Nora Bodin
Hanna Kai Barstad Golberg

Software Security Culture in Development Teams: An Empirical Study

Master's thesis in Communication Technology
Supervisor: Maria Bartnes
Co-supervisor: Robert Larsen

June 2021

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Kunnskap for en bedre verden

Nora Bodin
Hanna Kai Barstad Golberg

Software Security Culture in Development Teams: An Empirical Study

Master's thesis in Communication Technology
Supervisor: Maria Bartnes
Co-supervisor: Robert Larsen
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Title:	Software Security Culture in Development Teams: An Empirical Study
Students:	Nora Bodin and Hanna Kai Barstad Golberg

Problem description:

The aim of this thesis is to gain insights in factors that make a development team motivated and equipped to develop software that satisfies basic software security requirements. Several guidelines and help tools attempt to contribute with knowledge on how to create more secure code. However, developers still produce software that contains well known security vulnerabilities, which again can be relatively easy exploited.

In companies concerned with software development, a considerable aspect of the security culture is the culture that regard developing, implementing and maintaining software. A common understanding of security culture is that it is about organisational security such as password policies and overall awareness of security threats. A more development-oriented view of security culture concerns how software security is prioritised in development teams, which is what this thesis will explore.

In this project the students will investigate security culture in development teams by performing an empirical study on current practices and challenges. The aim is to contribute to improved software security culture in development teams, which again will result in raised security quality in finished code.

Date approved:	12-02-2021
Supervisor:	Maria Bartnes, IIK
Co-supervisor:	Robert Larsen, Bekk

Abstract

This thesis explores what factors that influence security culture in agile software development. We define *software security culture* as the sum of the developers' knowledge, motivation, attitudes, and behaviours that affect how the development team develops adequately secure code. It also covers the development teams' use of tools and their routines and practices that affect the quality of the finished software.

This thesis presents an empirical study based on interviews with a total of 21 developers in 13 different consulting firms with offices in Norway. Our findings are related to individual, organisational and material factors influencing software security culture in development teams. These are discussed in light of earlier research.

Findings include an individual's interest, group dynamics, security roles and security training. We have seen that the individual's security interest contributes to the team's security efforts. Additionally, individuals contribute to the organisation's attitude and share security knowledge through competence days and security discussions. Psychological safety is important for performing within the team and at an organisational level. The security work varies depending on what product the team develops. A security role positively influences the security work on a team. Further, both consultancies and educational institutions lack adequate security training.

Based on our research, we derive some recommendations. Security should be a compulsory part of study programs that educate developers. Additionally, consultancies should provide introductory security courses to all new employees. More projects should initiate a security role with defined tasks, responsibilities and mandate.

Software security culture is an interconnected field. We have researched individual, organisational and material aspects from an information security perspective. The field of software security culture influence other fields of expertise such as strategy and management, social anthropology and organisational psychology.

Sammendrag

Denne oppgaven undersøker faktorer som påvirker sikkerhetskultur i smidig utvikling. Vi definerer *programvaresikkerhetskultur* som summen av utviklers kunnskap, motivasjon, holdninger og handlinger som påvirker hvordan et utviklingsteam lager tilstrekkelig sikker kode. Det inkluderer også bruk av verktøy, rutiner og praksis som påvirker sikkerheten i den ferdige koden.

Dette er en empirisk studie basert på intervjuer med totalt 21 utviklere ansatt i 13 ulike konsulentselskaper med kontorer i Norge. Funnene våre er knyttet til individuelle, organisatoriske og materielle faktorer som påvirker programvaresikkerhetskulturen i utviklerteam. Faktorene er diskutert i kontekst av tidligere forskning.

Funnene våre inkluderer individets interesse, gruppedynamikk, sikkerhetsroller og sikkerhetstrening. Vi har sett at individets sikkerhetsinteresse bidrar til teamets sikkerhetsarbeid. I tillegg bidrar individer til organisasjonens holdninger og til å spre kunnskap om sikkerhet via fagdager og diskusjoner. Psykologisk trygghet er viktig for å prestere i team og på et organisatorisk nivå. Sikkerhetsarbeidet varierer ut fra hvilket produkt teamet utvikler. En sikkerhetsrolle har positiv innflytelse på sikkerhetsarbeidet i et team. Både konsulentselskaper og utdanningsinstitusjoner mangler tilstrekkelig sikkerhetsopplæring.

Basert på disse funnene har vi kommet opp med noen anbefalinger. Sikkerhet bør inn som en større del i studieprogrammer der man utdanner utviklere. Videre bør konsulentselskaper tilby et introduksjonskurs til alle nyansatte for å øke bevissthet og kunnskap. Flere prosjekter bør introdusere en sikkerhetsrolle med definerte oppgaver.

Programvaresikkerhetskultur er et komplekst tema. Vi har undersøkt individuelle, organisatoriske og materielle faktorer fra et informasjonssikkerhetsståsted. Videre påvirkes programvaresikkerhet av andre fagfelt som for eksempel strategi, ledelse, sosialantropologi og organisasjonspsykologi.

Preface

This master thesis is the final submission of the five-year Master of Science in Communication Technology at Norwegian University of Science and Technology (NTNU). The research is done from January to June 2021, continuing the previous semester's pre-project in autumn 2020. We have rewritten and used some parts of this pre-project.

We would like to extend our thanks to our supervisors, Maria Bartnes and Robert Larsen, for guidance and following up our questions. Your knowledge, ideas and encouragements have been highly appreciated. In addition, we are grateful for valuable guidance, feedback and discussions provided by Ole Smørdal. We also thank Inger Anne Tøndel for discussions and input on this field of research.

We want to thank all the interviewees for sharing thoughts, knowledge and experiences from their everyday lives as developers in consulting firms. Thank you for your interest, involvement and engagement. We hope that our study contributes to improving practices on software security culture.

Thank you to our dear families for being available through ups and downs. We appreciate the love, inspiration and support you provide.

Lastly, we feel incredibly grateful for all the experiences we have encountered. Trondheim, you have given us everything. We have found close friends and developed our potential. Thank you for the discussions, support and adventures over the last five years.

Nora Bodin and Hanna Kai Barstad Golberg
Trondheim, June 2021

Contents

List of Figures

List of Tables

List of Acronyms

BSIMM	Building Security In Maturity Model.
CSV	Comma-Separated Values.
CTF	Capture the Flag.
DDOS	Distributed Denial of Service.
EU	European Union.
GDPR	General Data Protection Regulation.
HTTP	Hypertext Transfer Protocol.
ICT	Information and Communications Technology.
IDE	Integrated Development Environment.
Microsoft SDL	Microsoft Security Development Lifecycle.
NDPA	The Norwegian Data Protection Authority (Datatilsynet).
NIFU	Nordisk Institutt for studier av innovasjon, Forskning og Utdanning.
NSD	Norsk senter for forskningsdata.
NTNU	Norwegian University of Science and Technology.
OWASP	Open Web Application Security Project.
OWASP ASVS	OWASP Application Security Verification Standard.
SAMM	Software Assurance Maturity Model.
SAST	Static Application Security Testing.

SDL	Secure Development Lifecycle.
SDLC	Software Development Lifecycle.
UiB	University of Bergen.
UiO	University of Oslo.
XP	Extreme Programming.

Chapter 1 Introduction

In all companies concerned with software development, there is a security culture regarding developing, implementing and maintaining the software. Cultural practices vary from company to company. New technology and software evolve rapidly to meet emergent challenges in all industries. Nowadays, it is close to inevitable to live without leaving digital traces in numerous places, and software services are becoming an increasingly important part of our lives.

Malicious actors would like to exploit vulnerabilities and valuable information such as personal data and secrets regarding research and trading. They keep getting better at finding ways to compromise software, and some actors have unlimited funds to do so. Therefore, it is only becoming more critical to take security measures seriously and avoid security breaches. This calls for a raised concern for vulnerabilities in software and constructive security culture in development environments.

Time to market can be a make or break point for new products [1]. Subsequently, time pressure to reach hard deadlines can lead to quick fixes and shortcuts in software security work. This may be perceived as a risk worth taking until the product possibly gets compromised [2]. With so many software services trying to reach and obtain a critical mass to survive, the competition is tough.

Traditionally, Information and Communications Technology (ICT) security has primarily been about implementing security mechanisms on the system or network level. In recent years, it has become clear that it is equally important to ensure that all software mechanisms are secure, including the code itself [3]. The term *security culture* is commonly used to describe organisational information security culture regarding password management and awareness of general security threats such as phishing and malware [4].

This thesis will address *software security culture.* We define software security culture as the sum of the developers' knowledge, motivation, attitudes and behaviours that affect how the development team develops adequately secure code. It also covers the development teams' use of tools and their routines and practices that affect the quality of the finished software. *Adequately secure code* is in this context assessing vulnerabilities and mitigating risks to an acceptable level.

Today, the majority of development teams strive to work agile and efficiently. The values and principles from the Agile Manifesto from 2001 convey an iterative way of product development [5]. Frameworks built on the agile way of thinking are widely used; some examples are Scrum and Kanban. A concern is that agile methods tend to neglect security [6].

We see an increased focus on software security among developers [7]. One of the reasons for this focus is General Data Protection Regulation (GDPR), which went into effect by the European Union (EU) in 2018 [8]. However, we can read daily about companies that are compromised and personal data that are falling into the wrong hands [9, 10]. Publicly known security incidents may damage any company's reputation. OWASP Top Ten is a list with the ten most common web application vulnerabilities [11]. Accordingly, these vulnerabilities often become the exploit vector used by attackers. Cyber attacks can have significant consequences for both the confidentiality, integrity and availability of data [12].

The measures needed to protect information could change during the information's life span. We can illustrate this with an exam sheet: Before the examination date, there are strict requirements for confidentiality. No student should know the tasks in advance. During the exam, the exam must be available for the students the whole time exam period. This calls for requirements for availability. When the exam is over, measures for integrity is necessary for the answer given by the students. After the answer is delivered, no one should be able to alter it.

Cyber attacks targeting businesses are often financially motivated. However, motives can also be, to prove own abilities, a political point or espionage to get an unfair advantage [13]. The consequences of cyber attacks have a varying degree of severity. Following are some examples where attackers exploited vulnerabilities in software with varying consequences.

In the summer 2018, a twelve year old student hacked the municipality of Bergen. He wrote a script to search for information linked to his own username, and found an unencrypted Comma-Separated Values (CSV)-file containing 35,000 passwords belonging to students and employees associated with the school system. He notified the municipality about the vulnerability, but was then reported to the police [14].

Earlier in 2018, hackers compromised the app MyFitnessPal[1] and got access to

[1]MyFitnessPal: App and website for tracking diet and exercise. https://www.myfitnesspal.com/

data belonging to 150 million users. The attackers got hold of usernames, passwords and email addresses, which they later sold on the dark web[2] Although there are no precise details on how the app got compromised, experts speculate that it was a vulnerability in the server running the app [15].

In 2017, the ransomware virus Wannacry infected more than 400,000 machines. The attackers exploited vulnerabilities in Microsoft Windows, and users who had not updated their software with Microsoft's March software fix became an easy target. Machines in at least 150 countries were compromised. In total, the attack cost approximately 4 billion US dollars, equivalent to 33 billion NOK [16].

Although we see an increasing focus on security work in software development environments, the quality of the security work depends largely on the individuals' interest in the field and the culture of the companies they are involved with [17].

Throughout this thesis, we use the term "developers" for software developers who are writing code. We do not include designers, testers, analysts and managers in that term.

1.1 Research Questions

In this thesis, we will identify and explore factors that constitute a software security culture. We decided on three research questions to investigate the topic from different perspectives. RQ1 focuses on the individual developer's perspectives on software security. The aim of the research question is to better understand individual-oriented factors contributing to creating secure code. RQ2 regards organisational factors contributing to software security culture. In particular, we include team culture, security training and activities, customer-relations and others. RQ3 regards material factors that are part of the software security culture. Materials include tools, guidelines, and other services. Via these three research questions we aim to look at driving forces for software security and how they are influenced.

To sum up, the three research questions are:

RQ1: What factors influence the software security culture on an individual level?

RQ2: What organisational factors influence the software security culture?

RQ3: What material factors influence the software security culture?

[2]dark web: The part of the World Wide Web that is only accessible by means of special software, allowing users and website operators to remain anonymous or untraceable, definition found in Oxford English Dictionary.

1.2 Contributions

This thesis explores software security culture among software developers in consulting firms with offices in Norway. Through an empirical study we will explore security cultures and gain insights in current software security practices in the consultancy business. Research has been done on agile security work and security culture [1, 18, 19, 7, 20, 2, 21, 17, 6]. During our research we have not seen research on software security culture in Norwegian consultants. Our aim is to contribute to improved security quality in the finished software by addressing factors influencing the software security culture.

Earlier research done in the field has explored security in an agile development environment [1, 22], how security tools are adopted [7] and the effect of security consultants [23, 6]. We view this topic from an information security aspect, although it can be viewed from other fields of expertise, including social anthropology, organisational psychology and business management. We will not consider these aspects in debt. Customer aspects are discussed, however we do not interview representatives from the customer side.

1.3 Outline

Chapter 2 reviews earlier research and identifies relevant concepts for this study. Topics include software security in agile software development, team culture, practices and attitudes towards security work.

Chapter 3 discusses the qualitative approach of this study and presents the selection of developers who participated.

Chapter 4 thematically describe findings from the interviews, including quotes for illustration.

Chapter 5 discusses the research questions using findings and interpretations of this study in light of earlier research. Recommendations for improved security practice are given.

Chapter 6 concludes this study and presents possible areas of future research that will evolve this field of knowledge.

Appendix A provides the information sheet and letter of consent the participants got in advance of our data collection.

Appendix B contains the interview guide used in this study.

Appendix C includes the follow-up questions sent to the participants after conducting all main interviews.

Chapter 2 Background

In this chapter we will review the concepts and related work used in this thesis. We have identified the following concepts as key to understand when considering security culture: agile software development, team culture, organisational culture and guidelines. Earlier research has explored software security attitudes and tendencies in organisations who develop, implement and maintain software. They have also explored challenges and suggestions regarding how development teams, individuals and organisations approach their security work.

2.1 Agile Software Development

Agile software development is rooted in the Agile principles described in the Agile Manifesto from 2001 [5]. The values and principles convey an iterative approach to product development with a close dialogue between customers and developers. Agile frameworks, which are built on Agile principles, have become the common practice of development projects. Examples include Scrum, Kanban and Extreme Programming (XP) [24, 25]. Paying attention to technical excellence and good design strengthen an agile process [5]. Such a process may conflict with security practices and security requirements as it lacks procedures for security requirements [26, 24]. In addition, security work reduces efficiency by delaying pushing new features [22]. Earlier research suggest there is a suspicion that security work is being neglected in the agile process [6, 24]. However, Bartsch suggests that the holistic view an agile methodology provides can increase the developers feeling of responsibility of security work [17].

About two decades ago, the norm was to use the waterfall model where project activities are divided into rigid phases with little flexibility [2]. In this model, security had its explicit phase for implementation. Nicolaysen et al. write that this did not signify that the security necessarily was better, only that there was time set aside for it. There still had to be knowledgeable people and dedication to develop secure software [2].

In the study by Nicolaysen et al., only one of six companies had tried to combine software security with agile software development. Security activities may need to be granted specific permissions, and considerable documentation must be developed in advance of each project. The flexibility that an agile process provide may be hindered by security work that has to be structured and organised beforehand. Examples on this include strict policies and regulations enforced by The Norwegian Data Protection Authority (Datatilsynet) (NDPA) [2].

A consultancy or consulting firm is defined as “a company that advises on a particular subject”[1]. Most development teams in consultancies in Norway would say they work agile in some way [2]. To better understand how the developers interviewed in this thesis work in practice, we use the team roles of 'product owner' and 'development team members' from the Scrum framework[2] to give an impression of the structure. The product owner often works for the customer company, but she can also be an employee at the consultancy. The role is responsible for maximising the value of the product by having an overview of the product [27]. The product owner needs to understand both the customer's business perspective, as well as the product value of what the development team delivers. This way she can prioritise work tasks accordingly [28]. The development team creates the product in iterative sprints and is self-organised to get assigned tasks done. It may consist of programmers, designers and testers [28].

This thesis will regard information technology consulting firms that lend teams or individuals' technological competence to other companies. Consultancies have various delivery models describing what team structures and assignments they offer to customers, as well as pricing and what time constraints they operate with [29].

2.2 The DevOps Paradigm

The DevOps, Development and Operations, paradigm has been increasingly popular to development teams [30]. The practice promotes closer collaboration between software development and software operations teams [31]. The developers monitor and manage their software autonomously, which consequently adds more responsibility to the team [30]. However, it is easier to keep an overview of the system and fixing issues because of closer integration between development and operations [32]. Furthermore, it can motivate to take care of the software, rather than just being a piece in the puzzle. This provides a feeling of ownership [33]. An essential part of agile development is the balance between autonomy and direction. In a self-organising

[1]Definition: https://dictionary.cambridge.org/dictionary
[2]Scrum: https://www.scrum.org/

team, developers intensely disliked interventions from managers higher up in the organisation to encourage security activities within the team [6]. Interventions from the managers would also reduce the autonomy of the team. The ideal balance will let the teams reach organisational goals while avoiding micro-management and arbitrary interference [6].

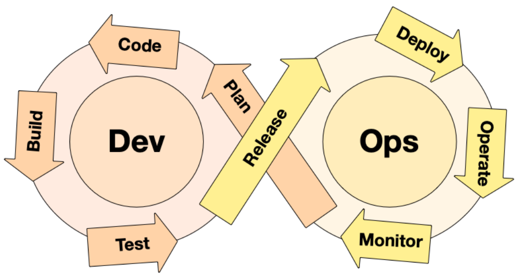

Figure 2.1: The DevOps process. Figure inspired by E-spin[3].

One of the challenges with DevOps is ensuring security while keeping its agility [31]. There are many challenges to the adoption of DevOps and simultaneously assuring security. DevSecOps, Development, Security and Operations, integrate security in the DevOps process [33]. The core principle of DevSecOps is to be an extension to DevOps to incorporate security in all phases of development and system operations. Security experts should be involved from the beginning [33].

2.3 Team Culture in Software Development

Agile development relies on people and their creativity rather than on descriptions [34]. Xiao et al. define security culture as the relevant social norms and habits that surround security in a team and an organisation [7]. The culture of a group is affecting how the team perform their tasks [35]. Schein defines culture as "that which is prominent and clear with a strong influence on the direction" [36].

The culture of the organisation is essential for efficiency and psychological safety [37]. Psychological safety is defined by as being able to show and employ one's self without fear of negative consequences of self-image, status or career [38]. In later years with more agile methods of working and autonomous teams, the team culture and subculture plays a more significant part in an employee's everyday life [39]. Psychological safety is also crucial for how the team communicates security concerns [17]. Research done by Bartsch concludes that motivation for security knowledge comes from a feeling of responsibility to the product. Further, they claim that this

[3]Figure inspiration: https://www.e-spincorp.com/common-mistakes-organizations-make-when-adopting-devops/

motivation is increased if the communication within the team is good [17]. Research done in 2019 concluded that teams who have a long-term plan have more focus on quality [40]. Further, they concluded that well-functioning teams focus on reflection.

Stray et al. has identified patterns that successful, autonomous teams in large, complex companies follow. Leadership can be a restrain of well-working teams because of constant interruptions, and disturbance of workflow [41]. Cross-functional teams encourage direct communication without proxies and decrease the probability of miscommunication and response time [42, 40]. A consequence of disruptions and complex communication chains will be less effective working hours. A developer spends on average twenty minutes to find focus after an interruption [43].

The paper by Wen et al. identified six dimensions of security culture. They addressed attitude, behaviour, competency, subjective norms, governance and communication to assess the security culture [18]. Most relevant for our thesis are attitude, behaviour, competency and communication. According to them, *attitude* affects how motivated the developers are to prioritise security work. Developers who do not believe that security is essential are unlikely to work securely, regardless of how much they know about security requirements [18]. *Behaviour* is what individuals do and relate to activities [18]. *Competency* is the collection of knowledge and skills which influences how well individuals or teams manage to meet demands [44]. *Communication* can be considered as an interactive process of sending and receiving messages among individuals, groups and organisations [18].

As mentioned in the introduction, we have defined *software security culture*. Inspired by Wen et al.'s dimensions of security culture, we defined software security culture as the sum of the developers' knowledge, motivation, attitudes and behaviours that affect how the development team develops adequately secure code. It also covers the development teams' use of tools, routines, and practices that affect the quality of the finished software.

In agile processes, there are different ways of sharing knowledge. Ways to share knowledge in teams could be pair programming, daily scrum meetings, and project retrospectives [42]. Bartsch observed that security knowledge spreads between developers through informal discussions [17].

We assume that development teams work with security in an infinite number of ways. According to Tøndel and Jaatun, the number of suggested security activities can be experienced as quite overwhelming. They state that projects can spend many resources on security, even overspending, if not correctly addressing the security needs [25].

Code reviews are technical activities done to decrease vulnerabilities in code [6].

The review practice may vary across the company, and be done by other developers for a peer review or external auditors [19]. An example of a technical code review is pull requests. When a developer wants to merge her code into the main code base, she can create a pull request. Another developer will do a peer review of the code to give input on concerns and discuss improvements before integrating the new code [45]. Code reviews by external security consultants are discussed in section 2.5.5.

2.4 The Individual's Perspective on Software Security

In this thesis, we will look at the individuals perspective on their interest, awareness and responsibility, among other aspects. Jaatun et al. concluded that the individual developer's interest, competence and initiative are significant for many organisations. They pointed out that it is crucial when it comes to keeping up to date on software security and ensuring that security is not forgotten in the development life cycle [3].

According to Bartsch, security competence is strongly dependent on the individual developer's interest in security. Many developers' security competence is built informally, often self-taught from security blogs and news articles [17].

Research done by Nicolaysen et al. and Bartsch conclude that expertise and knowledge influence software security in projects. An individual's software security competence, or lack thereof, impacts the software security [2, 17].

The individuals' security expertise, either on the customer or developer side, make up the overall security in a project [17]. Research from 2011 on security awareness among developers express that many of their interviewees could not explain concrete security practices, despite detailed knowledge of security issues [19].

In 2011, Xie et al. found an "it is not my responsibility"- attitude regarding responsibility for security in code. Further, they found out that their developers strongly trusted other people, processes and technology to take care of software security [19]. The findings of Xiao et al. from 2014 suggest a different attitude. They found that the majority of the developers believed that security is a shared responsibility among the developers. Teams who are supported by security and testing teams felt personally responsible for the security work in their software [7]. A third finding is from the paper of Poller et al. in 2017, where they found that developers feel responsible for keeping their software secure. The developers expressed that a part of delivering good quality included secure software features [6, 20].

2.5 Organisational and Environmental Influences

Reasonable approaches to promote and support security work in organisations are hard to find [6]. Security work is hard to manage because its benefits have delayed effect [7]. In addition, security is only one of many goals for an organisation [6].

2.5.1 Security Training Initiated by Consulting firms

Earlier research done by Xiao et al. has found little or no formal training from the organisations to learn security tools. They found that companies in their recruitment process asked questions on security before employment of new developers. Thus, they viewed it as not necessary to arrange security training for the company [7]. The attitude was that they already had checked that the developers had a basic level of security knowledge. Software security seems to be a tiny part of efforts to increase knowledge and awareness of information security in the various organisations [3].

2.5.2 Security Training in Educational Institutions

In 2017 a report about the future supply and demand of Information and Communications Technology (ICT) competence was released by Nordisk Institutt for studier av innovasjon, Forskning og Utdanning (NIFU). It concludes that the need for security courses in engineering educations is considerable. They present a need to raise the number of candidates in universities with ICT competence. They express the need for both generalists and specialists. By that, they mean candidates with general ICT security competence and specialists with specialised education in ICT. They emphasise the importance of including required security courses in all ICT study programmes. Several also want compulsory courses on IT security to be a part of all technical study programmes [8].

Today many study programmes that educate developers offer electable courses on security, but does not include security courses as an obligatory part. Examples are Computer Science [4] and Informatics [5]

2.5.3 Prioritisation of Requirements

Earlier research has concluded that functional requirements get prioritised over software security [22]. Another empirical study found that functional requirements are viewed as more critical than non-functional requirements [2]. Security is observed to be conceived only as a non-functional requirement, not a visual business goal of the software [6]. However, neither a feature nor a non-functional and quality perspective

[4]Study plan Computer Science: https://www.ntnu.no/studier/mtdt/oppbygning

[5]Study plan Informatics: https://www.ntnu.no/studier/bit/oppbygning.

adequately addresses security requirements. For this reason, incorporating security in any development process is challenging [6].

Customers influence how the development team works [1, 6, 17, 26]. Security requirements are often not included as part of the request from the requester, product owner, or the customer [1]. A customer may fail to communicate its security needs due to lack of knowledge or awareness [26, 1]. The non-technical customers often struggle to comprehend technical security measures, and in many cases, the customer trusts the developers to handle it [17]. In scenarios where the product owners or customers do not explicitly address security, it is expected to be fixed by the development team [6]. Thus, developers make assumptions about how security should be prioritised and determine the definition of adequately secure code for the system [6]. Furthermore, this can lead to security work being less prioritised or not included in the backlog [2].

Customers' awareness of security, as for developers', depends on the specific individual you talk to [17]. However, customers and product owners contribute to the security work with their knowledge from specific industries, even if their security awareness is low [17, 1].

A factor that plays a role in paying attention to software security is whether or not the team has to meet legal requirements. Legal requirements which need to be met is a crucial driver for performing risk analysis. Risk analysis to uncover the need for software security is rather driven by compliance than a general concern for vulnerabilities [22].

2.5.4 Trusted Third-Parties

When developers write code today, 85 per cent of their code is based on reusing existing code [46]. This way, the developers do not have to reinvent the wheel whenever they want a new feature. Thomas et al. found that open-sourced frameworks and libraries often have vulnerabilities. They derived that vulnerabilities are difficult to control as fixing them is dependant on the library vendors. In addition, they can contribute to false-positive errors in static code analyses because the source code is not available [47]. Books get quickly outdated and do not provide customised code snippets for all use-cases [20]. Developers use Internet resources, such as Stack Overflow[6]. This often leads to functionally correct results, but the answers lacks focus on a secure solution[48]. Xie et al. found that code security was never a criterion when choosing what reused code to choose [19].

[6]Stack Overflow: https://stackoverflow.com/

2.5.5 Separated Divisions and Security Consultants

It is unclear who is accountable for security actions [1]. Some development teams include independent security teams or hire external security consultants to review their code [2, 6].

After an external security consultancy in development teams, the awareness and motivation for security concerns increases, Poller et al. and Turpe et al. found. Nevertheless, they both observed security work did not get incorporated into the daily processes. Thus, in the long run it did not attract more engagement for security work [6, 23]. Security and legal competence in organisations do not necessarily benefit development [22]. It is difficult to make compromises as architects and security experts have different priorities, and it feels like they are chasing different goals [1].

2.5.6 Automatic Tools and Scans

Earlier research done by Xie et al. from 2011 concluded that interactive tools would effectively implement more secure software. They claim that developers need greater awareness of specific errors in the context of their development. Tools that detect and flag such code during program construction, not after code completion, may help alert them to fix the problem areas [19].

Today various automatic tools exist, some are open-sourced, and others are commercial. Some tools scan and identify known vulnerabilities for giving the developer a heads up before continuing. Others both identify vulnerabilities and automatically fix them. Some examples are Snyk[7], Detectify[8], OWASP Dependency check[9] and Retire.js[10]. Another type of automatic tools is static code analysis. These tools flag programmatic errors, bugs and suspicious constructions. Examples are ESLint[11] and Static Application Security Testing (SAST)[12].

2.5.7 Security Guidelines and Checklists

This master thesis addresses security guidelines and how they are used in practice. This section is strongly influenced by the pre-project to this thesis, the delivery in the course TTM4502[13] at Norwegian University of Science and Technology (NTNU) [49].

[7]Snyk: https://snyk.io/
[8]Detectify: https://detectify.com/
[9]OWASP Dependency check: https://owasp.org/www-project-dependency-check/
[10]Retire.js: https://retirejs.github.io/retire.js/
[11]ESLint: https://eslint.org/
[12]SAST: https://owasp.org/www-community/Source$_{C}ode_{A}nalysis_{T}ools$
[13]Pre-project: https://www.ntnu.edu/studies/courses/TTM4502

Several frameworks and guidelines are developed to ensure secure code development and software [25]. Their goal is to raise awareness and create a process for addressing software security. There are two types of guidelines, prescriptive and descriptive. The prescriptive models explain what to do to obtain an adequate level of security, whereas a descriptive model describes a system as it is now [3]. Both descriptive and descriptive models can view many security-related aspects of the development process.

Building Security In Maturity Model (BSIMM) is a descriptive tool, mostly used to measure security practices in different companies. The model aims not to evaluate whether a system is good or bad, but rather to observe its characteristics within four domains, each with three categories or practices. According to Jaatun et al., the BSIMM model is a reflection on security [50]. This model is based on research done on large American companies such as Cisco and Aetna [51]. A strategy described in BSIMM is the use of satellites of interested individuals. A collection of people across the organisation who show interest or skill in security could be collected into a satellite. This is a measure taken to scale security by creating social networks that contributing to adoption of security into development. The individuals could be picked out by finding individuals that stand out, or by asking for volunteers [52].

Seven Touchpoints for Software Security is a less comprehensive guideline, based on the BSIMM model. The book contains both destructive and constructive activities to adapt to the security work. Examples of destructive activities are exploits and attacks, and examples of constructive activities are design, defence and functionality. The seven parts presented are code review, architectural risk analysis, penetration testing, risk-based security tests, abuse cases, security requirements and security operations [53].

The global community OWASP created the model *Software Assurance Maturity Model (SAMM)* in 2009 [54]. It is based on the four areas: Governance, Construction, Verification and Deployment. The point is to assess the maturity level in each area and explain what could be done to obtain more secure software. It is an open framework to help teams and developers analyse and implement a security strategy that fits organisations' already existing work [3]. In addition, the OWASP community has created a list of the ten most critical security risks to web applications. On their website, they express that using the OWASP Top 10 may be the most effective first step towards changing the software security culture [11].

OWASP Application Security Verification Standard (OWASP ASVS) is a standard containing a list of application security requirements and tests that are designed to be used by architects, developers, testers, security professionals, tool vendors and consumers to define, build, test and verify secure applications [55]. It is built to fit applications that require three different levels of security. Each level increases the severity for a secure application [55].

The prescriptive *Microsoft SDL* model has focused on security and privacy considerations throughout all the phases of the development. It focuses on software developers and their work to gain secure software and how to reduce development costs [56]. However, a study done by Baca et al. concludes that this model scales poorly to a chosen agile security process because of too high costs and because the model was not beneficial enough for the agile conditions. Their preliminary findings from interviews showed that the design phase and the testing phase scaled especially poorly [57]. Later a new variant of Microsoft SDL was developed that aimed to be a better fit to agile processes [25].

The Norwegian Data Protection Authority (Datatilsynet) (NDPA) has made a guideline named "Privacy by Design". It is about how to incorporate data protection principles, subject rights and the requirements of GDPR into every step of the development process, and it is developed with inspiration from the Microsoft SDL and Secure SDLC [58].

There exists a significant amount of various checklists regarding software security. We find it on blogs, commercial sites such as Synopsis.com and official sites such as The Norwegian Data Protection Authority (Datatilsynet) (NDPA) [59, 60, 61, 58, 62]. The checklists from NDPA focus on how to develop code with mechanisms for build-in privacy [58, 62]. In addition, for building secure and robust infrastructure in cloud-computing systems, Amazon's AWS Well-Architected Framework can be used [63].

Issues Regarding Use of Security Guidelines

As presented, there are various guidelines describing methods for doing software security work in an agile environment [25]. Still, earlier research has noticed that many projects today are not giving security requirements enough attention [25, 22, 26].

Guidelines such as Microsoft SDL and Seven Touchpoints for Software Security fall short on how to get organisations and the actual development teams to start using them, Poller et al. argue. Security initiatives need to take organisational factors into account. By that, they mean that security initiatives should not only be considered from a security engineering point of view, but also from a management perspective with the organisational setting in mind [6]. This supports the findings of security being perceived to be in conflict with the trending "continuous development" methodology [64, 22, 24, 26].

Chapter 3

Method

In this thesis, we combine an empirical study based on qualitative interviews and a literature review on topics related to software security culture in development teams.

Empirical studies collect knowledge through direct or indirect observations, and experiences of a field [65]. We argue that an empirical study combined with a literature review in the field is the best way of answering our research questions regarding culture, further discussed in section 3.2.

3.1 Overview of the Process

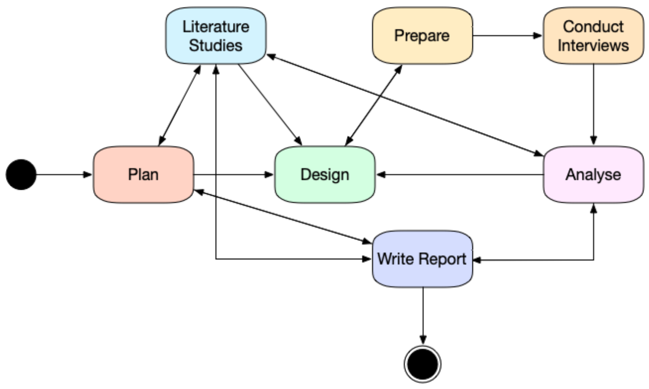

Figure 3.1: The process presented visually.

Figure 3.1 is a visual representation of our process. In practice, we have been in more than one state at a time and frequently switched between them. Still, it roughly illustrates how we have worked in different phases of this study.

After the initial state, represented by the circle filled with black, the first phase is the *Plan* phase. This phase included setting the research questions, deciding on what method to use, planning how to get in touch with relevant participants, and drafted a progress plan. When initial planning was done, we could start the literature study, the design phase, or write the report. From all these phases, we could go back to planning when new reflections occurred.

The next phase, *Literature Study*, includes searching and finding relevant literature using Google Scholar and Oria. We also received some articles from colleagues.

In the *Design* phase, we designed the interview guide we used when conducting interviews. We used insights from earlier research and our decided research questions to come up with relevant questions.

The *Prepare* phase includes preparations before interviews. We read up on the interviewee's careers to know more about them in advance. Based on this research, we could customise or add a couple of questions to the interview guide before their interview. From the prepare phase, we moved to *Conducting Interviews* phase.

In the *Analyse* phase, we used categories and tags to understand better what we had found out. We could go back to the design phase to change the interview guide to embody our research questions better. We could also go back to literature studies to put our findings in the context of earlier research.

The last phase, *Write report*, could be reached from the analyse, planning, and literature study phases. This phase includes writing the final report. We will reach the final state represented by the black circle with a double outline when the thesis is delivered.

3.2 Qualitative Semi-Structured Interviews

A qualitative analysis approach focuses on words and content used by participants rather than numbers and statistics [66]. However, quantitative techniques are good for statistical analyses, enabling to point out the average score on a variable, range of scores and strength of relations between variables [66].

A disadvantage of quantitative approaches is that questions may be misrepresented and oversimplified by the participants [65]. As the participants are self-reporting their answers, their interpretations of each question may differ. Furthermore, it can be difficult for the participants to communicate perspectives that the researcher did not anticipate.

When deciding which data collecting method to use, we considered that we were to research culture in different firms. We chose to use qualitative, semi-structured interviews as the primary data collecting method because culture is a composite topic. Individuals do not have standardised behaviour and reactions, and different firms follow different practices. Semi-structured qualitative interviews embody the aim of attempting to capture unique incidents and views of an individual. We wanted to gather a broad spectre of insights without needing to follow a strict plan. Semi-structured interviews gave us the flexibility and advantage to dive further into a topic if the interviewee mentioned relevant information we did not cover directly with our interview guide.

In the early stages of this thesis, we looked at the possibility to use a survey for collecting quantifiable data. Surveys are usually conducted with samples from a large population [66]. Due to surveys following a strict format with little flexibility, we found it challenging to create standardised questions for a survey on the topic of security culture [65]. In case they are not followed up with, for instance, a conversation, interpreting the results is time-consuming and a slow process. We thought of using open-ended questions to gather various insights. However, we decided not to use surveys because of the extensiveness in analysing them, and they would miss non-verbal communication. Such communication includes body language, facial expressions and pitch in speech, which is crucial to get a complete understanding of the expressions of the individuals. Moreover, the advantage of surveys and quantitative analyses is gathering standardised data. Consequently, we believe that surveys do not fit researching software security culture.

In addition to our qualitative semi-structured interviews, we also wanted to assess the software security interest among the participants in a numeric value. Thus, in the follow-up we included one question on a summated rating or Likert scale from 1 to 5. Score 1 indicate “Very uninterested” and 5 signify “Very interested”. We did not explicitly state the scores in between, but approximated them to the following. Score 2 to “Somewhat uninterested”, score 3 signify “Neither interested nor uninterested” and score 4 means “Somewhat interested”.

3.3 Literature Review

To find relevant literature and earlier research, we used Google Scholar and Oria. Additionally, our supervisors recommended articles of various relevance. Topics in the search query included “software security culture”, “agile software security” combined with words such as “security role”, “developer”, “training”, “consultancies”, “activities”, “factors”, “tools”, “team”, “silo” and “customer”.

When we successfully found relevant research, we used the references of these articles and papers to further explore the topics.

3.4 Selection of Participants

The participants in this study are full-time software developers with various backgrounds. Many of them have their technical education from Norwegian University of Science and Technology (NTNU), University of Oslo (UiO) and University of Bergen (UiB). The participants were found via our networks. They all work as developers in consulting firms with offices in Norway. Their firms provide developers as consultants to customers in a wide range of projects and industries. We have conducted interviews with consultants who are quite newly employed and with consultants who have experience from various projects through more than ten years. We have chosen to define a "junior developer" as a developer who has worked as a development consultant in less than three years. A "senior developer" has three or more years of experience. The communication with the participants has solely been in Norwegian. This includes the initial contact, the interview itself and follow-up correspondence.

The participants' consulting firms were of varying size. In total, we interviewed developers from 13 different consultancies. Small firms have less than 50 employees, and medium-sized firms are defined to have between 50 and 250 employees, while large firms have more than 250 employees [67]. The distribution could be seen in figure 3.2.

We used *snowball sampling* [65] by contacting people in our network. This means that the people we contacted either agreed to participate as interviewees and/or referred us to other relevant people from their network who may want to participate.

The participants we refer to as "having a security role" are also developers on a team, but have an additional security focus on the team. Three of the participants are responsible for the onboarding programs provided to new employees in their firm. Many developers mentioned competence groups for security in this thesis. Our main findings on competence groups are based on insights given by developers from the same competence group.

Total number of developers	21
Total of junior developers (experience < 3 years)	10
Total of senior developers (experience >= 3 years)	11
The most experienced developer	> 15 years
The least experienced developer	8 months
Total number of unique firms	13
The smallest firm	< 30 employees
The largest firm	> 2000 employees

Table 3.1: Our selection in numbers.

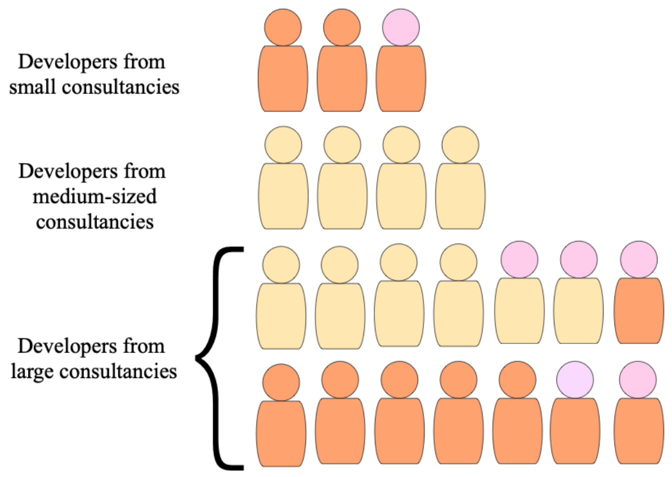
Developers from small consultancies
Developers from medium-sized consultancies
Developers from large consultancies

Figure 3.2: Distribution of the selection in this study. Yellow participants represent juniors and orange are seniors. The pink heads represent security roles. The participants are sorted by size of consultancy from small to large.

3.5 The Interviews

All the interviews were conducted over five weeks, from 05.02.2021 to 12.03.2021.

Before all interviews, we looked up information about the interviewee's career and interest fields to better utilise their specialised knowledge and insights. We found them on LinkedIn[1] and read news articles or blog posts if available.

The interview guide was created in advance of the interviews, see appendix B. We made the guide to plan the conversation and clarify what we would like to find out. The guide is in Norwegian, as all our interviews were conducted in Norwegian.

Before the interviews, we sent the participants the interview guide by e-mail. The guide allowed the participants to come up with relevant thoughts beforehand of the interview. If they did not find it relevant for their role, they could also reject the interview. Furthermore, we sent the participants an information sheet regarding privacy. The sheet can be found in appendix A. The sheet contained information about how we would store their data and how they could withdraw their contribution consent at any time. The sheet was inspired by a template made by Norsk senter for forskningsdata (NSD).

[1]LinkedIn: Social platform for professional networking, https://www.linkedin.com/

3.5.1 The Implementation

We started the interviews by presenting ourselves and the topic of the thesis. The purpose of this is to create a safe environment and make the interviewees feel comfortable [65]. Both parts of the interviews can have a more pleasant experience and possibly get a more comprehensive conversation if this is achieved. Then, we referred to the information paper on privacy and asked if they agreed to be audiotaped in the interview, which they all did. We continued by letting the interviewee know why we sought their insights and thoughts and transitioned the conversation over to them.

The interview was divided into five main parts. In the first part, we focused on the individual's consciousness of security in their everyday lives and their sense of responsibility for their code. Secondly, we focused on how security training and security activities are done in practice at their workplace and how they work to include new developers in the existing software security culture. In the third part, we examined team culture and activities practised to achieve the right level of security in the finished software. We also talked about factors that affect a team's software security culture. In the fourth part, the focus was on how the customer's culture affects the interviewee's security work. The customer is the actor who hires the competence of the consultancy. In the last part of the interview, we focused on security activities desired to be more accessible at their workplace and topics within software security they would like to learn more about.

As we used semi-structured interviews, we could be flexible when interviewees introduced new topics. When the interviewee had explained their view of an impromptu topic, the interview continued with questions from the interview guide. When a participant remarked a particularly relevant point, we added this to the interview guide before the following interview. Accordingly, the guide was updated during the interview period to embody our research questions better. We did not ask all the participants the same questions. If a question did not yield any new information, we let it out in the guide for the following interviews. The interviews felt more natural following up a conversation on exciting topics, rather than strictly sticking to an interview guide.

The guide was adapted to our interviewees in terms of their role and experience. Senior developers were asked additional questions on the topic of creating a software security culture. Developers in security and tech lead roles were asked to elaborate on these.

We conducted the interviews, with one of us being the primary questioner. The other one took notes of interesting parts as well as contributing with follow-up questions. Right after the interviews, we wrote a reflection note individually. We included immediate thoughts such as what surprised us and what impression we got.

3.5.2 Follow-up Questions

Approximately a month and a half after our last interview, we sent a follow-up e-mail to the participants. In this e-mail, we requested their view and experience on using guidelines, checklists and other resources. We also collected their self-reported scores from 1 to 5 of interest in software security. Also, we asked where they got their education from and whether it included security courses, obligatory or elective. Because of choosing semi-structured interviews, some of our respondents had already answered these questions, but we did not have this data standardised. The questions are available in appendix C.

We discovered during an interview that one of the interviewees was not a developer, but she still contributed to this topic. However, she was not further followed up with questions regarding a developer's view on guidelines and quantified interest.

We asked for a response either as a conversation or in a written response. Out of 20 interviewees, we had a 15-minute video meeting with 4 participants and got a written response from 13.

3.6 Thematic Code Analysis

We transcribed the highlights of the interviews by writing down the most valuable points and quotes that illustrated and described the topic. The name of the firm and details that could identify the interviewees were left out of the transcriptions.

First, we did the coding analysis physically. We decided on some main tags and hung up post-its representing each tag. Then, we printed out all the transcriptions and cut them into pieces based on topics. We matched the topics in the answers with the tags on the wall behind us. By doing this, we gathered different interviewees' views on each tag. After some time, we needed to be more systematic in our analysis. Therefore, we continued the analysis digitally.

First, we used the visualisation tool, OmniGraffle[2]. There we made a mind map consisting of the most important tags, quotes and findings. We did this to make our findings more visual. Later, we started using a tool called Obsidian [3], where it was possible to link notes. We imported the transcripts as text files. The tool lets the user tag the notes as well as link them to each other. The linking made us further understand the complexity in our findings and see contexts. We used 38 different tags, which can be found in the table 3.2. After that, we used the tags to define the

[2]OmniGraffle: https://www.omnigroup.com/omnigraffle/
[3]Obsidian: https://obsidian.md/

main themes, which can be found in table 3.3. In the later phases of the thesis, we used Obsidian to navigate back and forth in our transcriptions quickly. We used it, for instance, when checking up a quote or connecting the dots on specific topics. The tags allowed us to search for specific topics. An example is to search for the tag "#awareness" and get all sections where interviewees discussed security awareness.

responsibility	activitiesFirm	activitiesTeam	awareness
background	blackbox	experience	change
understanding	frontvsback	factor	guides
interest	individual	junior	customer
communication	corona	culture	learning
operational	leadership	diversity	maturity
environment	trainingFirm	traningCustomer	product
resources	securityConsultants	silo	securityRole
thirdParty	techLead	team	developmentProcess
challenge	wish		

Table 3.2: Tags found in the code analysis, translated from Norwegian to English.

The Individual Developer	Team	Organisational factors	Customer

Table 3.3: Main themes found in the code analysis.

Privacy In advance of starting this thesis, we sent an application to Norsk senter for forskningsdata (NSD) regarding data collection from interviews and storing personal data. The manner in which personal information is collected, stored and questions regarding privacy concerns, was determined in consultation with NSD. The audiotaped interviews and transcripts were stored in the cloud service Microsoft OneDrive which we got access to via NTNU.

3.7 Limitations

Our selection consists of primary developers in our network or people our network know. A consequence is that several of our interviewees have similar backgrounds. We have chosen to interview developers in consultancies and did not speak to anyone on the customer side. Nevertheless, we do present some findings regarding the customer-side. Our understanding of consultancy culture is from the 21 individuals of 13 different consultancies in Norway. As the period of this thesis is limited to 20 weeks, we delimited the scope by the number of interviews and what topics we consider in this study. Therefore, more research is needed on this topic before the results can be generalised.

The Likert scale is measured on an ordinal scale, which is a level of measurement that reports the ranking and ordering of the data without actually establishing the degree of variation between them [68]. The participants are self-reporting and therefore interpreting the possible choices. Due to individuals being reactive, their responses may be biased as a result of wanting to give a “decent” answer to the researchers [66].

3.7.1 Interview Method

Semi-structured interviews allow impromptu questions and conversation depending on the interviewee’s answers. Consequently, the interviews did not contain the exact same questions. We gathered insights from different topics depending on what the interviewees responded. We did not collect standardised data from all the participants. We think this is justified in the breadth of the answers collected. The interviews rely on self-reporting, in which we rely on the participants’ answers to be realistic and experience-based.

3.7.2 Interviews Over Video Link

This semester has been affected by the coronavirus pandemic in the world. That means that all interviews is conducted over a video link and not in person. Disadvantages of digital video interviews are related to the lack of physical presence. A physical interview is strongly affected by other factors than the particular things that are said. These factors include eye contact, body language, nuances in voice and the feeling of psychological safety. They are challenging to observe and control when conducting interviews over a video link. The natural small talk setting before and while getting seated for an interview is difficult to sustain over the video, which can influence the genuineness of the dialogue. Also, it is more challenging to understand each other’s reactions and body language during the conversation. Due to latency in the video connection, timing is tricky. We experienced consequences such as interruptions in the flow of the interview and difficulty to ask good follow-up questions.

Chapter 4 Results

This chapter presents the results of the qualitative interviews, including the follow-up questions. During analysis, we have observed common themes and challenges related to different parts of security culture. The structure of this chapter follows, to a large degree, the interview guide. The findings are sectioned into four parts: factors regarding the individual developer, team culture and security activities, organisational factors that affect the security work and the customer-related influences. Furthermore, we will present specific findings with examples and quotes to illustrate our findings and interpretation.

4.1 Findings Regarding the Individual Developer

Each individual's background, interests, and security training seem to affect how they contribute to team efforts regarding software security.

4.1.1 Personal Interest

The results of the interviewees' self-evaluation of their software security interest on a scale from 1 to 5 are presented in figure 4.1. We received 17 of 20 answers to the follow-up questions. As can be seen in the chart, almost half of the material (47%) assessed themselves as "Somewhat interested (4)" or "Very interested (5)" in software security, and arguably care about the field. Another 47.1% placed themselves as "Neither interested nor uninterested (3)", while one interviewee responded "Somewhat uninterested (2)". No one answered "Very uninterested (1)".

Some interviewees who answered "Neither interested nor uninterested (3)" elaborated that saying security is interesting if discussed or presented to them. However, they did not personally pursue security topics or resources. Whenever they attend conferences, they may seek out one track to stay updated. One interviewee expressed his lack of interest in software security openly. He had experience and knowledge of security from several projects. Continuing, he explained that although he under-

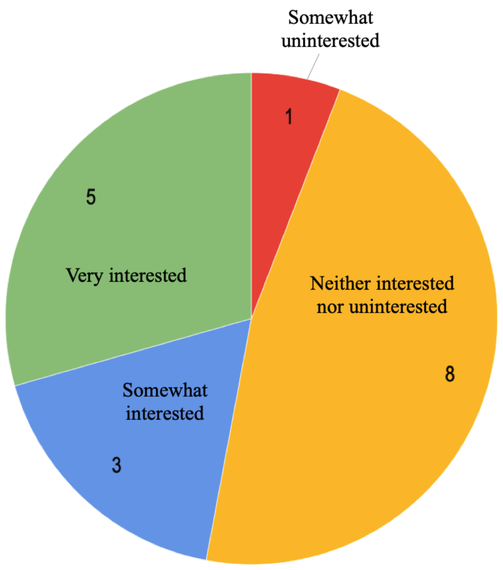

Figure 4.1: Pie chart of self-evaluated personal interest in software security.

stands the value and importance of security work, he finds it tedious and cumbersome. According to him, security is simply something that has to be done.

The majority of interviewees who answered “Very interested (5)” are individuals who have engaged in security activities in their spare time. Examples include reading security blogs, attending security conferences, being part of the Capture the Flag (CTF) community[1] and watching YouTube videos on security. We see that these individuals often initiated discussions and unstructured security activities within or across teams. Interviewees expressed that security-interested individuals contributed to keeping the whole team updated.

4.1.2 Awareness and Responsibility

Some senior developers reflected on the increased awareness of software security among developers. A senior developer observed that although not all developers express security interest, most of them will devote time to security measures. Security can be viewed as time-consuming, a junior developer remarked. He expressed that working with someone who cares a lot about software security can be demanding. He continued saying that focusing on functionality, software development is quicker, and the product will work, independent of security. Security tends to block practicality due to more strict data handling and data flows. Several point out that there is a

[1]Capture The Flag: A gamification of finding and exploiting vulnerabilities in software.

balance between practicality and security. Most developers in our material viewed security as a non-functional requirement, as opposed to a functional one. One senior developer reflected on the trade-off between the two types in quote 1. He conveyed that security is a vital part of the product and should not be overlooked.

> “The functional requirements must be present for us to launch, but if the non-functional ones are not, it means that the solution might as well not exist.” [2] (1)
>
> - *Senior developer*

Security work is often a priority when choosing architecture and design, an interviewee stated. Thus, developers may consider security work as finished in the programming and testing phases, she continued. Accordingly, security work is already recognised, and the developers think they do not need to consider it. She emphasised that the challenge is to get everyone to understand the importance of security in all development phases. The security pressure must be sustained in the later phases where security is not an obvious priority.

The importance of awareness in a development team was weighted in quote 2. The respondent with a security role expressed that a team needs awareness of security to find motivation and be responsible for security work.

> “Developers need to get the reflex of thinking ’Wait, did we think about security?’. With that reflex, they become afraid of doing something stupid and search for the resources themselves. I believe the most important thing is security awareness on the development team” (2)
>
> - *Senior developer, with a security role*

We asked the participants to which extent they thought of software security while programming. One of the answers is quoted below (3). His experience was that the majority of developers solely think of functionality when they code. He believed the customer only wanted him to focus on coding to meet functional requirements and not consider quality attributes and security. However, he admitted that he could not tell the value of focusing on security because he did not know much about it. Nevertheless, his view does not represent the majority of our selection. The majority said security is always in the back of their minds and part of their programming attitude. Most interviewees stated that although the team has security-interested people, security work should be on everybody’s minds.

[2]Quotes in this chapter are obtained from the interviews and translated from Norwegian to English.

> "I think one hundred per cent on functionality when I code. I do not think about security at all. I think this is true for the majority of developers" (3)
>
> - *Junior developer*

We observe that there is more assumed responsibility for the back-end developers than the front-enders. An interviewee summarised the perception in the quote below (4). Furthermore, we learnt that the purpose of the front-end often is to visualise the back-end. Thus, interviewees stated that the front-end does not carry independent data and does not need to care about security.

> "The back-end developers carry a greater responsibility. Anyone can send those HTTP-requests, and so this needs to be taken into account in the back-end." (4)
>
> - *Junior developer*

A senior developer reflected on a technical course in OWASP Top Ten vulnerabilities. Although he found the course interesting, he did not think it was relevant for him as a front-end developer. He suggested addressing another course focusing more on front-end security and figured that such a course would have been more relevant to him.

Nearly all of our interviewees claim that they share responsibility for writing secure code. An interviewee said that the responsibility is shared equally between developers, team leader, product owners and other stakeholders. Another interviewee said that every team member is responsible for striving towards a secure code. The whole team should participate in reviews of the code and discuss security topics whenever needed. A third interviewee emphasised that the team lead is responsible for giving necessary information to the developers. Our material mainly agrees that budget and prioritisation set by stakeholders along with code reviews and discussions on the team distribute the security responsibility.

One interviewee pointed out that she did not think one single individual could be held accountable for the product's security. The accountability for security should be split among the partakers of the product.

While most respondents had an idea of their duty in a shared security responsibility, one interviewee remarked that he had never given this a thought. After some contemplation, he concluded that the responsibility probably lies with the developers or the project leader. Regardless, he did not think that unfamiliarity had affected how he worked with security.

One respondent expressed that although the whole team shares responsibility to deliver secure code, he felt a particular, personal responsibility. Because of his security role, the team trusts his technical competence and knowledge to prevent common vulnerabilities in the product.

Shared responsibility for ensuring software security is a common expectation. Most interviewees feel that although the team has a security-interested individual, everyone should be aware of security. They mainly agree that the responsibility is shared through code reviews and discussions within the team and budgets and prioritisation from leadership and other stakeholders.

4.1.3 Knowledge and Experience

According to a senior developer in our material, it is not enough to be interested or aware of software security if the developer does not have a basic level of knowledge to make precise decisions. She emphasised that such knowledge does not come to a developer by chance. The developer needs to acquire security knowledge and experience through education or projects.

A junior developer expressed that she would like to learn more about software applications in general before diving into specific security measures. Another junior said that her experience at projects had made her more aware of the importance of software security. She elaborated that her education had not taught her the importance of security work. Generally, our interviewees expressed that they have learnt software security through project experience and other team members.

4.1.4 Educational Background

According to an interviewee, the study programme of Computer Science at NTNU mentioned security in some courses. He described the security part as forced and superficial. An interviewee with a background from NTNU stated that she wishes she had learnt more about security during education. Since her study programme, Computer Science, did not emphasise security courses during her studies, she did not view software security as necessary until she started working. Now she supports a change into mandatory security classes in her Master of Science degree.

Not all of the respondents have gained general development knowledge through their studies. Some studied more theoretical engineering and adapted the logical, analytical thinking aspect into software and web development. Despite this, they do not feel that they are less enlightened. However, their focus and awareness of security may be lower than developers who specialised in information security studies. Commonly from our material, the developers who completed security courses in their studies actively elected them.

4.1.5 Summary of Findings Regarding the Individual Developer

We see that almost half of our selection rate themselves as “Somewhat interested (4)” or “Very interested (5)” in software security from our material. Further, we see that the individuals’ interest influences the knowledge base the individual holds. In addition, we see that most developers believe they are partly responsible for the security in the code they are developing. Moreover, several developers said that their education and the lack of required security courses affect their security work when starting as consultants.

4.2 Findings Regarding Team Culture and Activities

Our material shows that the overall team culture and group dynamics influence the security work. Novice consultants in a team need to understand the culture, including written rules, norms and practices. According to an interviewee, it is easy to notice the culture through daily stand-ups by looking at the code and other activities. He continued saying that the developer then finds out if the culture feels comfortable to them or if they would have to learn something new to fit in.

4.2.1 Diversity and Team Membership

Several interviewees emphasised the importance of diversity in a team. Everyone cannot excel at security, and not everyone can be a champion of efficient back-end development. A variety of backgrounds, experiences, genders and interests are valuable.

From our material, it is clear that the team benefits from the experience and engagement of an interested individual. An individual with a security role said that he often brings up security-related topics and contributes as a discussion partner regarding security concerns. He expressed that other team members benefit from his knowledge. His view is illustrated with the quote below (5).

> “In practice, I believe that most teams are praised if they have someone who is greatly interested in security. In my last team, I interfered a lot, and the security work was improved. I think having that one person who cares matters.” (5)
>
> \- *Senior developer, with a security role*

A homogeneous environment will affect the team in a negative direction, a couple of respondents remarked. According to an interviewee, seniors usually have more experience and knowledge of how to avoid common vulnerabilities. An interviewee said that those who assemble teams should consider diversity. Otherwise, it is arbitrary if the team composition works well or not.

4.2.2 Psychological Safety

Multiple interviewees have talked about the importance of collaboration skills in the team. To promote this, an interviewee emphasised that psychological safety for all team members is desired. He said that constructive communication and feedback is vital for individuals to trust each other.

Several interviewees expressed that colleagues had become better at both giving and receiving constructive feedback. Some emphasised the importance of not blaming others for mistakes they have made, as shown in quote 6. An interviewee explained that a mistake never only has one cause. He further said that the importance of psychological safety within a team might be underestimated.

> "No one should experience feedback on code as criticism, no blame should be given, that will not help making us better." (6)
>
> - *Junior developer, with a security role*

4.2.3 Activities Within the Team

Teams arrange activities to improve the security work. It could be structured activities such as risk assessments and penetration tests and unstructured activities such as discussions during lunch and spontaneous initiatives from an individual. Activities are arranged to improve the team's psychological safety and improve the overall security work through more technical activities.

Our interviewees seem to use their peers to learn. Several interviewees highlighted the availability of their colleagues and their willingness to teach. An interviewee explained a practice where an experienced developer shared work assignments with the novice, such that the novice did not have explicit responsibility. He called the practice shadowing. The novice was assigned one specific senior to ask questions and gain knowledge from throughout the period. This activity was resource-intensive, he expressed, but the experiences the novice gained could make it worth it.

Technical lunches are arranged to improve the relationship between team members, an interviewee stated. Before each technical lunch, a team member prepared a lightning talk, possibly on the topic of security, and presented it. After they started this activity, they had experienced an improved working environment, both regarding psychological safety and security competence. An activity to raise security awareness and knowledge among developers was mentioned by a senior with long experience and a junior with security interest. They suggested showing developers how easily their code can be broken. In an informal meeting, they could live code a small application or show recognisable code. The point would be to use code snippets where everyone in the room could think, “I could have written that”. Further, they could show how the code could be broken. Alternatively, one could use code examples from actual security incidents, if there has been any, within the firm. The awareness would spread by using examples developers can recognise as standard practice and then show how they are exploited. The senior developer explained how effective this activity had been in earlier teams.

Risk assessments are mentioned as an activity that advantageously could be used more. An interviewee said that he sees great value in work with risk assessments, expressed in quote 7.

> “By thinking about what the team should do, evaluate the risks and what risks the work will introduce, we can see the wholesome risk we take in the organisation(...) It can often be a good thing to bring to the leaders and show the risks involved in developing the product they are asking for.” (7)
>
> - *Senior developer*

From the material, we see that code reviews before merging new code contributions play an essential role. This quality assurance could be in the form of a pull request. An interviewee said that the first code fragment delivered by a new employee is essential. He said that there is much culture in how the pull requests are handled. Novice developers’ first pull request should be reviewed carefully with concrete feedback such that the developer can learn from it. He emphasised the importance of being consistent in what comments are let through and what should be altered before acceptance. Another interviewee said that if he reviewed a pull request from a developer with more experience than himself, he would assume that the code was well written. He said that he probably would use less time reviewing it and put less effort into the task. This mentality was also observed from another interviewee.

Automatic testing and tools used are part of the team culture. An interviewee said that automatic tools are costly to use and should be customised to work optimally. He expressed that he wishes for a tool that preferably works in the Integrated Development Environment (IDE), which can help with automatic scans and fixes. Another said that in an ideal world, all software security would be automatic. Developers in our material wish to understand better what happens behind the interface of the tools they are using. Interviewees demand training to utilise the tools as they are meant and understand more of what the tools are doing. Further, some mentioned monitoring as a field they wish to understand better.

An interviewee with a security role reflected on the effect security activities had on the awareness and knowledge of the team. He said that all professional reminders such as lightning talks, CTF challenges or attendance at a security conference contribute to a motivational boost in the following weeks. He believes that the developers bring back ideas and new impulses to their respective projects, but did not think the effect of raised awareness lasted.

4.2.4 Summary of Findings Regarding Team Culture and Activities

We have seen that the overall team culture and group dynamics influence the security work performed by the team. As a new consultant, there is a culture to get into, both formal policies and informal norms and practices.

We see the positive effect of having an individual with a great interest in software security on a team. The firms' competence is the sum of every individual's competence. Hence, individuals with security interest can influence the culture positively. Competence sharing between team members is done in various forms, but mainly through unstructured activities. These include observations and feedback, as well as general discussions and priorities.

Several participants have talked about the importance of having a team where everyone feels psychologically safe. This sets the basis for how respect and communication take place. A culture where individuals are comfortable asking and answering questions benefits the security work.

4.3 Findings Regarding Organisational Factors

We define organisational factors as influences that the developers themselves do not decide. This includes security training, structures for competence development, technologies, tools, and guidelines that form the development environment. In addition, factors include security activities arranged by the consultancy and the use of external security consultants.

4.3.1 Security Training

Most of our interviewees replied "None" when asked about the software security training they received from the consulting firm when they were new employees. Most of them also described a lack of regular security activities provided to refresh knowledge and build competence. An individual informed that the consulting firm had never shown him any resources about software security. Several novices pointed out that the training probably had been weakened due to the pandemic.

We have interviewed three developers responsible for the onboarding programmes provided to new employees. One working for a large consulting firm reflected on the courses provided in the programme. Of the total of 28 courses, he expressed that “maybe one covers security”. The second developer, also working for a large firm, said that they focus on getting the developers comfortable working in teams. To do so, they introduced them to more experienced developers in knowledge-sharing competence groups based on their fields of interest. He stressed that they base the security training on standard technologies likely to be used in projects and do not prioritise details on more specific topics because it probably will not be used directly in work-related tasks soon. The third developer responsible for onboarding works for a small firm where they develop their own product in addition to selling consulting services. He stressed the importance of learning how to apply security in all parts of the process. He pointed out threat modelling and risk analysis as examples.

> “I experience that security training often lacks purpose. It does not become concrete enough for the developers’ tasks.” (8)
>
> *- Senior developer*

An interviewee, employed in a large firm, reflected on the effect of security training, see quote 8. Further, he said that he believes the focus should be on raised awareness on what values one tries to protect, rather than security training in general.

According to an interviewee, customers with a high maturity level provide adequate, specific software security training aimed at the developer’s task. With ’high maturity level’, we mean long experience with security work. These customers can typically be governmental customers within traditional banking, insurance, health and telecommunications. The interviewee said that mature customers often have more available people knowledgeable of security than less mature customers who have less security experience.

We have seen that developers believe that security is a confounding and complex field. Many developers declare that they struggle to grasp the software security most relevant for their tasks because they do not know where to begin. There is a need for concrete courses and specific activities relevant to the developer’s task. A front-end developer suggested recurring courses once in a while, for instance, on the OWASP Top Ten list of vulnerabilities.

Both interviewees with involvement in the CTF community and other developers mentioned that practising CTFs is valuable. Breaking down the complexity of software and viewing vulnerabilities as isolated components improve the understanding of how the system can be exploited. A couple of respondents pointed out that developers could understand how malicious actors break the code they develop. Then, if they learn how they can mitigate these vulnerabilities, they would not be entirely dependant on hiring security consultants in the final phase of the development.

4.3.2 Competence Development

We observe that consulting firms practise different kinds of knowledge acquisition schemes. Some of the participants' companies pay their employees for self-studying. With this arrangement, the consultants can develop their competence in freely chosen topics within working hours. Other initiatives include providing admissions to conferences and arranging lessons internally. From our material, we see that individual developers have freedom in prioritising the topics they want. The consultancy does not require specific topics but rather facilitates the possibility of getting creative and engaging in self-chosen projects.

A consequence of this freedom, expressed by several respondents, is that they do not regularly prioritise software security. There is an endless number of topics they wish to learn about. A junior interviewee said she would like to learn more about software security, but elaborates in quote 9 that she cannot learn everything. Another respondent explained that she does not learn anything before a task requires her to.

> "I would like to learn much more than I have capacity to. Software security is interesting, but there are a lot of other interesting topics as well. " (9)
>
> \- *Junior developer*

4.3.3 Security Roles

Some customers with a high level of maturity assign a security role to one of the developers on the team, which could be called "Security Champion". Several of our interviewees have such a role. They emphasised the value of having an appointed person who aims to raise awareness and quality of the security work. However, they mentioned factors that could enhance the security role.

All four interviewees with security roles were particularly interested in software security. They explained that the role consisted of having an extra focus on security, but had no defined tasks. A respondent reflected on his role in quote 10, and said that there should be a note on what the mandate includes.

> “With no mandate, it sounds boring. You get responsibility, but no opportunity to change anything.” (10)
>
> *- Junior developer, with a security role*

A developer that was responsible for security mentioned both formal and informal tasks. He experienced the formal tasks as defined. Formal tasks include security audits four times a year, which he would prepare and lead. Informal tasks could be to pay attention to change in technology or security threats that could affect the development team. An essential part of the role is to be a discussion partner for security issues. According to him, the security role makes tough decisions regarding security choices that may be unpopular by the team.

Having a security role on the team seemed to raise the team’s quality of security work. The developers with experience with security roles also accentuated that security roles are not a common practice. The priority of this role is dependant on the maturity of the customer and the product.

An interviewee without a security role reflected on how the role works in practice. He emphasised that it is crucial how the individual enacts the role. He points out that a vaguely defined role could contribute to the others leaning on the dedicated security role to fix all the security issues.

4.3.4 Delivery Models

Based on our interviews, consultancies’ delivery models seem to affect how developers work with software security. One interviewee worked for a consultancy that lends out consultants individually to a team responsible for the product. According to him, the consultancy is therefore not responsible for the security of the finished product. In quote 11, he elaborates this opinion.

> “I would have been more stressed about security if we [the consultancy] had responsibility for operating the system. I think there is a massive difference between consultants with this responsibility and not. I have noticed that developers from consultancies with such responsibilities have a much greater focus on security” (11)
>
> *- Senior developer*

An interviewee said that the culture for securing the product comes with the feeling of ownership the developers have for the product. Further, she expressed that her team had continuous responsibility for the products also after launching. She elaborated by explaining the consequence of making bad choices in quote 12. The feeling of ownership made them raise the quality of their work.

> "Products do not become better by themselves. If I make bad choices, it will bite me in the ass." (12)
>
> - *Junior developer*

4.3.5 Traditional Boundaries

Some respondents expressed that security and development are separated into different divisions and do not necessarily need to be integrated. Quote 13 illustrates that the security competence may not be perceived as available.

An interviewee expressed that the customer focuses on letting programmers code functionality and having an external team focusing on security. He also said that the customer does not focus on the individual developer to develop securely.

Another interviewee had operated with a large security team that scanned and fixed security issues in the developers' code. He experienced that the developers assumed that the security team fixed the security such that they did not have to think about security concerns. He emphasised that this is a pitfall because the developers stop thinking about vulnerabilities. He was also worried that the security teams neither knows the logic nor has a feeling of ownership of the code, which could weaken their security audit. In total, he believed this gave the project a false sense of security.

It would be interesting and probably valuable for his work to know more about how the security team work, a respondent figured. He continued saying that developers generally do not know how the dedicated security team works, what they do, or how they think.

> "There is a security team on the project, which it should be possible to talk to, but I have not been in a situation where it has been relevant to include them. (...) We have interest groups on different fields, but I do not know if we have one for security." (13)
>
> - *Junior developer*

4.3.6 Use of Security Guidelines and Checklists

From the follow-up questions on the use of guidelines, the majority of our interviewees had heard of, and to some extent used, the OWASP Top Ten list of common vulnerabilities. Based on these answers, most of our interviewees seem to have heard of OWASP Top Ten, but are not familiar with other guidelines.

A senior developer with a security background had experience with several guidelines. He expressed that he had used OWASP SAMM to measure teams' security position and Microsoft sdl to make SDLCs customised to customers. Thirdly, he mentioned owasp OWASP ASVS as his most used and preferred. He emphasised that these guidelines are a good starting point, but they have to be customised to fit the firm that should be using them. He said that he would like more focus on OWASP OWASP ASVS in addition to OWASP Top Ten, which only covers a small part of security measures. He also mentioned that the "Cheatsheet"-series of OWASP deserves publicity. Lastly, he mentioned the guidelines made by NDPA on build-in privacy.

A junior developer expressed that her development team use a checklist based on the "AWS Well-Architected Framework". It presented examples of what to do with different security concerns. A tech lead could typically arrange meetings where they assess the application against security measures. She also mentioned a checklist for Hypertext Transfer Protocol (HTTP) security headers that she used when developing web applications. The checklist consists of an overview of what headers to set and why they are important, and examples of how to implementing them.

An interviewee said that he experienced the guidelines as very theoretical. For instance, he wished for a more helpful guideline in practice, for instance, a checklist to use in smaller projects or development that does not feel like bureaucracy. Another interviewee with a security role reflected on how checklists work in practice in quote 14. GitHub can offer security checklists in conjunction with pull requests. These are often very general, he elaborated, and do not cover individual issues.

> "There is no checklist that works in practice, that is not how security works. I do not believe that it is possible to make it so simple" (14)
>
> - *Senior developer, with a security role*

4.3.7 Development Technologies and Environments

Some interviewees mentioned technical debt as a factor affecting the development environment. We learnt that many companies have agreements with cloud and infrastructure vendors. This way, multiple technological decisions are often already made when the developers are engaged in the project. An interviewee expressed that he had a limited impact on the environment where they develop their code.

Developers seem to rely on the software created by others significantly. A respondent presented his steps for evaluating the credibility of using third-party libraries. Firstly, he would look at the name of the library. If he recognised it, he would use it based on its reputation. Others explained that the popularity in the number of downloads indicated integrity. Amazon, Google and Microsoft were mentioned as companies trusted to provide secure services. Respondents elaborated that these vendors have a significant team that works on making the software secure. However, although some respondents mentioned that large companies are exposed to security breaches, they still use their services quite uncritically. These parties are considered to be trusted by the developers of our material.

Use of Automatic Tools and Scans

Almost all interviewees mentioned the use of automatic tools as one of the security measures the team do. Several interviewees mentioned that the tools are expensive and must be tuned and configured to work properly. Quote 15 illustrates this.

> "Automatic tools are very expensive. They have to be adjusted. Otherwise, it will be a lot of fuzz. I wish to be notified when I am about to do something stupid, preferably directly in the ide" (15)
>
> - *Senior developer, with a security role*

Multiple interviewees mentioned Snyk, Detectify, SASTs as tools they use. Increased use of the DevOps paradigm discloses the use of more automatic tools. Some also expressed enthusiasm for the possibilities that come with GitHub Actions, where other tools can be integrated as part of the everyday process.

4.3.8 External Security Consultants

An interviewee reflected on hiring external security consultants. He said that the development team worked harder to make secure code if they knew that experts were to review the code afterwards. Another interviewee, in quote 16, figured that the current distribution of responsibilities works well. Others hire security consultancies to review the security during the development phase to meet their required security level.

> "It makes sense how we do it today. The security experts are specialised in security, and other developers do the mindless development. I do not know so much about security, so I cannot say what good it would do to focus more on it" (16)
>
> - *Junior developer*

An interviewee explained that a colleague previously had worked in a security consultancy. Currently, he switched projects every couple of weeks to fix the security issues of many teams. She used the term “Use and discard” to describe how they practice their security work. She elaborated that their task was to move from project to project to “sprinkle” some security on the code. She did not see a constant need for them on a team as their job is to control and verify the system, she explained.

4.3.9 Security Activities Organised by the Consultancy

Our interviewees expressed that security-interested individuals mostly initiate organised security activities. However, it seems like security-related activities get less attendance than activities on other topics. Security-related activities occur less frequently relative to other activities. That said, an interviewee said that he experienced a shift. It seems that there has been an increased amount of security activities in the later years. Senior developers elaborate that more developers understand the importance and show up for security activities compared to earlier. They estimated that the increased initiative has been noticeable over the last five years or so.

Competence Groups

Competence groups for different fields of interests such as software security, web development and cloud computing were reoccurring in the consultancies. Interested people would learn from each other in an engaged environment. From the developers belonging to the same competence group for security, we have seen that security-driven individuals initiate these groups within the firm. The group has resources to arrange meetings within working hours where they share skills or recent security incidents. Here and there, they arrange activities for other coworkers as well, not only within the group. An activity could, for instance, be competence days. A competence day is often a technical and social day or afternoon, including lightning talks from colleagues and dining. The lightning talks could include security-related topics that the colleague recently learnt or understood. The dining part with food and drinks is an arena to better relationships with colleagues and increase affiliation.

Communicating and Discussions on Software Security

The most used communication platform among the interviewees is Slack[3]. The teams and firms use this for informal communication in different groups or channels dedicated to different purposes. Respondents describe a channel dedicated to discussing and highlighting software security concerns and challenges. Security-interested consultants are available for discussions and contributions. An interviewee also expressed that most of the security discussions took place in other Slack channels. He elaborated

[3]Slack: https://slack.com/intl/en-no/

that, for instance, the developers would probably discuss security concerns regarding a specific framework in the channel for that specific framework rather than in the security channel.

Some junior developers expressed that they were hesitant to participate in the Slack discussions on security concerns because they experienced a high threshold to participate. The developers who discussed the most are very often the more experienced security gurus, one remarked.

Several respondents started as developers in the autumn of 2020. Consequently, their onboarding process has deviated from other years due to the pandemic and home offices. The situation has affected their relationships with their colleagues and general knowledge sharing. An interviewee expressed her onboarding process in quote 17.

> “We are the wrong class to ask about security activities. Physical meetings and other activities are cancelled.” (17)
>
> - *Junior developer*

Another interviewee said that he was convinced that other developers would like to discuss security concerns, but these discussions had been lacking since the coronavirus outbreak in March 2020.

4.3.10 Summary of Findings Regarding Organisational Factors

We see that most novice developers lack software security training provided by consultancies. The developers describe security as intricate, and they experience a lack of capacity to learn about all the topics they want. Some teams have a security role that raises awareness and knowledge within the team. The practice seems to increase the quality of the security work, but the role does not contain concrete responsibilities. Consequently, the role’s success depends on how the individual enacts it.

The consultancies’ delivery models seem to affect how their developers work with security. In addition, we see that some consultancies have divided divisions working on different aspects of development. Developers do not always know whom to ask or where to get hold of the required knowledge.

Several interviewees explained that the development environment, including technical debt and using guidelines and tools, dramatically affects security work. The majority of the developers do not seem to use or know security guidelines such as

BSIMM or SDL. However, some mentioned a variety of checklists that they partially use. Almost all interviewees explain that their team use automatic tools and scans.

Whether or not external security consultants are doing code reviews or penetration tests, they seem to affect how developers work. Engagement from individuals is the most important driving force for internal security activities organised by the consultancy.

4.4 Findings Regarding Customers' Effect on Security Work

Almost every interviewee mentioned that the focus on security work in teams varies depending on the developed product. An interviewee explained that it is often easier to suggest security activities and get prioritisation with customers of high maturity level. As consequences of broken software for these industries can be severe, they have experience with prioritising security.

Another interviewee thought that an involved leadership that asks questions is meaningful for the security awareness within a team. He continued by explaining how he notices the customers level of awareness in practice. Positively influencing this are initiatives from the customers such as security training and introducing security roles.

The most experienced seniors developers agreed that there had been a change in software security awareness of the customers in the later years. As earlier mentioned, this also applies to the developers. Customers nowadays ask for consultants with competence in security work and include concrete security requirements in the project descriptions. He pointed out a change in how security gets prioritised and that security has become a topic of conversation. In quote 18, he explained that security gets more attention from the customers. From his formulation, it seems that this used to be a challenge earlier.

> "It is no longer the case that you have to fight a great battle in order to succeed. It is easier to get security tasks prioritised in the backlog because security activities have more focus further up in the organisation." (18)
>
> - *Senior developer, with a security role*

4.4.1 Start-Ups Differ from Mature Customers

In start-ups, the experience and maturity level is different. An interviewee hired for a start-up explained that the products are often Proof-of-Concepts made on low budgets. A Proof-of-Concept product aims to make something that works, so they have something to show investors to get money. They often create a product, which later is destroyed. As illustrated in quote 19, the interviewee emphasised the importance of not being naive even though security is not the main focus in these projects.

> "It is easy to think that the website does not have considerable traffic yet, but that is a naive approach. Crawlers try for vulnerabilities either way. It is important to take it seriously and not lean on existing experiences. Security is a constantly evolving field." (19)
>
> *- Junior developer*

A difference he noticed between start-ups and more experienced customers was that developers blend in with the existing culture with mature customers. However, with less mature customers, the developers affect and contributes to evolving the culture to a much greater extent. Another interviewee added that there more established frameworks are available in projects for experienced customers. He also pointed out more defined security requirements and colleagues qualified to discuss concerns. Further, his impression was that start-ups do not have any existing software to reuse and therefore may not prioritise using developer competence on security work. Moreover, he explained that he adapts to the customers level of focus on security. If the customers do not focus on security work, he lowers his shoulders and tries not to make completely idiotic mistakes.

4.4.2 Reputation

Several interviewees mentioned the effect news coverage of broken systems has on customers. Generally, they are afraid of being hacked and then receive bad publicity. Their product would then get a weakened reputation. An interviewee discussed that this is one of the most important reasons why customers have become more willing to prioritise resources to security activities in recent years.

4.4.3 Trade-Off Between Security and Business

An interviewee highlighted that there is a trade-off between software security and business considerations. He said that he had experienced that the customer did not follow his recommendation, see quote 20. He also expressed that economics and budgets play a large role in prioritising and doing security activities. Software security is costly, and product owners need to weigh it up against other business concerns.

> “The customers know their industry best, and consultants are often hired because the customers lack IT competence. This may create some unnecessary friction because we do not have the same understanding.” (20)
>
> - *Senior developer, with a security role*

4.4.4 Summary of Findings Regarding Customers’ Effect on Security Work

We see that the customers initiatives as well as the product developed influence the software security culture. Our interviewees expressed a change in customers’ overall awareness. It is easier to get security tasks and activities prioritised now, compared to earlier. Several interviewees expressed that this is due to news coverage, fear of bad publicity, and consequently weakened reputation. In addition, the developers emphasise the importance of understanding the trade-off between security and business considerations.

Chapter 5
Discussion

This chapter discusses the findings from chapter 4 and relate the findings to earlier research. The interview findings are based on both the direct responses and our interpretations from the participants as well as insights we have developed across all interviews.

The sections address the three research questions. Firstly, we view software security culture from an individual perspective. Secondly, we view organisational and predetermined factors that affect the software security culture. Then, we will present material factors that influence the software security culture. Lastly, we discuss factors affecting software security culture in general. The different topics presented in different sections are linked.

5.1 RQ1: What Factors Influence the Software Security Culture on an Individual Level?

Identified factors related to individuals regard individuals' interest, their effect on a development team and perceived responsibility for secure programming. We also discuss responsibility versus accountability in this section.

5.1.1 Personal Interest

Our research shows that the most interested individuals seem to have the most knowledge of software security. As described, they learn by reading blogs, watching videos, solving CTFs and seeking out resources to learn more. They derive their knowledge from self-initiated activities. This finding aligns with earlier research, where they conclude that security expertise is self-taught and often come from news and blogs [17].

We have seen that team members learn software security from each other through discussions and other spontaneous, non-organised activities. Earlier research concludes that security knowledge spreads between developers through informal discussions [17]. Without an interest in the field, there may be less spontaneous competence sharing, which blocks the advancement of security competence. Hence, a lack of interest can affect how the team performs security-wise in the long run. We have experienced that software security is a complex and dynamic field that transforms all the time. New technology introduces new vulnerabilities, and hackers find new ways of exploiting software.

More than half of our interviewees seemed indifferent or uninterested in software security, self-reporting to "Neither interested nor uninterested (3)" or "Somewhat uninterested (2)". These interviewees did not have more engagement for security than what they stumbled upon and found relevant for projects. Developers lack intrinsic motivation, and software architects do not seem to have security as their main interest [6, 22].

One of the participants surprised us with his self-report on interest in software security. During the first interview, we observed that he did not seem interested in security at all. In quote 2 he stated that he only regards the functionality of the product, where he did not enclose security. Also, he added that he did not know much about security and would not say that there is a focus on learning it.

In the follow-up conversation, he seemed somehow uncomfortable when self-assessing to "Very interested (5)" on our scale. We wonder why our initial impression and his answer does not align. In the follow-up conversation, he explained that he was entering a security role in his next project. Thus, he currently read up on web application security. He emphasised that he did not have experience or knowledge of the topic yet, but was motivated to learn. The radical change in interest could be a consequence of changed job assignments. Developers get motivated to learn security due to a feeling of responsibility [17]. The new responsibilities due to the new task may be the reason for the sudden experienced interest. Another theory is that his participation in our thesis has contributed to his awareness and interest in the field. The conversations we had may have affected him in thinking of why software security is necessary. Further, he may have wished to fit our material and not stand out negatively. Another possibility is that he had become very interested in software security in the last couple of months.

5.1.2 Individuals' Initiatives Within Development Teams

Our results show consistently that individuals' initiatives are essential for the culture in a development team. The respondents commonly emphasised that this is the most important factor for the quality of security work. Individuals make up the culture in a team. Thus the team culture varies within a firm. The overall security in a project depends on the knowledge of the decision-makers, either on the customer or developer side [17]. Lack of software security competence impacts the final software security in a solution [2]. As described in our findings, an interviewee expressed that a basic level of knowledge is required to make good security choices.

5.1.3 Increased Personal Responsibility

There seems to be a change in the extent developers feel responsible for software security. The “it is not my responsibility”-attitude on software security found by Xie et al. in 2011 [19] may have evolved. Research from 2014 and 2017 conclude that the developers believed that they had a shared responsibility for writing secure code [7, 20]. Even though testers and security experts helped the quality of the security work, the developers have the most significant responsibility. In research from 2017, developers expressed that it was part of their self-conception as professional technicians to deliver good quality, sound, and secure software features [6]. These findings align with our findings on experienced responsibility. Most interviewees believed that they were partly responsible for security in the software they develop. The change from 2011 until now, ten years later, may represent and reflect that a shift is happening in the position security work has in development teams. However, some individuals in the selection believed they were not responsible for the software security. Further, many reflect on the consequence news coverage will have on their consultancy’s reputation. They express that such bad publicity due to an exploitation of a common vulnerability would feel embarrassing.

Our material described that they are hired to do a job. Many stated that this includes fixing the most obvious vulnerabilities. Due to time constraints and customer deliveries, they need to do as they are told and expected. Simultaneously, their competence is valued, and if they take responsibility for adding security, this is mostly supported. However, the accountability for security actions is not clear [1]. One respondent commented that one single person could not be held accountable for the security. Although the security tasks can be delegated and the responsibility can be shared, being accountable for the code security when someone is to blame is tough. We wonder if someone could or should be held accountable.

The customers are buying competence to build a service or a product. They are responsible for setting requirements, providing enough money for security work in their budgets, and providing resources. On the other side, we argue that the developers are responsible for raising their voice about concerns when issues arise and making educated choices while developing. According to interviewees, the customers are more aware of the importance of security work now compared to earlier. They prioritise differently and asks more specifically about consultants they know are good at security work, as well as security requirements. Security measures cost money [2]. Consequently, only customers with sufficient financing may have the ability to buy all the security measures required. This might be a challenge for less fortunate customers.

We had a challenging time defining the fields of responsibilities for the consulting firm. Since not all customers can buy all the security measures they would like, we could ask if the consultants are responsible for guiding the customer to make adequate decisions regarding security. In addition, we argue that the consultancies are responsible for the quality of the security knowledge of the consultants they sell out. Then we could ask, are the consultancies aware of this responsibility? We think they might not, based on the lack of security training that several consultancies provide. As described, the delivery model of the consultancy seems to affect what responsibility they take. To conclude on who can be held accountable, further research on the topic is required.

5.1.4 Summary: Individual Factors

An individual's interest in software security is the most important factor for the development of their knowledge. Individuals' initiatives are important for the total security work in the teams, and all stakeholders will benefit from an individual who initiates security activities and maintains the security awareness. Developers seem to feel more responsibility for code security now compared to earlier.

5.2 RQ2: What Organisational Factors Influence the Software Security Culture?

Factors on an organisational level include security training in both onboarding processes and education. This also includes team culture, which could be influenced by psychological safety, diversity and security roles. In addition, security activities influence the security culture. Examples can be security training, engaging external security consultants, and unstructured activities such as informal discussions.

5.2.1 Compulsory Security Courses in Education

In the report on supply and demand for security competence from 2017 delivered by NIFU, they conclude that there is a considerable need for security courses in engineering educations is considerable. They express a need for generalists, meaning candidates with general Information and Communications Technology (ICT) security competence. They emphasise the importance of including required security courses in all ICT study programmes. Several also point out that security courses should be compulsory on all technical study programmes [8].

Today the master's degree programme, Computer Science, and the bachelor and master's degree programme in Informatics at NTNU could be completed without taking a single course that mainly focuses on security work. The students may prioritise taking some elective courses on security, but may not see the need for them. Not requiring any security course is concerning, as a significant part of the students become developers after a finished degree [8], without own reflections on why security is necessary and know how to mitigate common vulnerabilities.

Our interviewees have a variety of backgrounds. Some respondents have elected security courses during their education, and others have avoided them. Principally, individuals lacking security interest did not elect courses on security. The freedom to deselect security courses result in very different foundations. Even from the same study programmes, developers have a very unlike base of security knowledge when entering development projects as consultants.

A majority of our interviewees stated that learning by doing is their preferred way of gaining knowledge. For example, as suggested by an interviewee, one could include software security in existing software development courses. If security is included as a natural part of teaching and project assessment, it may be communicated that security is a valuable measure in the finished product. Thus, a possibility would be to include security measures as part of the assessment criteria in software development projects. A change in criteria could signal that software security is an integrated part of the craft of developing satisfactory code.

5.2.2 Team Culture

As the research from Stray et al. and Shin stated, an agile team is more efficient when there is psychological safety within the group [69, 35]. Shadowing a more experienced developer, suggested by a respondent, indicates that shadowing could ensure a quick adaptation of the norms and practices regarding software security culture and the culture in general. However, this requires the experienced developer to be aware of the culture she wants to pass on to the novice employee. Otherwise, the bad habits of the experienced developer could be passed on along with the team's habits and culture. This is a resource-intensive scheme, but practical to learn the culture quickly.

According to van der Heijden et al., all team members should share a common understanding of the team culture and the firm's culture on software security. To convey practices and attitudes concerning software security, it is valuable to raise awareness of the culture [1]. Experienced developers of our material have expressed challenges with including new developers in their software security culture. Consequently, if the culture is explicitly familiarised it can easily be passed on by the experienced developers on the team.

Some participants mention a high degree of turnover in teams and that projects are dynamic. Due to projects arising and disappearing frequently, hired consultants and internal competence are relocated. External consultants switch teams on demand of competence, and they will do as they are told when entering a new team [1]. The team culture has to be related to the inherent values of the team, so it does not disappear in turnover [70]. These values should be commonly known and used to navigate actions. A software security culture needs to get built in the same way, although it may be unclear unless specifically addressed. A possible software security culture will be hard to recognise without structured guidelines and established values.

According to an interviewee, using DevOps increases the field of responsibilities of a team. A consequence could be an implicit increased feeling of ownership in the team. Writing code to the production environment becomes easier since no other dedicated operating team should fix bugs. The team can, in other words, fix their bugs and push the changes to the production environment. This increased feeling of responsibility and ownership seems to affect the quality of the security work in a positive manner.

Diversity

Our data suggest that a diversity of knowledge and experience is a valuable strength in a team. Several interviewees said that there is no goal to have a team of equal developers. Different perspectives positively challenge the team.

Interviewees mentioned the value of combining juniors and seniors. Both experienced developers and novices could learn skills from each other. The seniors' experience and knowledge are built up through the years. This is valuable knowledge to share with less experienced developers. However, less experienced developers enter the team with fresh eyes and a different mindset. They may be more updated on technology and contribute with alternative methods. In addition, they may raise awareness by making unpredictable mistakes. These contributions could challenge established truths and push the team's progress and development further.

As commented in the results, senior developers are assumed to write more secure and correct code than junior developers. Thus, their code is less thoroughly examined. However, a senior can be unfamiliar with security practices or lack experience in the field. This mentality could involve a greater risk of pushing vulnerabilities into a production environment if this is the case.

Our data do not explicitly mention diversity in terms of educational background. From what we see, the developers with other educational backgrounds than developer-related studies did not feel disadvantaged when handling security issues. Possibly, they do not experience any disadvantage because developer-related studies may not involve enough emphasis on security content.

Earlier research from Kocksch et al. discusses that women improve the security work of a team [71]. Our research has not explicitly asked about gender diversity. Cultural diversity has neither been mentioned by our respondents nor been considered in earlier research.

Assigned Security Role

Security knowledge of individuals is essential for the security work done in teams [17, 26]. We have seen the same in our results. Without this driving force for security, teams may fall short in focusing on security. Van der Heijden et al. present that close involvement with a security role is valuable for teams, mainly because the role facilitates increased acceptance of security [1]. Introducing a security role is a way of formalising responsibilities for security focus. Our results suggest that developers with security interest often are delegated the security role on their team, in addition to develop code. We have seen that mature customers initiate this role. By that, we mean customers who take actions affecting efficiency and security actions because their products handle critical and personal data. Our participants with experience from security roles express that the role contributes to increased security focus. This is also described in van der Heijden et al.'s work [1].

We see that the security roles are vaguely defined. As described in our results, the role lacks specific tasks, responsibilities and mandate. When involving a security role, we consider it essential to define and describe how the developer enacts the security role. A more concrete definition of the role would make it easier to follow up on assignments and responsibilities. Additionally, the security role would be structured and established as a recognised practice. Today it seems arbitrary if the security role works well or not. The role depends on the initiatives of the individual developer. In advance of a project, the initiator could standardise the security role to be quantifiable and manageable, regardless of the individual's level of ambition, awareness and knowledge.

As mentioned earlier, the accountability for security actions is unclear [1] and one individual cannot solely be held accountable for the code security. Another interviewee explained that one mistake done by an individual never is the cause alone for broken code. He elaborated that a series of events and structural influences also play a role. Thus, we argue that the individual who holds the security role cannot be held accountable. If the security role is held accountable for the software security as part of their responsibilities, we argue that the scope of the role should be defined. The developer takes greater responsibility in organisational work and should be part of strategic and operational decisions affecting the team.

5.2.3 Security Activities

The respondents said they find it hard to initiate activities to raise awareness and knowledge on software security. We have observed several cases where developers have experienced software security training as irrelevant, redundant or superficial. It seems challenging to find activities that work well in practice.

There are both organised and unorganised activities in a consultancy. Organisational activities include, for instance, competence groups, security training and external security consultants. Our interviewees expressed that their security knowledge mainly results from knowledge sharing within the team and on other developers' experiences despite various structured activities. Security knowledge spread through informal discussions among developers [17]. In our material, these include discussions related to current security topics, solving project tasks and helping others. Discussions are either physical or digital in communication channels on Slack.

Competence Groups

As described in the results, some consultancies have competence groups for a variety of topics. This creates environments where interested people learn from each other and create engagement for learning. Competence groups are a type of satellites described in BSIMM [52]. Consulting firms that operated with such satellites are often categorised as large. From what we have seen of the competence group, its driving force is enthusiastic individuals within the firm. The group has available resources to arrange meetings for knowledge sharing within working hours. Occasionally, they arrange activities accessible to colleagues outside the group. Examples of activities are competence days, framework courses, lightning talks and lectures given by external experts.

An interviewee said that it can be challenging to spread knowledge to the rest of the consulting firm. Within the competence group, the initiatives and knowledge grow, according to him. However, the group has limited spots. This can be experienced as a barrier for developers who may be interested in security knowledge, but do not feel qualified or "passionate" enough to join. One interviewee expressed that she needs basic knowledge of software development before joining the competence group on security. A novice developer may experience a high threshold to join a competence group in fear of not contributing. Furthermore, the spot could have been offered to someone else. We believe there is value in spreading knowledge to the whole organisation, making an input on security available to more people.

Our results suggest that attendance to security-related activities is lower than for activities related to, for instance, web development, software architecture and artificial intelligence. This may reflect the personal prioritisation of security. We have seen that developers do not manage to cover all topics because there is a lot to learn.

Security Training

There are wide variations in security awareness, and for that reason, training is crucial [1]. In our material, the security training provided by the consultancies during the onboarding process varied. While some encountered a broad set of coursing and certifications in their first period, others were sent straight to customer projects without training. Organisations lack structured software security training [22]. Several interviewees explained that they learnt how to protect their computer and avoid publicly open networks in their onboarding process. Some mentioned lessons on handling personal data according to GDPR and an ethics lesson. However, few referred to training on software security. This aligns with earlier research done by Jaatun et al. They conclude that software security does not seem prioritised in the general effort to increase knowledge and awareness of information security [3].

Our overall impression is that minimal security training is offered to novices. Some developers in our material seemed to have the attitude that security training should not be mandatory. They said it is not necessarily desirable that all developers have a baseline of software security knowledge. Team diversity, as mentioned earlier, will not be urged if everyone is forced through security training.

We see that the developers experience the current security training as not fitted or relevant for their tasks. The attitude is that security training is costly and hard to make specialised for the developers' tasks.

Novices may not be aware of the typical security pitfalls, according to an interviewee. Their capacity to learn new skills is limited, and security is a complex and comprehensive field. We grasp that it would demand excessive resources to customise a security lesson to all developers, which take into account their prior knowledge and the project they are becoming a part of. However, we believe that it is beneficial to provide a baseline course. This will raise the general awareness of software security. Additionally, it would present them with various resources to seek solutions and further expand their knowledge. The developers would maybe feel better equipped to implement security tasks. An interviewee said that he had never before been shown resources on software security. Senior developers also wish for a general course on security to stay updated and learn about new vulnerabilities.

External Security Consultants

Earlier research observed a short-term change in motivation regarding security when engaging security consultants in development teams. However, in the long term, the sustainable changes and adoption of new practices and tools did not continue after the security consultants were out of the projects [6, 23]. We have not researched the effect of security consultants, but we can see a similar effect after completing security activities. We observed that some developers had noticed differences in motivation and interest a few weeks after successful security activities. The developers more often initiated discussions and expressed their security concerns. However, this change only lasted a few weeks after the activity. We interpret that the effect is not sustainable and long-lived. One respondent mentioned that security work could be considered resources to "use-and-throw", only initiated by security consultants. This attitude does not seem common in our material, but should be addressed. This attitude does not seem to conform with the goal of security in all parts of the process.

Consulting Firms' Incentive for Structured Security Activities

We have seen that consultancies often provide inadequate software security training. It is plausible that consulting firms' incentive to have structured training for software security is influenced by their delivery model. This includes what kind of assignments and teams they mainly offer to customers and the time horizon they operate with. From our material, consultancies provide software security training more often if they operate their own products or have a great responsibility for their customers' infrastructure than if they do not.

As several respondents expressed, their security knowledge is mostly obtained through earlier customer projects. When a customer project requires specific knowledge of a field, the developer can learn that skill set at the customer's expense. This may influence the consultancy's incentive to provide software security training.

5.2.4 Separated Divisions and Security Teams

From our material, the size of the consultancy seems to affect security work. Employees in small and medium-sized firms seem to have an overview of all their colleagues. They know what knowledge they possess and whom to contact for questions on different topics. On the other hand, teams in larger firms work more isolated. Different competence is often divided into different departments of the firm. Security issues are mostly discussed over Slack. Formulating written questions and asking them in front of many observers in a chat may create a barrier to addressing security concerns. Accordingly, the size of consultancies seems to affect the availability of security knowledge. Van der Heijden et al. concluded that security-related information should be easily available to the team [1]. The information should be available to those who need it when they need it. Quote 13 from the previous chapter may indicate that security knowledge does not reach all developers. The quote refers to a junior developer who expressed that allegedly, there was a security team on the project, but he had never contacted them.

Xiao et al. conclude that having security teams in the presence of development teams may negatively affect their adaptation for security. Their research concludes that security teams may lead to developers feeling pressured to code securely. However, the developers may get the impression that the overall security is not their responsibility [7]. We have observed similar behaviours where neither the developers nor the security consultants fully understand or get ownership of the code. This may weaken the software's security.

Tøndel et al. express a tension between different groups, for instance, between architects and security experts [22]. Additionally, van der Heijden et al. conclude that there is a lack of understanding between information security consultants and the development team. They elaborate that it often feels like they are chasing different goals [1]. Developers express a wish to understand more of what the security team does. In addition, it would be valuable if security consultants understood more of how development teams practise and how they prioritise security. Consequently, greater understanding could contribute to adequate involvement at earlier stages of the process and security insights adjusted to their needs. Our findings do not cover possible tension, but they support the lack of insight and understanding of work across departments.

5.2.5 Customers and Customer Relations

Earlier research states that customers and product owners contribute to the security work with their domain knowledge, even if their security awareness is low. Further, they conclude that close involvement of the customer and product owner is endorsed [25]. The benefits of customer involvement align with our findings. An interviewee expressed the importance of being aware that the customer knows their industry. Therefore, they may act on a different basis than a consultant and can contribute with different knowledge. The nature of a consultant is to pivot through different industries. A consequence may be that they do not have specific competence regarding business concerns. Hence, the interdisciplinarity of teams seems to be valuable for security work.

As mentioned in the results, several interviewees experience that customers generally seem more knowledgeable of security now than earlier. Further, they find it easier to get through security activities to the customers. The raised awareness among customers may be caused by increased news coverage of great hacks and compromised companies in recent years. Customers may feel responsible for securing their products. Perceived threats, in particular related to reputation, increase security awareness [2]. The fear of news coverage and receiving a bad reputation may lead to more consciousness from the customers' side. This may also be the reason why the customers seemingly have a more compliant attitude towards security work. Regulations taking effect, such as GDPR, can also influence the responsibility towards security work [22].

Our interviewees express that customers' security awareness impact the attention security work gets and how security requirements get prioritised. Earlier research support these findings and elaborate that customers' awareness can both be a drive and a hinder for security work [25]. Security is viewed as an implicit goal, and many customers assume it gets done by the developers [17, 6]. If the customer is knowledgeable of security, they may contribute to concretising security requirements for the product. In the end, it seems like the customer's software, resources and ambitions decide how security is weighted in their project.

The vast majority of our interviewees said that the product they develop affects how they work with security. For instance, a proof-of-concept prototype developed for a start-up encounters an entirely different threat assessment than a product created to handle user data for an established insurance company. Multiple interviewees expressed that they do not think about software security because their system does not handle sensitive data. We believe this attitude is unfortunate. When systems without any sensitive data are broken, there are possible other consequences even though no data is compromised. The reconnaissance phase of attackers, where they gather information about the infrastructure, such as a stack trace of used database version, server type or network services, becomes easier if the software is insecure [72]. If the vulnerable software reveals system components and versions, an attacker could find research on known vulnerabilities to these specific systems and exploit them. This information could be a way into other applications and dependencies with possible sensitive data. Other systems using the same components and versions may be exposed [72].

An example of another possibility is to perform a Distributed Denial of Service (DDOS) attack on the vulnerable software and take down the server. If other applications also run on the same server, the consequences could be significant, disconnecting multiple services and blocking their availability [8]. To sum up, the consequences of breaking a system, whether it contains personal data or not, could have extended effects outside its own service.

Security as a Functional Requirement

Developers view security as a non-functional requirement [6]. Additionally, functional requirements get prioritised over non-functional. Further, security is viewed as costly and not adding value [25]. The developers in our material have the same attitude. Hence, if security can become a functional requirement, it can be easier to prioritise security tasks daily. Use-cases play a primary role in defining a goal-oriented set of interactions between the user and the system, while functional requirements capture the system's intended behaviour [73]. Integrating security concerns with existing use-cases or defining use-cases based on security issues create concrete goals for how the system should work. For instance, a use-case could be: "As a user, I want to feel comfortable that my data is protected from eavesdropping in the transmission channel". Further, a functional requirement could be: "Encrypt the communication end-to-end" Then, concrete work tasks could be identified from this functional requirement and prioritised accordingly. In other words, security becomes part of the functionality of the software.

5.2.6 Summary: Organisational Factors

Today, IT students go through their education without compulsory security courses. To some extent, the educational institutions communicate security and development as two fields of expertise that does not have to be intertwined. Psychological safety and trust in the team are important for security-related work. A team diversity benefits from different experiences, views and knowledge. There exist various structured and unstructured security activities. The software security training provided to novices in a consulting firm is minimal. The consultancy delivery model seems to affect the incentive to provide security training and other security activities. Assigning security roles is a good initiative, but the role should be defined with tasks, responsibilities and mandate. Some projects engage external security experts or a dedicated security team to review their code. This can affect the security work of the team to be better, but the developers may care less about security. Knowledge sharing between developers mostly happens in unstructured, informal discussions [17]. The customer affects how the team works depending on what product is developed, and their level of maturity. Customers' security awareness is increasing, and developers achieve security activities more often than before. Security may be a part of the system's functionality instead of being viewed as a non-functional requirement.

5.3 RQ3: What Material Factors Influence the Software Security Culture?

Material factors that influence the software security culture include the trust in third-party dependencies, consequences of technological dept and the use of guidelines and automatic tools.

5.3.1 Dependencies

According to industry estimates done on using code from third-parties in large development projects, 85 per cent of a typical application is third-party code [46]. This percentage is an estimate for development in large projects. The convenience of reusing code applies to both efficiency and complexity such that the developers do not have to reinvent well-functioning features repeatedly.

Developers in our material also seem to trust third-party software. The number of up-votes, monthly downloads and reputation guide whether the software is trustworthy or not. Services provided by leading vendors such as Google, Amazon or Microsoft seem to be trusted. As far as we have seen, there do not seem to be standardised protocols for using third-party code. The developers must learn this through the culture.

Technological choices made earlier by the organisation or on the team may affect the software development through the years. This could be licence agreements with a cloud service vendor, frameworks or tools that all solutions build upon. Organisations may be locked-in when committing to one actor or technology [32].

Most of our interviewees use cloud service platform when developing. We argue that all strategies for involvement in cloud-based development should include an escape plan. If not, the cloud service platforms' lock-in effect may be decisive. Bass et al. state that all the tools a platform provide raise the possibility of vendor lock-in. This problem has existed in the computer industry for 50 years and is not a new problem, they continue. To mitigate the risk of vendor lock-in, standard languages and interfaces can be used [32]. Developers in our material feel inflexible in using other tools than those provided by their firm's cloud service vendor. They rely on their cloud service vendors' security tools to be secure and fitted to their purpose. Using these tools makes it hard to avoid non-standardised tools and to mitigate vendor lock-in.

5.3.2 Use of Security Guidelines

The use of security guidelines and checklists in practice are lower than we initially expected. We anticipated that consultancies would assess guidelines to some degree and that the average developer would know the most common ones. However, our findings align with findings from 2017 [74, 6]. In 2010, a study showed that the adaption rate to Secure Development Lifecycle (SDL)s are low. "Too time-consuming" was the most common response to reasons not to implement an SDL [74]. A study from 2017 mentions that guidelines such as Microsoft SDL and Seven Touchpoints for Software Security fall short on how to get organisations and the actual development teams to start using them [6]. The respondents who knew guidelines such as SDL had experience as security consultants or had a particular interest in the field.

5.3.3 Automatic Tools and Scans

Multiple of our interviewees mentioned the need for better automatic tools. Research by Xie et al. from 2011 point out that automatic tools could help bridge the gap between general security knowledge and concrete secure programming practices [19]. Over the past decade, automatic tools have become a natural part of many teams security practice. Nevertheless, developers in our material still wish for better and more efficient automatic tools. Several interviewees said that in an ideal world, all security work would be fully automatic. Thus, we can imagine that it may be tempting to use many tools at once. Challenges related to this are to understand the tools' limitations and to comprehend the full scope of what follow-up is required [33]. Tools must be configured for their purpose and regularly updated. We interpret that a consequence of misconfigurations could be that the tools do not monitor correct parameters and do not reflect the state of the software. This situation can provide a false safety of believing that the software is secure when it may not be. We argue that the individual developer cannot stop thinking independently. In order for the tools to help, their feedback must be interpreted and acted upon. For this, the basic security knowledge of developers is required.

5.3.4 Summary: Material Factors

The adoption of security guidelines in development teams is not very widespread in practice. Third-party code is used to a great extent in developed code today. Developers often trust third-parties based on their reputation, usage rates and if the actor is well-known, such as Amazon, Google or Microsoft. Development teams use automatic tools to scan for vulnerabilities and to fix them. However, the developers cannot stop thinking independently.

5.4 Relationships Between Individual, Organisational and Material Factors in a Software Security Culture

Software security culture is an interconnected phenomenon covering individual, organisational and material aspects. In this section, we discuss software security culture across the three described research questions.

Agile development is common among our interviewees. Everyone in our material express that they use a variant of this methodology. For an individual developer, this may influence the time-pressure they have to meet deadlines. Short iterations lead to pressure, which can further lead to problems integrating security activities [17]. Methodologies that do not consider defined iterations, such as Kanban, do not involve this pressure.

Agile development methodologies are dynamic, and the priorities are constantly shifting [25]. Thus, it is easy for security tasks to have insufficient priority if not explicitly addressed. It seems reasonable to assume that an agile development methodology also affects how the customers determine security requirements and allocate resources. We believe the methodology also affects how consultancies define their delivery models. Because security is commonly regarded as a non-functional requirement, it often gets lost in daily work due to other functional tasks that need to be completed [6, 22].

The psychological safety within the team may be valuable for the performance [35]. Our findings suggest that psychological safety also contributes to discussions on security concerns. We argue that trust between team members also affects the competence sharing between team members and across teams. If a developer feels comfortable in their team, it may be easier to contribute to discussions on Slack or competence days across teams in the organisation. Thus, competence is spread dynamically across the organisation, and a knowledge-sharing community is obtained. Another consequence of psychological safety within a team could be an increased feeling of ownership, which again affects the incentives and motivation for performing good security work.

Organisations impact the formal ownership development teams carry for security work [25]. A holistic approach of the product motivates gaining security knowledge due to a feeling of responsibility [17]. By introducing the DevOps paradigm to development teams, developers are getting more responsibility for the software. A possibly increased feeling of ownership of the product may motivate security work. The DevOps paradigm increases the usage and needs for automatic tools [33]. Through this practice, developers may impact what tools they use. However, organisations influence what security tools the team have available [25].

Our material suggests that educational institutions and components in the study programmes influence attitudes towards software development. Studies contribute to shaping future developers and is often the first impression individuals have of programming. Moreover, professors and courses emphasise different subject fields within software development. We argue that their influences can affect what the students find interesting, valuable and essential. If study programmes had emphasised security work, the student's motivation to further learning might have increased. Further, the individuals can earlier contribute with their security experience in their professional careers and find joy in further exploring it.

Based on our interviews, we see that software security culture is affected by a variety of factors. Software security is not exclusively dependant on developers' knowledge. Business and management aspects influence resource allocation, prioritisation, employments, and strategy, affecting the software security culture. If all parties influencing a project knew a little more about how their actions affect software security culture, the sum could better the quality security-wise.

To sum up, software security culture is an intertwined field that must be seen in the context of other fields of expertise. This thesis has touched upon other fields such as business aspects in strategy and management, organisational psychology and social anthropology. We argue that software security culture needs contributions from other fields of expertise to raise the overall security work in all industries developing software.

5.5 Recommendations

We turn the focus from knowledge building in this field and derive recommendations to improve software security culture. We base these recommendations on insights obtained through this study, including suggestions from interviewees and earlier research.

5.5.1 Security Knowledge and Training

- Technical study programs should introduce compulsory security courses. Already existing project-based courses in programming should also include security. By integrating software security as a natural part of the course and adding specific assessment criteria on security measures, the educational institution will communicate that security is a natural part of the code craft.
- Basic software security training should be a part of onboarding courses. Such training can raise awareness and provide resources to expand developers' security knowledge further.
- The developers using automatic security tools should get training in interpreting the outputs from the tools correctly. This training seems essential to be able to make adequate decisions and measures.
- When using third-party code, developers should be aware of security risks regarding unfamiliar code. They should get training in how to consider such risks, as well as how to initiate adequate measures to mitigate them.

5.5.2 Team Culture

- Security-engaged individuals should be encouraged to contribute to the team's awareness and knowledge by initiating activities and raising discussions.

- The individuals should contribute to psychological safety through constructive feedback to peer developers and participation in social activities that form the development group.

5.5.3 Structure

- Security work should not mainly rely on personal initiative. Consultancies would benefit from structured security work and organised activities in addition to activities initiated by individuals.
- A security role should be initiated in more projects. The practice seems to increase the value of the security work done by a team. Further, the security role should be more precisely defined, including tasks, responsibilities and mandate. Defined roles will set the practice in structure and make it less arbitrary if the role works as intended. By giving a developer both responsibility and mandate, the individual will have room for manoeuvre when needed.
- Organisations using guidelines should allocate resources for the guidelines to be customised precisely according to the organisations' needs.
- Automatic tools' abilities and limitations should be explored before putting in use. In addition, what resources required to configure and maintain the tool should also be considered.

5.6 Limitations

In the consulting industry, we see that the developer's responsibilities and environment vary greatly depending on the customer and project. This is a challenge when developers on different projects for different customers are being compared in a study.

To better discuss the factors that affect the consultancies' security work, we could need a broader understanding of their delivery models. We have not researched what contracts the consultants work with or how teams are composed.

Many interviewees had experienced an abnormal work situation this year due to the coronavirus. Thus, their experiences may not reflect how teams and firms operate during a normal year without a global pandemic. Onboarding processes, informal discussions, competence days, psychological safety are some of the areas affected.

Chapter 6

Conclusion and Future Work

We have researched factors that influence software security culture in development teams. Factors studied are divided into three categories: individual, organisational and material. This thesis has conducted semi-structured interviews along with a literature review as main sources of information. Findings from the interviews are discussed in light of earlier research on relevant topics.

Our main findings on individual factors regard interest and perceived responsibility for the security in the finished product. Almost half of our material place themselves as “Somewhat interested (4)” or “Very interested (5)” in software security. Most of them are interested in learning more concrete security measures relevant to their working tasks. In the context of earlier research, our findings indicate an ongoing shift regarding how developers perceive their responsibility for securing code. Whereas research from ten years ago described a disclaiming attitude towards security, most of our respondents felt co-responsible. However, the developers experience security as an intricate and extensive field, making it hard to initiate learning measures. Even though they are aware, interested and feel responsible for the software security, several do not take actions to raise their specific competence on the topic. Challenges may lie in the capacity of the individual developer, prioritisation and attitudes mirrored from the leadership.

Our main findings on organisational factors regard undefined incentives for security activities and security roles in practice. The delivery model of the consultancies seems to affect their incentive for structured security activities. The nature of consultancies, with consultants in and out of projects, may affect their feeling of ownership. Consequently, security activities seem to be less prioritised, and security work lacks attention. Security training in the developer’s education also seems weak. Accordingly, it is possible to go through a five-year study in Informatics or Computer Science without taking one security course.

Today, customers and projects that require a high security level are often based on practices where a developer has a security role on the team. These projects are often associated with traditional industries, such as banking, insurance, health and telecommunications, where their products handle a significant amount of critical personal data. Still, the practice seems to have potential for improvement, as the security role often lacks proper work tasks. With defined responsibilities and mandate, the practice would be structured, and the role's contributions would be less arbitrary. We see that a successful security role is strongly dependent on the individual's initiative and how she or he enacts the role.

Our main finding on material factors is that automatic tools and scans are broadly used to detect vulnerabilities. However, they may lead to a false sense of security if not used correctly. The importance of knowledgeable, involved developers who know how to interpret the tools' outputs cannot be underestimated.

Based on our research, we derived some recommendations. The most important regards education, security training and security roles. Firstly, security should be a more significant part of study programs that educate developers. Secondly, basic software security training should be provided as a part of the onboarding process for new employees. Moreover, security roles should be initiated in more projects. In addition, the role should come with a set of defined tasks, responsibilities and mandate.

To sum up, software security culture is an intertwined field that must be seen in the context of other fields of expertise. This thesis has touched upon other fields such as organisational psychology, strategy and management and social anthropology. Our research shows that factors across fields of expertise affect the software security culture. Consequently, we believe a broader spectre of the population should know a thing or two about security work.

Future Work

As we have researched factors that influence software security work in consulting firms, the findings may not be valid for all sectors based on this limited selection of participating consultancies. We have researched factors influencing software security culture in consulting firms with offices in Norway. From earlier, there has been little research on how security work is done in Norwegian companies. Thus, we believe it would be valuable to continue researching what factors influencing software security culture in public sector or in sectors of various product companies. A more extensive, but similar study, with a broader spectre of consultancies participating, could also reveal new factors that influence software security culture.

Further, we have not researched diversity in particular. We believe the results could have been different if we had considered factors such as gender and cultural background. Research on how such factors influence software security culture is not yet explored.

We have seen that an individual with a security role affects the security work. However, how a security role affects the psychological safety within a group is not researched. The lack of defined tasks and responsibility could affect the relationships within a team. What effect a security role has on a team is yet to be explored.

As the accountability of software security is unclear today, further research is required to conclude on what divisions or roles that are most equipped to be held accountable for the quality of security.

Security in the development process and how security requirements are prioritised against other considerations is still not sufficiently explored [25]. Further research on how to include security work in agile development processes could be valuable. In addition, research on how security requirements could be presented as functional requirements could also drive this field forward.

References

[1] A. Van Der Heijden, C. Broasca, and A. Serebrenik, “An empirical perspective on security challenges in large-scale agile software development,” *ESEM ’18: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pp. 1–5, 2018, DOI:10.1145/3239235.3267426.

[2] T. Nicolaysen, R. Sassoon, M. B. Line, and M. G. Jaatun, “Agile Software Development: The Straight and Narrow Path to Secure Software?” *International Journal of Secure Software Engineering*, vol. 1, no. 3, 2010, DOI:10.4018/jsse.2010070105.

[3] M. G. Jaatun, D. S. Cruzes, K. Bernsmed, I. A. Tøndel, and L. Røstad, “Software Security Maturity in Public Organisations,” *Information Security: Lecture Notes in Computer Science*, vol. 9290, 2015, DOI:10.1007/978-3-319-23318-5_7.

[4] H. A. Kruger and W. D. Kearney, “A prototype for assessing information security awareness,” *Computers and Security*, vol. 25, no. 4, 2006, DOI:10.1016/j.cose.2006.02.008.

[5] K. Beck, M. Beedle, A. v. Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, “Manifesto for Agile Software Development,” 2001, accessed: 04-05-2021. [Online]. Available: agilemanifesto.org

[6] A. Poller, L. Kocksch, S. Türpe, F. A. Epp, and K. Kinder-Kurlanda, “Can security become a routine? A study of Organizational change in an agile software development group,” *Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW*, 2017, DOI:10.1145/2998181.2998191.

[7] S. Xiao, J. Witschey, and E. Murphy-Hill, “Social influences on secure development tool adoption: Why security tools spread,” *Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW*, 2014, DOI:10.1145/2531602.2531722.

[8] M. S. Mark, C. Tømte, T. Naess, and T. Røsdal, “IKT-sikkerhetskompetanse i arbeidslivet - behov og tilbud,” Tech. Rep., 2017.

[9] S. Gausen, T. Knutsen, S. Ruud, and T. Strandberg, “Stortinget utsatt for IT-angrep: «Et angrep på vårt demokrati».” 2021, accessed: 18-05-2021. [Online]. Available: https://www.aftenposten.no/norge/i/PRnGRX/stortinget-utsatt-for-it-angrep-et-angrep-paa-vaart-demokrati

[10] E. Knudsen, “Kritiske sårbarheter i Adobes programvare utnyttes allerede av hackere,” accessed: 18-05-2021. [Online]. Available: https://www.digi.no/artikler/kritiske-sarbarheter-i-adobes-programvare-utnyttes-allerede-av-hackere/510100

[11] A. v. d. Stock, B. Glas, N. Smithline, and T. Gigler, “OWASP Top Ten,” 2021, accessed: 24-05-2021. [Online]. Available: https://owasp.org/www-project-top-ten/

[12] “Confidentiality, Integrity and Availability - The CIA Triad - CertMike,” accessed: 29-05-2021. [Online]. Available: https://www.certmike.com/confidentiality-integrity-and-availability-the-cia-triad/

[13] “Reasons behind cyber attacks | nibusinessinfo.co.uk,” accessed: 29-05-2021. [Online]. Available: https://www.nibusinessinfo.co.uk/content/reasons-behind-cyber-attacks

[14] “13-åringen til kode24: - Slik fant jeg de 35.000 passordene - Kode24,” 2019, accessed: 07-06-2021. [Online]. Available: https://www.kode24.no/kodenytt/13-aringen-til-kode24---slik-fant-jeg-de-35000-passordene/70983531

[15] “MyFitnessPal Breach: Learn About MyFitnessPal Hack - IDStrong,” accessed: 29-05-2021. [Online]. Available: https://www.idstrong.com/sentinel/myfitnesspal-data-breach/

[16] “At least 100,000 groups in 150 countries hit by ransomware | Inquirer Technology,” accessed: 29-05-2021. [Online]. Available: https://technology.inquirer.net/62619/least-100000-groups-150-countries-hit-ransomware

[17] S. Bartsch, “Practitioners’ perspectives on security in agile development,” *2011 Sixth International Conference on Availability, Reliability and Security*, pp. 479–484, 2011, DOI:10.1109/ARES.2011.82.

[18] S. F. Wen, M. Kianpour, and S. Kowalski, “An empirical study of security culture in open source software communities,” *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2019*, 2019, DOI:10.1145/3341161.3343520.

[19] J. Xie, H. R. Lipford, and B. Chu, “Why do programmers make security errors?” *Proceedings - 2011 IEEE Symposium on Visual Languages and Human Centric Computing, VL/HCC 2011*, 2011, DOI:10.1109/VLHCC.2011.6070393.

[20] Y. Acar, C. Stransky, D. Wermke, C. Weir, M. L. Mazurek, and S. Fahl, “Developers Need Support, Too: A Survey of Security Advice for Software Developers,” *Proceedings - 2017 IEEE Cybersecurity Development Conference, SecDev 2017*, 2017, DOI:10.1109/SecDev.2017.17.

[21] H. Assal and S. Chiasson, ""Think secure from the beginning": A survey with software developers," *Conference on Human Factors in Computing Systems - Proceedings*, 2019, DOI:10.1145/3290605.3300519.

[22] I. A. Tøndel, M. G. Jaatun, D. S. Cruzes, and N. B. Moe, "Risk Centric Activities in Secure Software Development in Public Organisations," *International Journal of Secure Software Engineering*, vol. 8, no. 4, 2017, DOI:10.4018/ijsse.2017100101.

[23] S. Türpe, L. Kocksch, and A. Poller, "Penetration Tests a Turning Point in Security Practices? Organizational Challenges and Implications in a Software Development Team," Tech. Rep., 2016.

[24] K. Rindell, J. Ruohonen, J. Holvitie, S. Hyrynsalmi, and V. Leppänen, "Security in agile software development: A practitioner survey," *Information and Software Technology*, vol. 131, no. December 2018, 2021, DOI:10.1016/j.infsof.2020.106488.

[25] I. A. Tøndel and M. G. Jaatun, "Towards a Conceptual Framework for Security Requirements Work in Agile Software Development," *International Journal of Systems and Software Security and Protection*, vol. 11, no. 1, pp. 33–62, 2020, DOI:10.4018/ijsssp.2020010103.

[26] E. Terpstra, M. Daneva, and C. Wang, "Agile practitioners' understanding of security requirements: Insights from a grounded theory analysis," *Proceedings - 2017 IEEE 25th International Requirements Engineering Conference Workshops, REW 2017*, 2017, DOI:10.1109/REW.2017.54.

[27] Scrum.org, "What is Scrum?" accessed: 27-05-2021. [Online]. Available: https://www.scrum.org/resources/what-is-scrum

[28] "Agile Scrum Roles | Atlassian," 2021, accessed: 27-05-2021. [Online]. Available: https://www.atlassian.com/agile/scrum/roles

[29] I. M. S. Torgersen, "Kjære teknologistudent - Har du hørt uttrykket "leveransemodell"?" 2020, accessed: 27-05-2021. [Online]. Available: https://www.linkedin.com/pulse/kj\T1\aere-teknologistudent-har-du-h\T1\ort-uttrykket-strand-torgersen/?articleId=6682603513693949952

[30] L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, "A survey of DevOps concepts and challenges," *ACM Computing Surveys*, vol. 52, no. 6, 2019, DOI:10.1145/3359981.

[31] R. N. Rajapakse, M. Zahedi, M. A. Babar, and H. Shen, "Challenges and solutions when adopting DevSecOps: A systematic review," 2021.

[32] L. Bass, I. Weber, and L. Zhu, *DevOps: A Software Architect's Perspective.* Pearson Education, Inc., 2015.

[33] H. Myrbakken and R. Colomo-Palacios, "DevSecOps: A multivocal literature review," *Communications in Computer and Information Science*, vol. 770, no. October, 2017, DOI:10.1007/978-3-319-67383-7_2.

[34] A. Cockburn and J. Highsmith, "Agile Software Development: The People Factor," *Computer*, 2001, DOI:10.1109/2.963450.

[35] Y. Shin, M. Kim, J. N. Choi, and S.-H. Lee, "Does Team Culture Matter? Roles of Team Culture and Collective Regulatory Focus in Team Task and Creative Performance," *Group & Organization Management*, vol. 41, no. 2, 4 2016, DOI:10.1177/1059601115584998.

[36] E. H. Schein, "Organizational Culture," *American Psychologist*, vol. 45, no. 2, 1990, DOI:10.1037/0003-066X.45.2.109.

[37] E. H. Schein, *Organizational culture and leadership*, 2010, DOI:10.1016/j.sbspro.2011.12.156.

[38] W. A. Kahn, "Psychological conditions of personal engagement and disengagement at work," *Academy of Management Journal*, vol. 33, no. 4, 1990, DOI:10.5465/256287.

[39] B. Adkins and D. Caldwell, "Firm or subgroup culture: Where does fitting in matter most?" *Journal of Organizational Behavior*, vol. 25, no. 8, 2004, DOI:10.1002/job.291.

[40] C. Benjaminsen and N. B. Moe, "Slik jobber de aller beste teamene," 2019, accessed: 05-05-2021. [Online]. Available: https://www.sintef.no/siste-nytt/2019/slik-jobber-de-aller-beste-teamene/

[41] V. Stray, N. B. Moe, and R. Hoda, "Autonomous agile teams: Challenges and future directions for research," *ACM International Conference Proceeding Series*, vol. Part F1477, 2016, DOI:10.1145/3234152.3234182.

[42] T. Chau and F. Maurer, "Knowledge sharing in agile software teams," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3075, 2004, DOI:10.1007/978-3-540-25967-1_12.

[43] C. Benjaminsen and N. B. Moe, "Hjemmekontor gjør oss ikke mindre produktive," 2020, accessed: 05-05-2021. [Online]. Available: https://www.sintef.no/siste-nytt/2020/hjemmekontor-gjor-oss-ikke-mindre-produktive/

[44] M. A. Campion, A. A. Fink, B. J. Ruggeberg, L. Carr, G. M. Phillips, and R. B. Odman, "Doing competencies well: Best practices in competency modeling," *Personnel Psychology*, vol. 64, no. 1, 2011, DOI:10.1111/j.1744-6570.2010.01207.x.

[45] GitHub Inc., "About pull requests," 2018, accessed: 28-05-2021. [Online]. Available: https://help.github.com/articles/about-pull-requests/

[46] Sonatype, "State of the Software Supply Chain," 2019, accessed: 16-05-2021. [Online]. Available: https://www.sonatype.com/hubfs/SSC/2019SSC/SON_SSSC-Report-2019_jun16-DRAFT.pdf

[47] T. W. Thomas, M. Tabassum, B. Chu, and H. Lipford, “Security during application development: An application security expert perspective,” *Conference on Human Factors in Computing Systems - Proceedings*, vol. 2018-April, 2018, DOI:10.1145/3173574.3173836.

[48] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek, and C. Stransky, “You Get Where You’re Looking for: The Impact of Information Sources on Code Security,” *Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016*, 2016, DOI:10.1109/SP.2016.25.

[49] N. Bodin and H. K. B. Golberg, “Security Guidelines in Practice for Software Development Teams Facing Great External Time Pressure,” 2020.

[50] M. G. Jaatun, “Modenhetsmodell for innebygd sikkerhet (BSIMM). Måling av programvaresikkerhetsaktiviteter i utviklingsorganisasjoner,” Tech. Rep. A27495, 2016.

[51] Synopsys, “What Is the BSIMM and How Does It Work? | Synopsis,” accessed: 30-05-2021. [Online]. Available: https://www.synopsys.com/glossary/what-is-bsimm.html

[52] “Software Security Metrics and Strategy | BSIMM,” accessed: 01-06-2021. [Online]. Available: https://www.bsimm.com/framework/governance/software-security-metrics-strategy.html

[53] G. McGraw, “Seven Touchpoints for Software Security,” 2006, accessed: 10-06-2021. [Online]. Available: http://www.swsec.com/resources/touchpoints/

[54] S. Deleersnyder and B. D. Win, “OWASP SAMM,” 2020, accessed: 30-05-2021. [Online]. Available: https://owasp.org/www-project-samm/

[55] “Application Security Verification Standard 4.0 Final,” Tech. Rep., 2019, accessed: 22-04-2021.

[56] “Microsoft Security Development Lifecycle,” Microsoft, accessed: 10-06-2021. [Online]. Available: https://www.microsoft.com/en-us/securityengineering/sdl

[57] D. Baca, M. Boldt, B. Carlsson, and A. Jacobsson, “A novel security-enhanced agile software development process applied in an industrial setting,” *Proceedings - 10th International Conference on Availability, Reliability and Security, ARES 2015*, 2015, DOI:10.1109/ARES.2015.45.

[58] Datatilsynet, “Software development with Data Protection by Design and by Default,” 2017, accessed: 22-04-2021. [Online]. Available: https://www.datatilsynet.no/en/about-privacy/virksomhetenes-plikter/innebygd-personvern/data-protection-by-design-and-by-default/

[59] C. Cummings, “The complete web application security testing checklist,” 2016, accessed: 28-05-2021. [Online]. Available: https://www.synopsys.com/blogs/software-security/complete-web-application-security-testing-checklist/

[60] Synopsys Editorial Team, “The Complete Application Security Checklist,” 2020, accessed: 28-05-2021. [Online]. Available: https://www.synopsys.com/blogs/software-security/complete-application-security-checklist/

[61] D. P. Gilliam, T. L. Wolfe, J. S. Sherif, and M. Bishop, “Software security checklist for the software life cycle,” *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE*, vol. 2003-Janua, 2003, DOI:10.1109/ENABL.2003.1231415.

[62] Datatilsynet, “Virksomhetenes plikter: Sjekkliste,” 2018, accessed: 28-05-2021. [Online]. Available: https://www.datatilsynet.no/rettigheter-og-plikter/virksomhetenes-plikter/sjekkliste/

[63] Amazon, “AWS Well-Achitected,” accessed: 28-05-2021. [Online]. Available: https://aws.amazon.com/architecture/well-architected/?wa-lens-whitepapers.sort-by=item.additionalFields.sortDate&wa-lens-whitepapers.sort-order=desc

[64] B. Fitzgerald and K. J. Stol, “Continuous software engineering: A roadmap and agenda,” *Journal of Systems and Software*, vol. 123, 2017, DOI:10.1016/j.jss.2015.06.063.

[65] C. Robson, “Real world research : a resource for users of social research methods in applied settings,” 2016.

[66] M. Hewstone and W. Stroebe, *An Introduction to Social Psychology*, 7th ed. John Wiley & Sons Ltd, 2020.

[67] European Commission, “Commission recommendation of 6 May 2003 concerning the definition of micro, small and medium-sized enterprises,” *Official Journal of the European Union*, May 2003.

[68] “Ordinal Scale: Definition, Level of Measurement and Examples | QuestionPro,” accessed: 27-05-2021. [Online]. Available: https://www.questionpro.com/blog/ordinal-scale/

[69] V. Stray, T. E. Fægri, and N. B. Moe, “Exploring norms in agile software teams,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10027 LNCS, no. 2, 2016, DOI:10.1007/978-3-319-49094-6_31.

[70] J. Cole and A. J. Martin, “Developing a winning sport team culture: organizational culture in theory and practice,” *Sport in Society*, vol. 21, no. 8, 2018, DOI:10.1080/17430437.2018.1442197.

[71] L. Kocksch, M. Korn, A. Poller, and S. Wagenknecht, “Caring for IT Security,” *Proceedings of the ACM on Human-Computer Interaction*, vol. 2, no. CSCW, 2018, DOI:10.1145/3274361.

[72] H. P. Sanghvi and M. S. Dahiya, “Cyber Reconnaissance: An Alarm before Cyber Attack,” *International Journal of Computer Applications*, vol. 63, no. 6, 2013, DOI:10.5120/10472-5202.

[73] R. Malan and D. Bredemeyer, “Functional requirements and use cases,” 2001.

[74] D. Geer, “Are Companies Actually Using Secure Development Life Cycles?” *Computer*, vol. 43, no. 6, pp. 12–16, 2010, DOI:10.1109/MC.2010.159.

Appendix A
Information Sheet to Participants

The information sheet, including letter of consent, was sent to the participants ahead of the interviews. The sheet is written in Norwegian. This sheet is strongly influenced by the template provided by NSD.

Vil du delta i masteroppgave om «Programvaresikkerhetskultur i utviklingsteam»?

Dette er et spørsmål til deg om å delta i et forskningsprosjekt hvor formålet er å identifisere suksessfaktorer til god programvaresikkerhetskultur i et utviklingsteam. I dette skrivet gir vi deg informasjon om målene for prosjektet og hva deltakelse vil innebære for deg.

Formål
Formålet med prosjektet er å finne ut hvordan utviklingsteam forholder seg til programvaresikkerhet og suksessfaktorer for god programvaresikkerhetskultur. Prosjektet er del av en masteroppgave i informasjonssikkerhet, og vi kommer derfor til å ha søkelys på sikkerhetsarbeid og personvernsutfordringer.

Forskningsspørsmål vi ønsker å besvare er:

1. Hvilke faktorer utgjør god kultur for arbeid med programvaresikkerhet i utviklingsteam i konsulentselskaper?

Hvem er ansvarlig for forskningsprosjektet?
Norges tekniske naturvitenskapelige universitet er ansvarlig for prosjektet.

Hvorfor får du spørsmål om å delta?
Du jobber med å utvikle programvare i et konsulentselskap.
Vi tar kontakt med totalt cirka ti personer som i jobben sin er involvert i utvikling av programvare.

Hva innebærer det for deg å delta?
Dersom du ønsker å delta i prosjektet innebærer det at du er med på en samtale (med mulig oppfølging) i løpet av våren 2021.

Vi ønsker å snakke med deg om hvordan du opplever kulturen i ulike team knyttet til programvaresikkerhet.

Dersom det er i orden for deg (frivillig) ønsker vi å ta opp intervjuet på video eller kun lydopptak for å unngå å måtte notere underveis.

Det er frivillig å delta
Det er frivillig å delta i prosjektet. Hvis du velger å delta, kan du når som helst trekke samtykket tilbake uten å oppgi noen grunn. Alle dine personopplysninger vil da bli slettet. Det vil ikke ha noen negative konsekvenser for deg hvis du ikke vil delta eller senere velger å trekke deg.

Ditt personvern – hvordan vi oppbevarer og bruker dine opplysninger
Vi vil bare bruke opplysningene om deg til formålene vi har fortalt om i dette skrivet. Vi behandler opplysningene konfidensielt og i samsvar med personvernregelverket.

De som vil ha tilgang på personopplysningene dine er Hanna-Kai Barstad Golberg og Nora Bodin (studentene som gjør masteroppgaven), samt veilederne våre Maria Bartnes og Robert Larsen.

Deltakerne eller bedriftene i prosjektet vil ikke kunne gjenkjennes i publikasjonen av masteroppgaven.

Hva skjer med opplysningene dine når vi avslutter forskningsprosjektet?
Opplysningene slettes når prosjektet avsluttes/oppgaven er godkjent, noe som etter planen er november 2021.

Alle opptak fra intervjuer vil bli slettet ved prosjektslutt.

Dine rettigheter
Så lenge du kan identifiseres i datamaterialet, har du rett til:
- innsyn i hvilke personopplysninger som er registrert om deg, og å få utlevert en kopi av opplysningene,
- å få rettet personopplysninger om deg,
- å få slettet personopplysninger om deg, og
- å sende klage til Datatilsynet om behandlingen av dine personopplysninger.

Hva gir oss rett til å behandle personopplysninger om deg?
Vi behandler opplysninger om deg basert på ditt samtykke.

På oppdrag fra NTNU – Norges Teknisk- Vitenskapelige Universitet har NSD – Norsk senter for forskningsdata AS vurdert at behandlingen av personopplysninger i dette prosjektet er i samsvar med personvernregelverket.

Hvor kan jeg finne ut mer?
Hvis du har spørsmål til studien, eller ønsker å benytte deg av dine rettigheter, ta kontakt med:
- NTNU ved Maria Bartnes, 45218102
- Vårt personvernombud: NTNU ved Thomas Helgesen, 93079038

Hvis du har spørsmål knyttet til NSD sin vurdering av prosjektet, kan du ta kontakt med:
- NSD – Norsk senter for forskningsdata AS på epost (personverntjenester@nsd.no) eller på telefon: 55 58 21 17.

Med vennlig hilsen

Maria Bartnes (Forsker/veileder)	Nora Bodin (Student)	Hanna-Kai Barstad Golberg (Student)

Samtykkeerklæring

Jeg har mottatt og forstått informasjon om prosjektet «Masteroppgave: Programvaresikkerhetskultur i utviklingsteam», og har fått anledning til å stille spørsmål. Jeg samtykker til:

- ☐ å delta i samtale
- ☐ at opptak av samtale blir lagret til prosjektslutt (valgfri)

Jeg samtykker til at mine opplysninger behandles frem til prosjektet er avsluttet

----
(Signert av prosjektdeltaker, dato)

Appendix B
Interview Guide

Due to the semi-structured interviews, we have not asked all interviewees the same standardised questions. The interview guide is adapted to the interviewees career, role and the natural development during the conversation. The interview guide is written in Norwegian.

Intervjuguide til masteroppgave

Oppvarming:

- Fortell litt om deg selv og hvilke arbeidsoppgaver du har.
- Hvor bevisst vil du si at du er på programvaresikkerhet i arbeidshverdagen?
- Hva legger du i det å inkludere sikkerhet i utviklingsprosessen?

Hoveddel:

- Hvilke typer sikkerhetsaktiviteter arrangeres i løpet av et semester/prosjekt? (Her mener vi alle aktiviteter som er med bevisstgjøring av "sikker programvareutvikling)
- Hvordan legger dere til rette for kunnskapspåfyll knyttet til programvaresikkerhet?
- Hvordan jobber du for å inkludere nyansatte i bedriftens/ teamets programvaresikkerhetskultur?
- I hvilken grad opplever du at kollegaer er interessert i å drøfte en sikkerhetsproblemstilling?
- Hvordan påvirker kundenes sikkerhetskultur deres arbeid med sikkerhet i programvaren?
- Hva tenker du er viktige faktorer for god programvaresikkerhetskultur?

Avslutning:

- Hvilke sikkerhetsaktiviteter skulle det vært mer av, eller lettere tilgjengelig i din bedrift?
- Hva innenfor programvaresikkerhet skulle du ønske at du kunne mer om?
  - Hva skal til for at du lærer deg det?
- Er det noe mer du vil legge til?

Appendix C
Follow-Up Questions

One and a half month after we finished the interviews, we sent follow-up questions to the interviewees by e-mail. These questions are written in Norwegian.

Oppfølgingsspørsmål

1. **Bruk av ressurser**
    a. Har du kjennskap til noen veiledere for sikkerhetsarbeid (f. eks. BSIMM, Microsoft SDL, OWASP SAMM, Seven touchpoints)?
    b. Hvilke har du brukt i ditt arbeid som utvikler? Fortell også hvordan du har brukt dem.
    c. Hvis du har brukt noen sjekklister i forbindelse med utvikling, hvilke har du brukt, og hvordan brukte du dem?
    d. Hvordan fikk du kjennskap til ressursene du brukte?
    e. Vet du om andre i bedriften har brukt veiledere eller sjekklister i prosjekt?
    f. Er det noen ressurser du skulle ønske fantes?
    g. Noe annet du vil tilføye?

2. **Utdanningsbakgrunn**
    a. Hvilken studiebakgrunn har du? (Studie og skole, evt. om du er selvlært)
    b. Hadde du noen fag med hovedfokus på informasjonssikkerhet i løpet av studiet? Var det obligatorisk eller valgfag?

3. **Interesse**
    a. På en skala fra 1 til 5, hvor 1 er svært uinteressert og 5 er svært interessert: Hvor interessert vil du si at du er i programvaresikkerhet?

Nora Bodin and Hanna Kai Barstad Golberg

Software Security Culture in Development Teams: An Empirical Study

NTNU

Kunnskap for en bedre verden