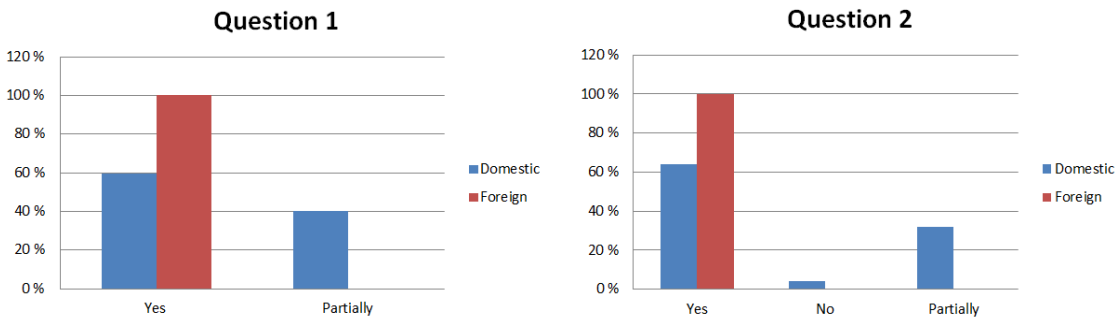


Figure 17. Benchmark between questions 1 and 2.

Figure 18 displays the benchmark results in question 3.

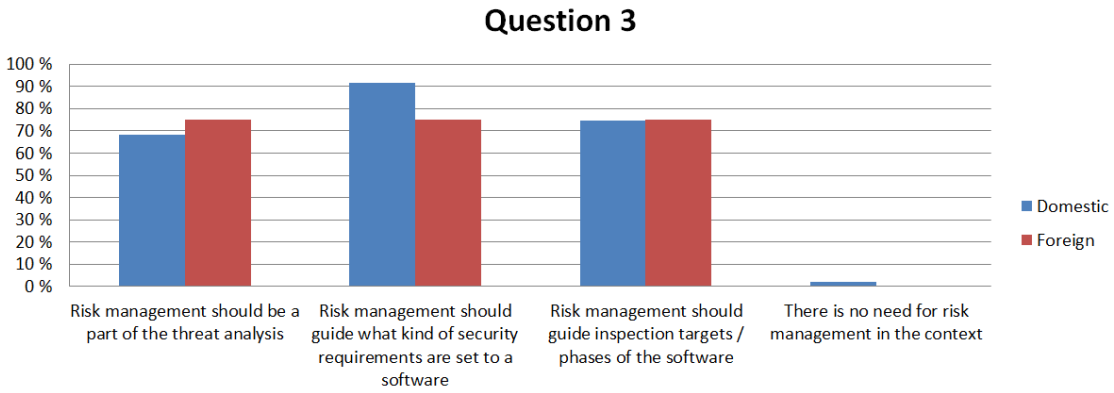


Figure 18. Benchmark question 3.

Figure 19 displays the benchmark results in question 4.

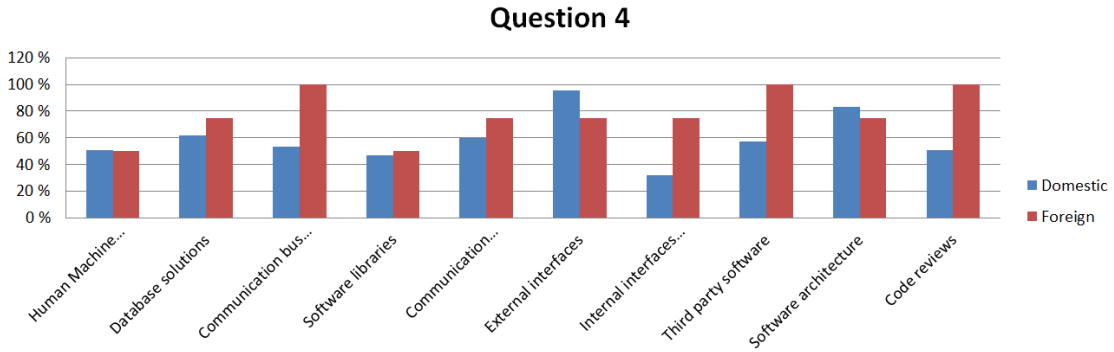


Figure 19. Benchmark question 4.

Figure 20 displays the benchmark results in question 5.

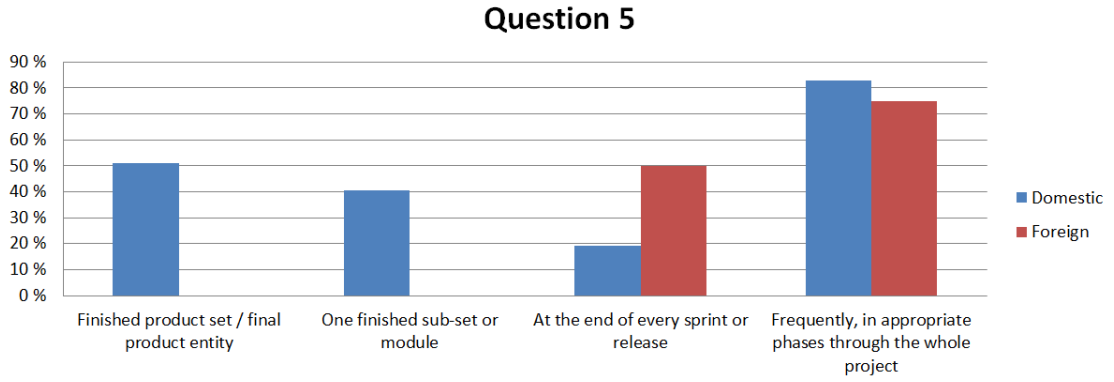


Figure 20. Benchmark question 5.

The information in the key findings displayed in Figures 17 – 20 discovers that the answer percentages in foreign and domestic industry are quite the same. Both industries saw the importance of frequently performed security tests in appropriate phases through the whole project. Pinpointing security tests to software architecture, software libraries and databases was also seen as quite similar between the industries.

The most significant variations between the industries were third party software, internal interfaces and communication data busses. The reason for this might be that the survey respondents work in different positions with each their own background for the secure software development.

8.4 Connecting survey results to the theory and literature reseach

During the assessment of the survey research results, the connection to the literature research and theoretical framework of the thesis was connected iteratively. In the literature research, certain secure software frameworks were introduced and during assessment and benchmarking the literature research results, it was clearly seen that there is no solid standard for secure software development at the moment. Continuous testing and training were also points which were constantly noticed. Threat model and secure architecture were also points which were constantly spotted.

This leads to the analysis that all findings introduced in Table 18 are found in the literature research and some parts are directly connected to the introduced theory. The open feedback results of the survey were analyzed according to the research methods displayed in chapter 5.2.4, and the answers shown in Figure 15 mainly correlate with the information shown in Table 2.

From the open feedback section in the survey research it can be analyzed that observation O_4 displayed in Table 18 correlates with the data displayed in Figure 8, Figure 10, Figure 11 and Table 9.

8.5 Answers to the research questions

The answers to the research question number 1 “What are the best and beneficial points in the software development project where software code shall be security checked?” are discussed below:

- Answers can be found in Figure 15, which gives the basis to consider all proposed points to be points that should be beneficial to check. The severity of the answers varies so it should be planned against the software project what should be the priority order for checking the points against the project framework. This analysis also connects with the answer provided in chapter 8.1.1.

The answers to the research question number 2 “How big software entities would usually be wise to expose to security check?” are discussed below:

- Figure 16 provides the answer to this question, and it can be clearly seen that it is mandatory to check the code security after every sprint through the whole coding project. What this answer does not straight state is the exact amount of the code that would be beneficial to check. The amount of the code to be checked always depends on what framework is used, what the planned architecture of the code is and what the size of the project is.

The answers to the research question number 3 “What are appropriate entities or phases in the framework where the security checks should be focused at to ensure that the final product of the development process fulfills the requirements set on the product (i.e. modules, interfaces or HMI) according to the framework used?” are discussed below:

- According to the open feedback, it can be clearly concluded that the points displayed in Table 18 provide the answers to this question. The main message

here is to conduct security measures constantly from the planning phase to the coding phase as well as to the deployment and maintenance phase.

The answers to the research question number 4 “What would be the best capabilities that can be used for the software security checking during the software development process?” are discussed below:

- The open feedback section of survey research was built up with the hopes to get answers to this question; however, no specific manufacturers or testing tool names were stated.

9 Conclusions

9.1 Literature research

One clear conclusion is that there is no certainty of the best framework model for the secure code development that can be natively used within every software project. The frameworks and measurement tools should be selected case by case in accordance with what is more suitable for the available resources and how organizational processes support the selected framework. Some corporations have self-developed secure software development models, which are applicable in accordance with organizational processes.

During the research, the Finnish Communications Regulatory published a document which defines the guidelines for secure software development towards approval. As NCSC-FI has been the highest authority in the cyber domain in Finland, this publication was omitted from the literature research, so that the validity of the research results could be benchmarked against it. This led to the conclusion that the results of the research support NCSC-FI baseline in accordance to Table 21 which displays the cross reference matrix where the NCSC-FI baseline and the results of the research are benchmarked. (Finnish Communications Regulatory Authority NCSC-FI 2018.)

Table 21. The benchmark results

NCSC-FI publication topics for the secure software development towards approval.	**Covered by the results of the literature research.**	**Covered by the results of the survey research.**	**Scoped in the future research chapter.**	**Not included in the scope of the research.**
Security requirements	x			
Threat modeling	x		x	
Built-in vs. add-on security				x
Privacy				x

Minimize attack surface	x (partially)			x
Establish secure defaults	x (partially)			x
Sanitize input	x			
Separate duties	x			
Give minimum privileges	x			
Defend in depth	x			
Fail securely	x			
Don’t trust external services	x			
Avoid security by obscurity or secrecy			x	
Keep it simple				x
prepare to fix security issues correctly	x			
Platform choice				x
Software components	x			
Supply chain			x	
Cryptography	x			
Manage dependencies	x			
Conduct code reviews	x			
Continuous integrations	x			
Fuzzing	x			
Penetration testing	x			
Stress or torture testing	x			
Reverse engineering	x			
Testing summary				x
Deployment			x	

Another clear conclusion is that the threat analysis with the risk management shall be conducted prior to the coding process and all security measures shall be embedded in the software development process regularly. What tools and measures shall be used cannot to be said straightforward as these factors depend on the used programming language and planned code architecture. There are still some basic measures, although that applies to every code development project e.g. code reviews and the usage of testing tools, which is also to be considered as a quality control of the software development project from the security point of view.

The threat models should be built and maintained during the software development process, and the models should be designed in accordance with the standard process integration. By implementing this method, the number of security vulnerabilities can be reduced and the enterprise's security requirements and security constrains as well as business needs can be met (IriusRisk 2018).

The literature research method is a good way to find answers to the research question; however, during the analysis of the results, it was discovered that the interview research method would also have been one way to discover answers. On the other hand, interviewing might provide false results if the audience is chosen incorrectly and if an interviewee is giving sweeping answers being under confidentiality agreement or company secrecy agreement.

During the research, it was quite fast discovered that many other topics around the research questions shall be investigated in the future. This set the challenge to scope the research to apply to and focus only on the research questions applied. The literature research is also a good way to find answers, as well known and trusted sources are widely available and easy to access.

9.2 Survey research

One clear conclusion is that the answers displayed in Figure 15, which provides the answers to the question number 4, were all considered important points. It can be seen that the severity and priority level of the answers provided vary; however, there was not a single topic, which the respondents did not see as an important point in the scope. If the four most important points were to be shown from the domestic point of view, these are:

1) External interfaces
2) Software architecture
3) Database solutions
4) Communication protocols.

From the foreign industry point of view, the four most important points were:

1) Communication bus between modules
2) Third party software
3) Code reviews
4) All rest points as they were answered as equal.

The biggest variance between the answers displayed in Figure 15 was:

1) Code reviews
2) Third party software
3) Internal interfaces between the modules
4) Communication bus between modules.

Despite the priority and the severity levels of the answers, the variance between the answers was that small that it can be concluded that all points should be considered as important points when the software code is under the secure development process. It can be concluded that all points shall be taken in to consideration according

to the software architecture when planning and executing the secure software development project.

As it can be seen from Figure 17, a bigger variance was discovered between the answers received from the questions one and two, where foreign industry did not see the threat analysis and the governmental level cyber strategy implementation as that important as the domestic industry.

In conclusion, based on the findings displayed in chapter 9.2, it can be analyzed that variance phenomenon between the answers can be explained with cultural differences between domestic and foreign industry and with different backgrounds of the respondents. An open feedback displayed in Table 18 can be interpreted in the way that all answers received from the survey questions can be found in the final results, which were constantly repeated among the respondents.

9.2.1 Critical analysis of the survey questions

When analyzing the survey research data received in a critical point of view, it can be benchmarked that were the survey answer series scoped in a proper way and were the questions proposed selected in a way that selection was serving the goal of the research? It can also be analyzed how the questions proposed and answers received were analyzed in a proper way.

The questions proposed were selected using well-known variables and well-known road map of the environment software development architecture. Prior to building a set of the questions, it was surveyed on a detailed level what main points should be taken into consideration when a software code is developed into the environment. This leads to the conclusion that the questions were selected with the critical assessment taking place.

In open feedback section, the question was proposed as a high-level question to provide more space to the respondents. If it were to be analyzed whether the question was proposed in a proper way or not, the answer might be “yes and no”, because the open feedback section provided the right data answering to the research question. However, still there were some functionalities left out of the results as the goal also was to collect information about the actual testing capabilities. This leads one to think that the open feedback question might have been proposed in a slightly different way. When thinking critically from the other side of this question, it can be also concluded that the respondents did not want to expose their actual capabilities and tool manufacturers in their answers.

10 Discussion

During the literature research and also the survey research, it was well learned how a software development project should be done in a secure way to response to the constantly developing cyber threats. In addition to increasing knowledge around this thesis topic, Finnish Defence Forces Logistics Command as the assignor of this thesis also wanted to send out the message to foreign and domestic industry that the cyber security around the software procurements will be focused on more and more in the future.

The interaction with the industry using the survey research was extremely valuable in order to get the situational picture how the industry saw the questions proposed. The results and analysis conducted in this thesis will be put into practice when requirements are set to future procurements.

11 Further research

11.1 Security tools

As machine learning and artificial intelligence are penetrating every industry area, it can be clearly seen that these capabilities can also be exploitable in the cyber security domain. During the secure software development, these capabilities can be used for detecting malicious code and coding errors in real time. Before this capability can be trusted sufficiently, there might be some challenges for the development of algorithms to work in a way that the machine learning and artificial intelligence can be considered as a security tool set. This is clearly one research area which should be focused on in the future. One interesting point around the topic is how these capabilities are going to be connected to agile secure software development frameworks. (Tuovinen 2018, 4.)

Another area of a future research scope is to find an appropriate tool set for security testing during the software development. By achieving this, certain steps are needed such as the definition of a common goal, the creation of a criteria matrix and definitions of the main problem that needs to be solved. The next phase is to define the test scenarios, test out the defined scenarios around the topic and build the product roadmap. After this, one should focus on maintaining the future aspirations. Defining the right tool set for the security testing is a key function when a secure software development project is to be launched. (Rapid7 2013.)

11.2 Development according to the different frameworks

The agile software development can be carried out in accordance with the various frameworks embedded into a software development projects. Currently, it is not recognized that a certain framework is better than another. The framework used always depends on the type of the code under development. Some manufacturers have self-developed secure software development life cycle processes; however, this

is still one research area that might be good to focus on in the future after secure software development has achieved some kind of wider standard baseline (De Win 2016).

With threat analysis being one of the key factors for successful secure code development and scoping critical infrastructure systems with the long lifecycle, it can be seen that the future research focus shall be set on the solutions proposed around the subject. One area of the future research is to look how the implementation of a threat analysis can be more deeply conducted from the end user's perspective. In addition of this, the focus shall also be put on how the data from the previous results of the threat analysis can be used and updated during the process where iterations of detecting threats are taking place and how to manage the subject of threat changes to the systems in production. This thinking should be implemented into a long-term testing concept embedded in a SDLC process to reduce security threats in lifecycle products with a notably long lifetime. This thinking is described in Figure 21 in accordance with critical interface testing. (Huopio 2018, 18 and 19.)

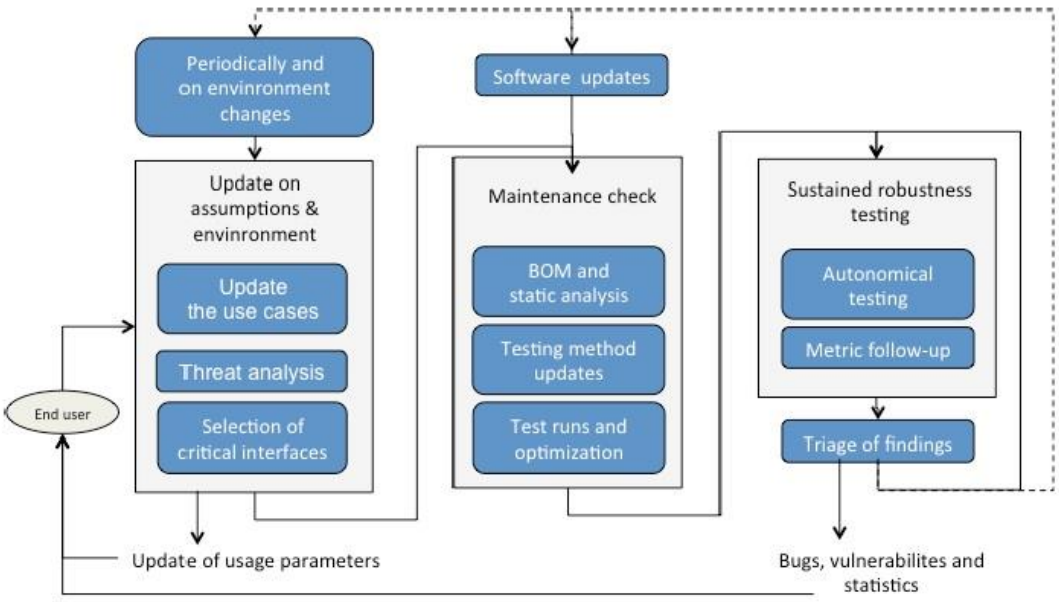


Figure 21. Testing process for critical interfaces.

11.3 How to respond to developing cyber threat theatre in the future

When planning secure software code and cyber threat mitigation measures, it is very important that threat analysis is done prior to the coding process and maintained during the project. Threat analysis shall be connected with the risk management process for identifying hot topics around the subject. One future research area is to discover what kind of mechanisms, tools and processes should be brought together for building a better situational picture of changing hybrid cyber threats where every threat vector is scoped and maintained.

Another interesting threat vector discovered during the research was how to build attacks into the supply chain during the software development and how to affect the code via people who are producing the code. Developers usually have access to several assets with administrative privileges. Access to a developer workstation might open the doors for further assets where code is developed. This threat vector leads to difficult questions around the trust, such as: can developers' laptop be considered as a trusted asset or can a build-server connected to a developer´s laptop also be considered as a trusted asset? Can one trust the developers as persons that obey organization's instructions and are aware of social engineering threats? Can the developers be trusted to act correctly if some nice person they met on an airplane during the business trip gives them a USB stick? Humans are still considered as the biggest threat vector and finding the solutions for mitigating this threat vector is definitely one of the biggest research questions in the future (Disobay 2018).

References

Adding Security to Agile's Scrum. 2017. Software Engineering Institute | Carnegie Mellon University. 2017. Accessed 24 October 2018. Retrieved from https://www.youtube.com/watch?v=jM8BeY2pXPI

Agile Business Consortium. 2018. What is DSDM. Accessed 2 October 2018. Retrieved from https://www.agilebusiness.org/what-is-dsdm

Aura, T. 2017. Threat analysis. Aalto University. Course presentation. Accessed 24 December 2018. Retrieved from https://mycourses.aalto.fi/pluginfile.php/537035/mod_resource/content/1/01-security-threat-analysis.pdf

Binary Tech. 2017. Top 5 Hacking Software Used by Hackers and Security Professionals. Accessed 24 December 2018. Retrieved from https://www.youtube.com/watch?v=F_ng9107TsQ

Bloomberg Businessweek. 2018. The Big Hack: How China Used a Tiny Chip to Infiltrate U.S. Companies. Accessed 30 October 2018. Article on bloomberg.com webpage. Retrieved from https://www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america-s-top-companies

Center for strategic & international studies. 2018. Significant Cyber Incidents. Article on csis.org webpage. Accessed 30 December 2018. Retrieved from https://www.csis.org/programs/cybersecurity-and-governance/technology-policy-program/other-projects-cybersecurity

Cisco Systems. 2018. What is cybersecurity. Accessed 30 October 2018. Retrieved from https://www.cisco.com/c/en/us/products/security/what-is-cybersecurity.html

De Win, B. 2016. Secure Development LifeCycles (SDLC). Accessed 5 October 2018. Retrieved from https://handouts.secappdev.org/handouts/2016/Bart%20De%20Win/SecAppDev2016%20-%20SDLC%20Session%20Bart%20De%20Win%20v1.0.pdf

Disobay 2018. Software Developer as an attack vector. Seminar material. Antti Virtanen. Accessed 30 October 2018. Retrieved from https://www.youtube.com/watch?v=3rOSrNpjj9o

Extremeprogramming. 2013. Extreme Programming: A gentle introduction. Accessed 7 October 2018. Retrieved from http://www.extremeprogramming.org/

Finnish Communications Regulatory Authority. NCSC-FI. 2018. Secure development: towards approval. Accessed 22 December 2018. Retrieved from https://www.kyberturvallisuuskeskus.fi/sites/default/files/media/publication/Turvallinen_tuotekehitys_003_2018J.pdf

Finnish Transport and Communications Agency. CERT-FI. 2018. Accessed 7 October 2018. Retrieved from https://www.traficom.fi/en/communications/cyber-security

Finnish National Security Authority. National Security Auditing Criteria (KATAKRI). 2015. Accessed October 2018. Retrieved from Aura, T. 2017. Ministry Of Finance. VAHTI instructions. 2016. Accessed 7 October 2018. Retrieved from https://www.vahtiohje.fi/web/guest/home

Huopio, S. Finnish Defence Research Agency. 2018. 23rd ICCRTS "Multi-Domain C2". Interoperability, Integration and Security. Thou shalt not fail – Targeting Lifecycle-Long Robustness while being vigilant for the Black Swans. Plenary sessions publication. Accessed 22 December 2018.

Homanen, M. 2017. Suunnittelumallien hyödyntäminen tietoturvallisessa ohjelmistokehityksessä. Tietotekniikan kandidaatintutkielma. Jyväskylän yliopisto, tietotekniikka. Accessed 7 October 2018. Retrieved from https://jyx.jyu.fi/handle/123456789/54296

IriusRisk. 2018. Create threat models and manage application security risk throughout the software development process. Accessed 16 December 2018. Retrieved from https://continuumsecurity.net/threat-modeling-tool/

IEEE 39th ACSAC. 2015. Case Base for Secure Software Development Using Software Security Knowledge Base. 2015 IEEE 39th Annual Computer Software and Applications Conference. Accessed 16 December 2018. Retrieved from https://ieeexplore.ieee.org/document/8093383

Implementation Programme for Finland's Cyber Security Strategy for 2017–2020 2018. Accessed 7 October 2018. Retrieved from https://turvallisuuskomitea.fi/wp-content/uploads/2018/10/Implementation-programme-for-Finlands-Cyber-Security-Strategy-for-2017-2020-final.pdf

ISO/IEC 27032:2012. Information technology -- Security techniques -- Guidelines for cybersecurity. Accessed 7 October 2018. Retrieved from https://online.sfs.fi/fi/index/tuotteet/ISO/ISO/ID9998/2/192337.html.stx

ISO/IEC 15408-3:2008. Information technology -- Security techniques -- Evaluation criteria for IT security -- Part 3: Security assurance components. Accessed 7 October 2018. Retrieved from https://online.sfs.fi/fi/index/tuotteet/ISO/ISO/ID9998/1/106230.html.stx

Kaspersky. 2018. What is cyber-security. Accessed 20 of October 2018. Retrieved from https://www.kaspersky.com/resource-center/definitions/what-is-cyber-security

Kuula, A. 2015. Tutkimusetiikka, aineistojen hankinta, käyttö ja säilytys. Accessed 20 October 2018. Retrieved from https://janet.finna.fi/Record/janet.328266

KALI tools. 2018. Kali Linux Tools Listing. Accessed 20 November 2018. Retrieved from https://tools.kali.org/tools-listing

Karppinen, N. 2014. Ohjelmistotuotannon tietoturva. Opinnäytetyö, AMK. Lapland University of Applied Sciences. Accessed 1 October 2018. Retrieved from https://www.theseus.fi/handle/10024/80816

Keiski, T. 2013. Developing security in the system development life cycle. Master's thesis: Agenteq Solutions Oy. Accessed 20 October 2018. Retrieved from https://www.theseus.fi/handle/10024/62644

Kirmanen, J. 2016. Kyberhyökkäysasiantuntija, Silverskin Information Security Oy. Tietoturva ohjelmistokehityksessä. Article on sytyke.org webpage Accessed 1 De-

cember 2018. Retrieved from http://www.sytyke.org/tietoturva/tietoturva-ohjelmistokehityksessa/

Koistinen, P. 2013. Security Model for Agile Software Development. Master's Thesis. Accessed 20 October 2018. Retrieved from https://www.theseus.fi/handle/10024/65247

Kuosmanen, H. 2016. Security Testing of WebSockets. Master's thesis, Technology, communication and transport, Cyber Security, Master's Degree Programme in Information Technology. Accessed 1 October 2018. Retrieved from https://www.theseus.fi/handle/10024/113390

KvantiMOTV. 2007. Mittaaminen, menetelmaopetus. Accessed 1 October 2018. Retrieved from https://www.fsd.uta.fi/menetelmaopetus/mittaaminen/mittaaminen.html

KvantiMOTV. 2007. Mitä laadullinen tutkimus on. Lyhyt oppimäärä. Accessed 1 December 2018. Retrieved from https://www.fsd.uta.fi/menetelmaopetus/kvali/L1_2.html

KvantiMOTV. 2008. Mittaaminen: Mittarin luotettavuus. Accessed 1 October 2018. Retrieved from https://www.fsd.uta.fi/menetelmaopetus/mittaaminen/luotettavuus.html

KvantiMOTV. 2010. Kyselylomakkeen laatiminen. Accessed 1 December 2018. Retrieved from https://www.fsd.uta.fi/menetelmaopetus/kyselylomake/laatiminen.html

Microsoft. 2012. Microsoft Security Development Lifecycle (SDL) Process Guidance - Version 5.2. Accessed 1 December 2018. Retrieved from https://www.microsoft.com/en-us/download/details.aspx?id=29884

Mini, Z. Feng, Z. 2016. A Step by Step Guide to Building Secure Software. Proceedings of IMCIC – ICSIT. Department of Computer Science, the University of Alabama in Huntsville. Article. Accessed 25 October 2018. Retrieved from http://www.iiis.org/CDs2016/CD2016Spring/papers/ZA990QL.pdf

Nessus. 2018. Accessed 25 November 2018. Retrieved from https://www.ohjelmistot.com/nessus-professional

Nmap. 2018. Accessed 25 November 2018. Retrieved from https://nmap.org/

North Atlantic Treaty Organisation. 2018. Cyber defence. Accessed 25 December 2018. Retrieved from https://www.nato.int/cps/en/natohq/topics_78170.htm

Oracle. 2018. Secure Coding Standards. Accessed 9 December 2018. Retrieved from https://www.oracle.com/corporate/security-practices/assurance/development/

Owasp Zap. 2018. Accessed 25 November 2018. Retrieved from https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

OWASP. 2015. The OWASP Application Security Verification Standard. Accessed 25 October 2018. Retrieved from https://www.owasp.org/images/6/67/OWASPApplicationSecurityVerificationStandard3.0.pdf

Puhakainen,A. & Sääskilahti, J. 2012. Ericsson Network Security Competence Hub. RSA Conference Europe presentation. Accessed 25 December 2018. Retrieved from https://www.rsaconference.com/writable/presentations/file_upload/asec-107.pdf

Puolustusvoimat kohottaa valmiuttaan kyberuhkia vastaan. 2016. Article on mtvuutiset.fi webpage. Accessed 25 December 2018. https://www.mtvuutiset.fi/artikkeli/ksml-puolustusvoimat-kohottaa-valmiuttaan-kyberuhkia-vastaan/6197082#gs.8jahetY

Puolustusvoimien logistiikkalaitos. Accessed 25 October 2018. Retrieved from https://puolustusvoimat.fi/tietoa-meista/logistiikkalaitos

Prime Minister`s Office Finland. 2012 Finnish Security and Defence Policy 2012. Accessed 25 October 2018. Retrieved from https://vnk.fi/julkaisu?pubid=2205

Rautio, P. 2007. Tiedon hakeminen teksteistä. Accessed 25 October 2018. Retrieved from http://www2.uiah.fi/projects/metodi/040.htm#ainkasit

Rapid 7. 2013. Evaluating Security Tools. Accessed 25 December 2018. Retrieved from https://www.youtube.com/watch?v=Pjq2RLES9lY

Råman, J. 2006. Regulating secure software development: analysing the potential regulatory solutions for the lack of security in software. Väitöskirja, Lapin Yliopisto. Accessed October 25 2018. Retrieved from http://lauda.ulapland.fi/handle/10024/61695

ScienceSoft. 2018. A Step-by-Step Guide to Secure Software Development. Accessed 25 October 2018. Retrieved from https://www.scnsoft.com/blog/secure-software-development-guide

SFS-EN 301549:2018:en. 2012. Accessibility requirements for ICT products and services. Accessed 16 December 2018. Retrieved from https://online.sfs.fi/fi/index/tuotteet/SFS/CEN/ID2/3/706410.html.stx

Sibbigui, ST. 2017. Significance of Security Metrics in Secure Software Development. Department of Computer Science, College of Computer Science & Information Systems Jazan University, KSA. Article Accessed 25 November 2018. Retrieved from https://www.researchgate.net/publication/319614081_Significance_of_Security_Metrics_in_Secure_Software_Development

Smartbear. 2018. Best Practices for Code Review. Accessed 25 December 2018. Retrieved from https://smartbear.com/learn/code-review/best-practices-for-peer-code-review/

Software Engineering Institute. Carnegie Mellon University. SEI CERT C Coding Standard: Rules for Developing Safe, Reliable, and Secure Systems (2016 Edition). 2016. Accessed 22 December 2018. Retrieved from https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=454220

Software testing help. 2018. Top 10 Most Popular Code Review Tools for Developers and Testers. Accessed 25 November 2018. Retrieved from https://www.softwaretestinghelp.com/code-review-tools/

Statista. 2018. Number of breaches and records exposed 2005-2018. Accessed 25 October 2018. Retrieved from https://www.statista.com/statistics/273550/data-

breaches-recorded-in-the-united-states-by-number-of-breaches-and-records-exposed/

Statista. 2018. Proposed budget of the U.S. government for cyber security in FY 2017 to 2019 (in billion U.S. dollars). Accessed 25 October 2018. Retrieved from https://www.statista.com/statistics/675399/us-government-spending-cyber-security/

The Guardian. 2018. Cyber-attack risk on nuclear weapons systems 'relatively high' – thinktank. Article on theguardian.com webpage. Accessed 25 October 2018. Retrieved from https://www.theguardian.com/technology/2018/jan/11/cyber-attack-risk-on-nuclear-weapons-systems-relatively-high-thinktank

The Scrum Guide. Official Scrum Guide. 2017. Accessed 25 October 2018. Retrieved from https://www.Scrumguides.org/index.html

Touko, H. 2015. Tietoturvan parhaat käytänteet ohjelmistokehityksessä. Tietoturvan parhaat käytänteet ohjelmistokehityksessä. Tietotekniikan kandidaatintutkielma . Jyväskylän yliopisto, tietotekniikan laitos. Accessed 25 December 2018. Retrieved from https://jyx.jyu.fi/handle/123456789/46180

Tuovinen, J. 2018. Haitallisen JavaScript-koodin tunnistaminen koneoppimismenetelmiä käyttäen. Jyväskylän yliopisto Informaatioteknologian tiedekunta, Tietotekniikan pro gradu –tutkielma. Accessed 25 October 2018. Retrieved from https://jyx.jyu.fi/handle/123456789/59736

Turvallisuuskomitea. 2013. Finland`s Cyber security Strategy. Accessed 25 October 2018. Retrieved from https://turvallisuuskomitea.fi/en/finlands-cyber-security-strategy/

Tutorialspoint. 2018. SDLC – waterfall model. Accessed 25 October 2018. Retrieved from https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm

Uzility, 2017. Introduction to Scrum. Accessed 25 December 2018. Retrieved from https://www.youtube.com/watch?v=9TycLR0TqFA

VAHTI 1/2013. Sovelluskehityksen tietoturvaohje. 2013. Accessed 25 October 2018. Retrieved from https://www.vahtiohje.fi/web/guest/vahti-1/2013-sovelluskehityksen-tietoturvaohje

Valli, R. Aarnos, E. 2018. Ikkunoita tutkimusmetodeihin : 1, Metodin valinta ja aineistonkeruu : virikkeitä aloittelevalle tutkijalle Accessed 25 November 2018. Retrieved from https://janet.finna.fi/Record/janet.338221

Veracode. Secure coding best practices handbook. A developer`s guide to proactive controls. Accessed 25 October 2018. Retrieved from https://info.veracode.com/secure-coding-best-practices-hand-book-guide-resource.html

Walker, J. 2017. Top 7 penetration testing tools for the small business. Article on monitis.com webpage. Accessed 25 October 2018. Retrieved from http://www.monitis.com/blog/top-7-penetration-testing-tools-for-the-small-business/

Wireshark. 2018. Accessed 25 November 2018. Retrieved from https://www.wireshark.org/

Wordfence. 2017. 51 Tools for Security Analysts. Article on wordfence.com webpage. Accessed 25 October 2018. Retrieved from https://www.wordfence.com/blog/2017/04/tools-for-security-analysts/

Yurdakul, S. 2013. Tietoturvallinen web-ohjelmointi. Insinöörityö, AMK. Metropolia Ammattikorkeakoulu, tietotekniikan koulutusohjelma. Accessed 25 December 2018. Retrieved from https://www.theseus.fi/handle/10024/56746

11.5.2007/551. Laki puolustusvoimista. Accessed 25 December 2018. Retrieved from https://www.finlex.fi/fi/laki/ajantasa/2007/20070551

12 Appendices

Appendix 1. Questionnaire form

Software procurements FDFLC

1. Should your country´s cyber strategy at the national government level be taken into account in software development? *

- ○ Yes
- ○ No
- ○ Partially

2. Should an effective cyber threat analysis on current threats be conducted before starting a new software coding project? *

- ○ Yes
- ○ No
- ○ Partially

3. How should risk management be taken into account during the software development (Choose one or multiple answers, please)? *

- ☐ Risk management should be a part of the threat analysis
- ☐ Risk management should guide what kind of security requirements are set to a software
- ☐ Risk management should guide inspection targets / phases of the software
- ☐ There is no need for risk management in the context

4. What should data security checks focus on during the software development (choose one or multiple answers, please)? *

- ☐ Human Machine Interface
- ☐ Database solutions
- ☐ Communication bus between modules
- ☐ Software libraries
- ☐ Communication protocols
- ☐ External interfaces
- ☐ Internal interfaces between modules
- ☐ Third party software
- ☐ Software architecture
- ☐ Code reviews

5. What would be the best phase to perform a security check to software code (choose one or multiple answers, please)? *

- ☐ Finished product set / final product entity
- ☐ One finished sub-set or module
- ☐ At the end of every sprint or release
- ☐ Frequently, in appropriate phases through the whole project

6. Please use your own words to describe what would be the best way to manufacture and develop secure and cyber threat resistant software.

Appendix 2. Survey email

The results of this survey will be utilized in the FDFLC (Finnish Defense Forces Logistics Command) software procurements bidding competitions. The answers in the survey are dealt with 100 % anonymity, and personal or organizational data is not collected.

The goal of the survey is to build a picture how industry sees the importance of secure software development. The survey consists of just 5 short questions, and answering the survey takes only few minutes of your time. You also have possibility to provide open feedback.

Please forward the link below in your organization to as many experts as possible. The survey is aimed at Scrum masters, programmers, product owners and data security specialists.

The survey can be found on https://www.webropolsurveys.com

The survey will close on 30th of March 2019.

The survey is part of my Master's thesis and my studies in Master's Degree Programme in Cyber Security. The answers will be published as a part of my thesis at https://www.theseus.fi/

If you need further information, I am happy to provide more details about the thesis and the survey.

Thank you in advance.

Mr. Mikko Sara

The Finnish Defense Forces Logistics Command