NTNU
Norwegian University of
Science and Technology

TDT4501 Specialization Project

Autumn 2018

DevSecOps: How to Facilitate Programmers in Writing Secure Code

Written by Nora Tomas

Supervisor: Jingyue Li

Monday 19th December, 2018

Contents

List of Figures

1 Introduction

1.1 Research Question

Information security is becoming increasingly important as cyber crime is accelerating. In the first half of 2018 945 data breaches caused 4,5 billion data records to be compromised worldwide [1]. The number of lost, stolen or compromised records increased by 133 percent compared to 2017 [1]. Breaches happen for a variety of reasons, one of these is vulnerabilities found in code. British Airways was recently hacked because of 22 lines of insecure JavaScript code [2]. Programmers are central agents in creating secure applications. However, external factors such as time pressure and high workload can cause programmers to neglect security [3].

Additionally, software engineering and software security have historically worked in separate silos [4]. When software engineers create requirements it is often assumed that an application will only be used as intended. No identification of risks is performed. Code written for one environment is frequently deployed in a different environment, and other insecure processes take place [4]. It could be beneficial to shift security to the left in the software development process. Creating late fixes is more costly than designing a system with security in mind [5]. Unifying the silos of software engineering and security could facilitate programmers to write more secure code.

One research paper has attempted to solve this problem by proposing a unified model for software engineering and security [4]. However, research on the divide between software engineering and security is scarce. Some investigation has been done on the term "DevSecOps" which refers to the unification of Development, Security and Operations. A literature review has been written about DevSecOps where it was concluded that almost no white-paper literature on the topic is available [6]. The literature review used grey paper literature to define the meaning of DevSecOps and to check if DevSecOps is an increasing trend.

Previous research has also attempted to provide tools that programmers can use for static security analysis [7]. The idea behind static analysis is to warn programmers about

vulnerabilities in an early stage of the development process. However, there are not many investigations on the actual effect of these tools.

Earlier research has failed to investigate how a combination of culture-changes and tools can enable programmers to write secure code. This is the idea behind DevOps (or DevSecOps): To combine the four pillars of Culture, Automation, Measurement and Sharing to break down silos.

This report aims to study the state-of-the art of how organizations can use DevSecOps to manage security. More specifically, what can be done to increase security with roots in each of the four pillars mentioned previously.

The research questions is as follows.

- RQ1: How can DevSecOps enable programmers in writing code fast and secure?
  - RQ1.1: What is the state of art of DevSecOps in relation to the four pillars of DevOps (culture, automation, measurements and sharing)?

The research question will be answered by investigating white and grey paper literature that deal with DevOps and DevSecOps. Further, qualitative interviews will be conducted to see which DevSecOps practices companies already have implemented.

2 Background

2.1 Software Industry Silos

Historically, there has been a divide between development and operations [8]. The job of developers is to implement new features while operations keeps the application running reliably. Delivering functionality and making sure the system is stable can be conflicting missions. In the DevOps Handbook the authors write that "*Every IT organization has two opposing goals*"[9]. The quote refers to the pull between creating new features and keeping systems stable.

A case study that shows the development-operation divide can be seen by examining Target prior to 2012 [9]. If a development team wanted to build new functionality they did not have access to APIs but had to manually build integrations to get the data they needed. These new integrations would often break the system because the system was tightly coupled. Operations would need to manage the servers and make sure the system did not break. QA teams would spend 3 to 6 months testing new features. This means that development would often stay in queue while other teams were finishing their tasks. Operations and QA would have an increasingly difficult job the more functionality development wanted to create. Development, in turn, had to wait in order to produce functionality. From the story of Target it is apparent that these teams have conflicting goals and work in separate silos. Image 2.1 shows the traditional team division in software development.

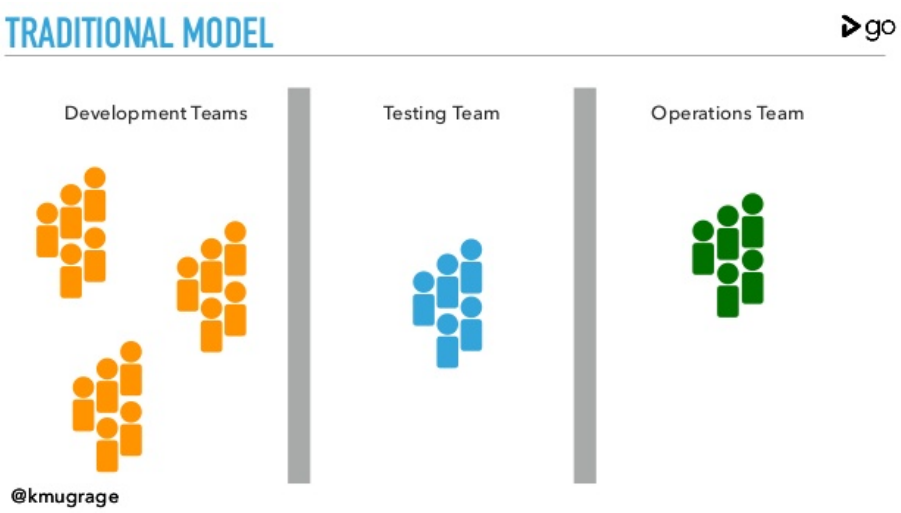

Figure 2.1: Developers, QA and operations in information silos [10]

2.1.1 Security Silos

Operations, QA and development are not the only teams in separate silos. Security can also become isolated. Viktorija Almazova emphasized during AppSec EU 2017 that traditionally developers and security teams were far away from one another [11]. Development teams would push code to production. A security professional would then visit the development team once every few months and stop code from going into production because of a security review. Programmers would feel like the security professionals were just breaking their stuff.

Programmers also seem to lack security training. DevSecOps global skills survey shows that developers don't have the formal education and skills they need to produce secure software [12]. While 80% of the respondents hold a bachelor or master degree, 70% said that their security education did not meet their current position requirements. Programmers are the ones who ultimately need to write code to close vulnerabilities, but somehow security has been made to belong in a separate category.

At Devopsdays Oslo 2018 we proposed the discussion topic "*How to create a security culture*". Several developers and security professionals were present. A key point in the discussion was that tools alone don't solve problems. Security professionals shared how they would often provide developers with new tools for security testing. However, if the tools slowed down the build too much some developers would turn the tools off. The importance of implementing a culture change was emphasized. It is essential that developers themselves see the importance of writing secure code. Creating a security culture is difficult if there is a wall between the development and security team. Our qualitative interviews show that this divide is present. The results are presented in

section 5. DevOps philosophy is about breaking down these walls with a combination of principles. The principles and what DevOps is are presented in the next section.

2.2 Terminology of DevOps and DevSecOps

The purpose of DevOps is to unify software development and operations in order to enable companies to quickly deliver services [13]. However, the term can be challenging to define because different companies have different ways of doing DevOps. Therefore, an initial, short literature study was conducted to check how DevOps is defined.

The used databases were Google Scholar and Oria. Search terms were ("DevOps" OR "DevSecOps" OR "SecDevOps" OR "DevOpsSec") AND ("definition" OR "characteristics"). The search was split into two parts where each part contained permutations of DevOps or DevSecOps and either the word "definition" or "characteristic". The inclusion and exclusion criteria were as follows:

- Inclusion criteria
  - Research Articles
  - Published after 2014
  - Explicitly define DevOps or DevSecOps
  - Found in first three pages of search results
- Exclusion Criteria
  - Books/blogs or other grey paper literature
  - Published before 2014

Several research articles point out that the term DevOps is not yet defined. Roche writes that "*It is noted that there is no set definition of DevOps among software engineers*" [14]. Similarly, a literature study conducted by Smeds, Nybom and Porres concludes that "*providing a complete and clear definition of the term is a challenge*" [15]. Other articles also say that the term DevOps is not entirely clear [16] [17].

One definition points out end-to-end automation [16]. Several definitions emphasize that DevOps is about enhancing communication and merging of roles [18] [19] [6].

The DevOps literature review conducted by Smeds, Nybom and Porres points out that while there is no clear definition of DevOps, common DevOps characteristics have been specified [15]. These characteristics are culture, automation, measurement and sharing. Also refered to as CAMS, the four pillars of DevOps. These are the features that will be used to evaluate how to implement DevSecOps within an organization.

2.2.1 Definition of DevSecOps

There is controversy about if DevSecOps as a term should be used. While some define DevSecOps as the integration of security into DevOps [6], others claim that there is no reason to create a new term [20]. DevOps is about cooperation between development and operations, but is not limited to the two departments. In his Devopsdays presentations Ken Mugrage points out that "*If they are doing DevSecOps and you are doing DevOps, sorry, you still have to do security*" [10]. This makes sense as security, in theory, should be no different than other quality assurance.

Yet, DevSecOps as a term has been used by security professionals [12]. Additionally, n their literature review Myrbakken and Colomo-Palacios argue that security practices can not keep up with DevOps agility and speed. They present DevOps and security as two things that don't naturally go together [6].

In this report, DevSecOps will be used for clarity to emphasize the focus on security. However, most sources agree that security is not excluded from DevOps [9]. Similarly, business requirements and data are not excluded from DevOps. If DevOps is seen as only involving development and operations, and excluding security, business requirements and etc. then a lot of different terms will appear. Such as DevSecOps, BizDevOps, DataDevOps and so on.

2.3 History of DevOps

As previously mentioned, historically there has been a divide between development and operations [8]. The reason for this is that developments job is to implement new features, while operations need to make sure applications are running reliably. Adding new functionality and making sure the application is stable are conflicting missions. DevOps is a response to this divide and has developed from other movements such as lean and agile. This section gives an overview of DevOps history.

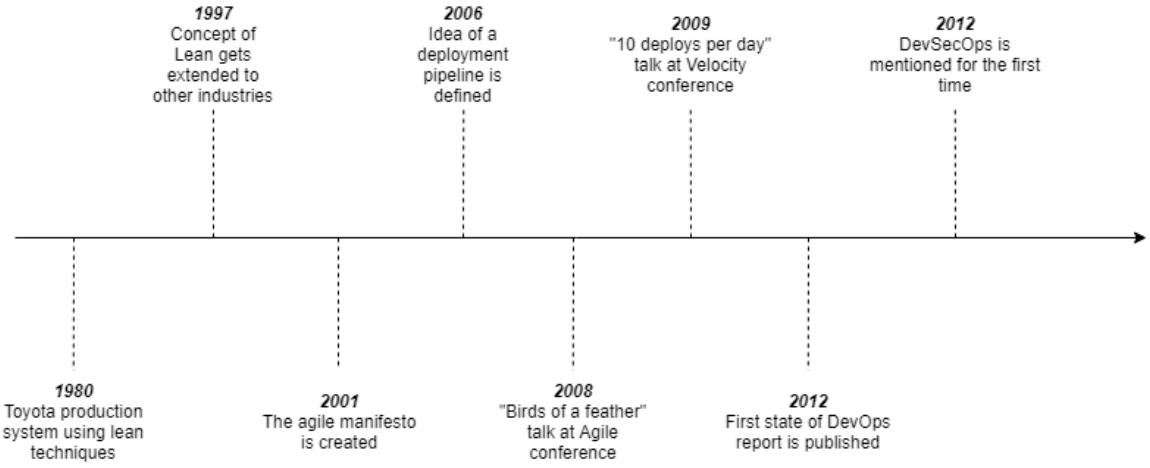

Figure 2.2: DevOps Timeline

2.3.1 The lean movement

The Toyota production system was using techniques such as value stream mapping and Kanaban boards in the 1980s [9]. In 1997 the concepts of lean were extended to other industries, such as service and health-care. Lean principles are about minimizing batch sizes in order to decrease lead time. In IT minimizing batch sizes means using iterative methods instead of a waterfall model. Small batches of software are developed and iterated upon, instead of first planning the entire system, then producing it, testing and finally deploying it. Lean is a central part of DevOps as DevOps methodology encourages iterations and frequent deployments.

2.3.2 The agile manifesto

In 2001 the agile manifesto was created. One key principle is to "deliver working software frequently, from a couple of months to a couple of weeks, with a preference to the shorter timescale". As in lean, small batch sizes is a key concept of the agile manifesto [21]. Deployments are now happening even more frequently than weekly. Companies such as Amazon are deploying software every minute [9].

2.3.3 Agile infrastructure and velocity movement

In 2008 Patrick Debois and Andrew Shafer held a session at the Agile conference named "birds of a feather". The idea of the session was to apply agile principles to not just code, but also infrastructure [9]. In 2009 at the Velocity conference John Allspaw and Paul Hammond had a talk named "10 Deploys per Day: Dev and Ops cooperation at Flickir" where they described how Dev and Ops cooperate closely and share goals [22].

These talks ignited two key ideas behind DevOps. The first one is Infrastructure as Code (IoC) where infrastructure is treated as code and lean principles are applied. The second is the collaboration between Dev and Ops.

2.3.4 The continuous delivery movement

The idea of continuous delivery defined the concept of a "deployment pipeline". The purpose of a deployment pipeline is to ensure that the code always is in a deployable state. The idea was first presented in 2006 by Jez Humble and David Farely. In 2009 the idea was also developed by Tim Fitz in his blog-post named "Continuous Deployment" [23].

2.3.5 State of DevOps reports

The first "State of DevOps" report was published in 2012 and was the first systematic collection of data about DevOps. One of the key findings of the 2013 report was that DevOps adaptation was accelerating with a 26% increase in adoption rate compared to previous years [24]. The report also mentioned that high-performing organizations enabled by DevOps deploy code 30 times more often than low-performing organizations. These reports were the first to quantify the effects of DevOps.

3 Related Work

3.1 Characteristics of DevOps

As explored in section 2 a clear definition of DevOps has yet to be agreed upon in research. However, there are features that characterize DevOps. These are divided into categories of Culture, Automation, Measurment and Sharing (CAMS) [15]. CAMS will be described in this section.

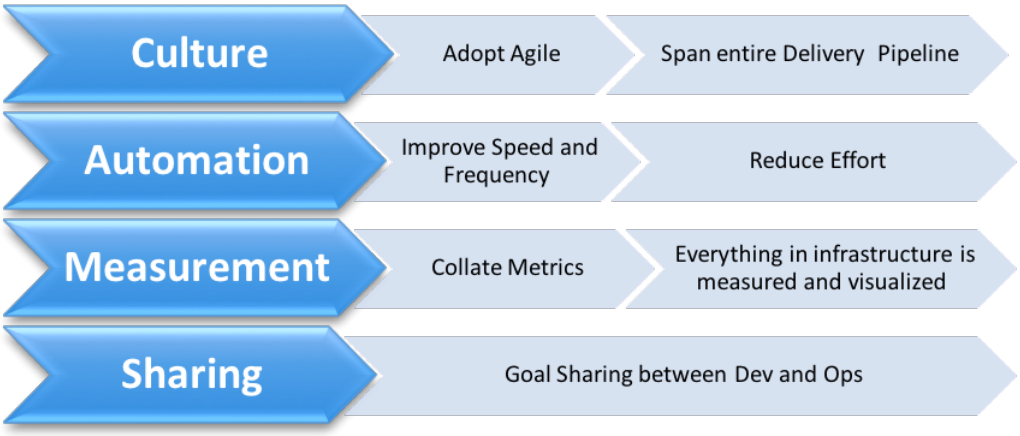

Figure 3.1: CAMS: Culture, Measurement, Sharing and Automation [15]

3.2 Culture

Culture is an important part of DevOps. It is difficult to have self-organizing teams if people are working on systems that are hard to build and deploy. It is also not easy to build and automate deploy processes if an architecture is hard to test [10]. DevOps culture is also about increasing collaboration between different roles such as development, operations, security and etc [25]. An attitude of shared responsibility is important to achieve this collaboration. If ,for example, developers don't feel responsible for security it is likely that they will not prioritize it.

3.2.1 Product Teams

In figure 2.1 we presented the traditional model of diving roles in a software company. Part of doing DevOps is stepping away from the traditional team division and move on

to product teams.

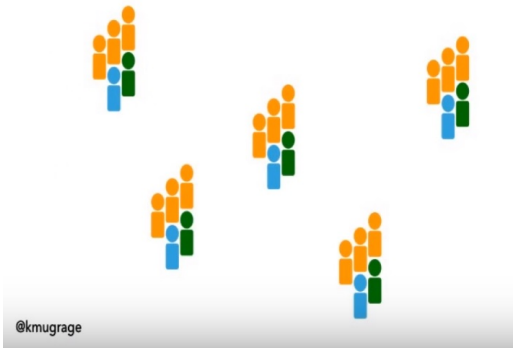

Figure 3.2: Product Teams. Each team has members from both QA, development operations and others [10]

Each product team works on a specific part of the system and is able to independently deploy this part. Each product team consist of employees from development, QA, security and etc. However, making product teams can become impossible if the original system has a monolithic architecture [10]. A monolithic system is composed all in one piece. Its components are interconnected and interdependent rather than loosely coupled [26]. With a monolith when one product team makes a change it will affect what other teams are doing.

In order to enable product teams microservice architectures can be used. In a microservice architecture each element of functionality is in a separate service [10]. It scaled by distributing these services accross servers. A microservice architecture also usually means that each service (that a product team works on) is independently deployable.

3.3 Automation

When someone think about DevOps what they usually consider is the pillar of automation. The title "DevOps engineer" often means working with automation [10].

The State of DevOps Report in 2016 measured how frequently code is deployed [27]. In order to quickly deploy code it is beneficial to adapt continuous practices such as continuous integration, delivery and deployment. The following sections describe these practices.

3.3.1 Continuous Integration

Continuous integration is when members of a team integrate and merge development work frequently, such as multiple times a day [28]. This practice includes automatic software testing and building. Meaning that tests can be written and then automatically run on the continuous integration server. These tests can also be security-related. It is important to note that simply having a CI server is not enough to say that a team is doing CI. A team needs to have practices for how they will do continuous integration. Ken Mugrage gives an example of CI that not only includes automatic testing, but that every developer commits daily and build can be repaired within 10 minutes [10].Image 3.3 shows an example of this process.

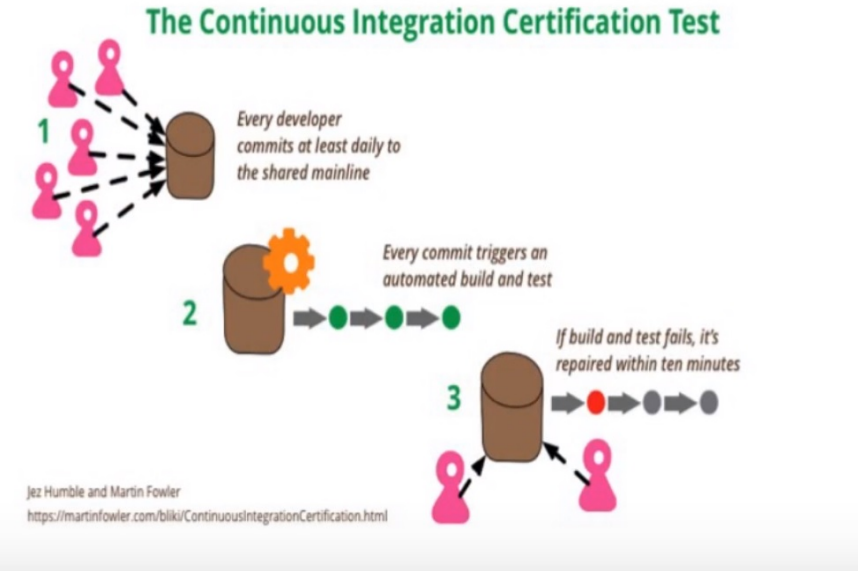

Figure 3.3: Example of doing continuous integration [10]

3.3.2 Continuous Delivery

Continuous delivery is an extension of CI to make sure that changes can be released to customers quickly. In addition to having automated testing the entire release process is also automated [29]. It means that the application can be deployed at any time by the click of a button.

3.3.3 Continuous Deployment

Continuous Deployment goes one step further again and continuously deploys the application [15]. An important distinction between CDE and CD is that there is no manual step involved in CD. As soon as developers commit a change it is deployed through the pipeline. In CDE, on the other hand, not all changes need to be deployed.

Image 3.4 illustrates the relationship between CI, CDE and CD.

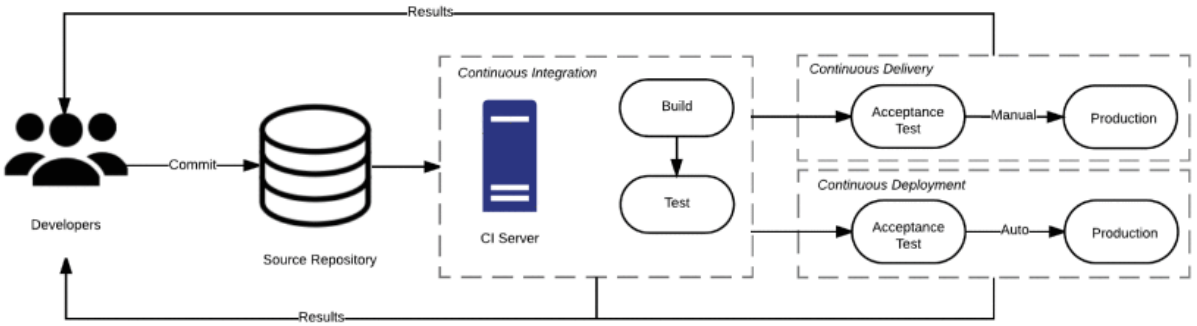

Figure 3.4: Relationship between CI, CDE and CD [28]

3.3.4 Automation Tools and Infrastructure

In DevOps practices appropriate tools and infrastructure are necessary to automate the code delivery process [9]. The success of adapting continuous releases heavily relies on deployment pipelines.

Build tools are used for tasks such as compiling code and managing dependencies. One purpose of build tools is to check that the newly created code of one developer will not negatively affect the environment it will be integrated into [30]. For example, Maven (a popular build tool for Java) can be used to keep track of what external libraries are needed in order to run the project. When a developer integrates new code they can make sure that all external dependencies are present.

Continuous integration tools also go under the build category. These tools can be used to run tests before code is integrated into the shared environment. Image 3.5 shows an overview of automation tools for DevOps.

Automation tools for DevOps.

Tool	DevOps phase	Tool type	Configuration format	Language	License
Ant	Build	Build	XML	Java	Apache
Maven	Build	Build	XML	Java	Apache
Rake	Build	Build	Ruby	Ruby	MIT
Gradle	Build	Build	Based on Groovy	Java and a Groovy-based domain-specific language (DSL)	Apache
Jenkins	Build	Continuous integration	UI	Java	MIT
TeamCity	Build	Continuous integration	UI	Java	Commercial
Bamboo	Build	Continuous integration	UI	Java	Commercial
Puppet	Deployment	Configuration management	DSL similar to JSON (JavaScript Object Notation)	Ruby	Apache
Chef	Deployment	Configuration management	Ruby-based DSL	Ruby	Apache
Ansible	Deployment	Configuration management	YAML (YAML Ain't Markup Language)	Python	GPL (GNU General Public License)
Loggly	Operations	Logging	—	Cloud based	Commercial
Graylog	Operations	Logging	—	Java	Open source
Nagios	Operations	Monitoring	—	C	Open source and GPL
New Relic	Operations	Monitoring	—	—	Commercial
Cacti	Operations	Monitoring	—	PHP	GPL

Figure 3.5: Automation tools often used in a DevOps practice [16]

3.4 Measurement

Technology work happens within complex systems with a high risk of failures. Often times, faults are not detected before they happen and cause system failures [9]. A way to attempt to prevent this is to implement monitoring and measure system metrics. Table 3.1, 3.2 and 3.3 give an overview of possible metrics to measure divided into three distinct categories.

The first category is basic application and infrastructure health. These measurements track application performance and are presented in Table 2.1 [31].

The second category are measurements for reliability and system health. These are shown in table 3.2.

Metric	Description
Apdex	Measures CPU usage. In on-premise applications high CPU usage could lead to application failure. While low CPU usage in the cloud could mean that paid-for resources are not entirely utilized.
Average response time	Measures the average time it takes for an application to process a transaction. For example the average time a file download takes compared to a login request.
CPU percentage usage	Measures CPU usage. In on-premise applications high CPU usage could lead to application failure. While low CPU usage in the cloud could mean that paid-for resources are not entirely utilized.
Error rates	Measures the percentage of transactions that result in an error during a specific time.If an application handles 1000 transactions and 50 of them have unhandled exceptions, the error rate is 50/1000 = 5
Load average	Measures the number of tasks that are waiting for the CPU. Can help in detecting an overloaded system or a processes that use too many resources.
Memory percentage usage	}Measures the amount of free memory bytes compared to used memory bytes. Similarly to CPU usage, too high memory usage can negatively affect performance in on-premise applications. Too low usage can mean cloud resources are not fully used.
Throughput	Throughput measures requests per minute made towards an application. This measurement can give information about how a new feature affects requests in the application as a whole.

Table 3.1: Measurments for basic application and infrastructure health

Metric	Description
Mean time to detection (MTTD)	This metric tracks the amount of time between the start of an issue and the detection of the issue, ideally at which point some action is taken.
Mean time to recovery (MTTR)	MTTR tracks the average time it takes to repair a failed component in a system, from the moment the failure is detected until the point at which the system is operating normally again
Service-level agreements (SLAs)	SLAs are the (sometimes legally binding) contract between the company and customers. They are a combination of all other measurments.
Service-level objectives (SLOs)	SLOs are goals the company sets for what the customers can expect from the system in terms of availability, performance, error rates, and anything else the two parties agree upon measuring

Table 3.2: Measurments for reliability and system health

Metric	Description
Code commits	If a team has too many or too few commits both can be a cause for concern. It is useful to track the number of commits.
Deployment time and deployment frequency	It is important to measure how often software is deployed and how often it can be deployed. However, it is important to remember that measurements drive behaviour. Frequent deployments with higher time to repair might not be ideal either.
Iteration length	This metric tracks the amount of time between development cycles during the execution of a project
Passed/failed unit tests	Tracking the number of unit tests that pass or fail during a development cycle can offer an indication of whether the team is writing well-designed code

Table 3.3: Measurements for team health

The final suggested standard measurements are for team health and performance.

3.5 Sharing

3.5.1 SECI Model

An aspect of overcoming the divide between different departments is to share knowledge. The SECI model provides four mechanisms of knowledge sharing between development and operations [32]. The four mechanisms are divided into two categories: tacit knowledge and explicit knowledge. Explicit knowledge is formal, systematic, easy to store and document. Tacit knowledge, on the other hand, is more personal and difficult to express in a systematic manner. Skills that require tacit knowledge, such as riding a bike, cannot simply be listed, but have hidden dimensions.

The four ways of sharing knowledge are presented in figure 3.6 and form the basis of DOKS (DevOps Knowledge Sharing Framework).

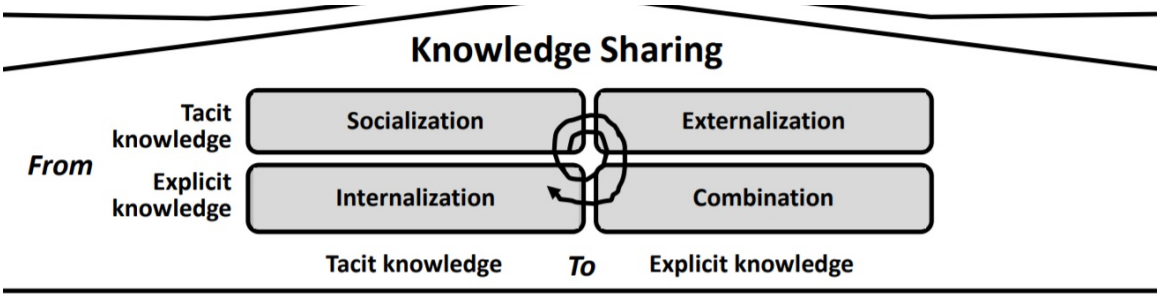

Figure 3.6: Developer knowledge sharing framework [32]

Socialization refers to face-to-face knowledge sharing between individuals where they can convert tacit knowledge into new tacit knowledge. *Externalization* happens when knowledge of individuals is made explicit and shared with a group. An example of where this happens are Scrum meetings. *Combination* happens when explicit knowledge is collected, processed and turned into new explicit knowledge. Finally, *internalization*

refers to when individuals reflect on the knowledge they have and start using it through learning by doing.

Knowledge sharing can be even more difficult for a company to implement than using the correct tools. The "DevOps Maturity Model report: trends and best practices in 2017" shows that companies are sharing tools, but not knowledge [33]. Most companies do not have dynamic knowledge sharing between departments, such as chat rooms or wikis. Knowledge is mostly shared only upon request.

3.5.2 Programmers and Security Knowledge

Developers often lack security knowledge [34]. In a recent article about Norwegian computer science students the lack of a software security focus was pointed out. The most re-tweeted quote of the 2011 OWASP summit was that "developers don't know anything about security" [35]. The same is the case in the United States where none of the top 10 computer science programs require security [35]. Sharing knowledge about security might therefore be one of the most important aspects of DevSecOps.

3.6 Previous work on DevSecOp

Myrbakken and Colomo-Palacios conducted a literature review on DevSecOps. They found out that not much white-paper literature on DevSecOps is available [6]. They studied grey paper articles and came to the conclusions presented in the following paragraphs.

According to the grey paper literature they conducted there seems to be somewhat of a consensus on how DevSecOps is defined. DevSecOps is seen as a necessary extension to DevOps. DevSecOps is about unifying security with operations and development.

DevSecOps can be characterized by the four pillars of DevOps defined by CAMS. *Culture* is the first pillar and in relation to security this means creating a company culture where security is considered to be everyone's responsibility. It also means involving the security team in the whole development process. In relation to tools this could mean having shared interfaces that both developers and security staff use to keep track of vulnerabilities. *Automation* involves using tools to automate security testing. One example of such a tool is Gauntlt. It can be used to perform routine security tasks, such as using nmap to check which security ports are open [36]. These automated security checks are referred to as "*Security as Code*" [36]. *Measurement* in DevSecOps refers to not only measuring system metrics as in regular DevOps, but also keeping logs of threats and

vulnerabilities. *Sharing* in DevSecOps refers to knowledge sharing between the security department and the rest of the organization. If the security team knows about challenges faced by developers and operations they can improve routines.

Overall, DevSecOps is a term that promotes shifting security to the left. Traditionally, security testing is done towards the end of the software life-cycle. However, fixing issues later in the process is more costly than making sure defects don't happen in the beginning [29].

The main benefits of DevSecOps were considered to be shifting security to the left, automating security and economical value. By shifting security to the left time does not have to be spent on adding extra security fixes to a backlog that has already been created. Adding unpredictable, extra backlog items could delay projects. Automating security allows processes to be predictable and scalable. It is also an important part of keeping up the necessary fast pace in DevOps. If tests are run automatically it also frees up more time for developers to write code instead of managing tests. Finally, as previously mentioned it is more costly to fix faults later in the development cycle. High performance organizations spend 22 percent less time on unplanned work and rework and therefore save time and money [27].

4 Method

4.1 Method

This section describes the methods used to collect data and to analyze the qualitative interviews. It also provides reasons to why the particular research method was chosen.

4.1.1 Research Design

The research strategy in this report is mixed and consists of an initial literature study combined with interviews. A literature study was first conducted to find out how DevOps, as a term, is defined. In the second step of the research employees from different companies were interviewed to find out about their experiences with DevSecOps. The interviews were semi-structured and conducted with the help of five informants from different companies.

Thematic analysis, which is a qualitative method, was used for data analysis. Codes were identified from the literature to facilitate the search for common themes in the interviews. A potential risk with this method is that the interview data can end up being too shallow. In this case the sample size of six participants will not be large enough to find statistically significant results. Therefore, open-ended questions were asked.

The research paradigm is interpretivism as there is no hypothesis to prove or refute. A central part of DevSecOps is culture. In order to understand how to break down silos between software engineering and software security it is necessary to understand human factors. The research question partially deals with the social context of a technical system which is also in line with interpretivism.

4.1.2 Reasoning behind research method

DevSecOps is a topic relatively new in white paper literature. Myrbakken and Colomo-Palacios conclude that there is almost no white-paper literature available on the topic of DevSecOps [6]. A search for DevSecOps (and iterations of the term) on Web of Science and Scopus only gives four results. Further, a detailed literature study on continuous integration, delivery and deployment, notes that only 2 out of 29 papers mention security

issues in the deployment pipeline [29]. On the other hand, there is a lot of grey-paper literature about DevSecOps available online. Myrbakken and Colomo-Palacios have already written a research which summarizes this grey-paper literature.

Therefore, we wanted to go a step further and pick a research method that would enable us to get new information about DevSecOps. We were curious about what companies already were doing in relation to security and DevOps and what challenges they were facing. A way of achieving this could have been to do a quantitative survey instead of semi-structured interviews. However, surveys on DevOps are already done frequently and on a large scale. An example being the "State of DevOps" reports [27]. An extensive survey on DevSecOps also already exists [12].

A survey would not necessarily provide the in-depth data that is needed. As there is no standard for implementing DevSecOps companies might use vastly different tools and have vastly different practices. A survey with standardized questions might only be able to capture differences that the survey creators have already thought about, and not add anything new. Additionally, DevSecOps is not just about automation and tools, but also sharing and culture. Getting a grasp on the security culture of a company is difficult to do without directly speaking to those involved. Further, an advantage of thematic analysis is that it allows flexibility in research and for multiple theories to be applied in the analysis. DevOps is a term with many layers, therefore it is beneficial to use an approach which allows for several theories to emerge.

4.1.3 Data Collection

Participants

Numerous tech companies were contacted and asked if they had staff available to participate in the interview. It was preferable to interview informants who either had experience with DevOps, security, or both of those. There were no specific requirements regarding company size or type. We got subjects from a variety of company sizes, from 30 employees to 42 000 4.1. Though all companies were private, one of the engineers was currently consulting a public office in Norway. Therefore, the interview gave insight on how the public sector does DevOps.

Interview Questions

One interview guide was created based on the four pillars of DevOps (CAMS: Culture, Automation, Measurement and Sharing). Even if a subject had no prior experience with DevOps they could still answer questions about security culture, automation and etc.

	Role	No. of Employees	Company Product	Type
Informant A	Platform Tech Lead	200	SaaS for Smart Cities	Private
Informant B	Software Engineer	11 000	Software Security Services	Private
Informant C	Software Engineer	1300	Consulting	Private, but in a public sector project
Informant D	Software Engineer	30	Platform for Web Sites	Private
Informant E	Software/Data Engineer	40	Mobile App	Private
Informant F	Security Architect/ Manager	44 000	Consulting	Private

Table 4.1: Informant Details

The interview guide purposely had few questions that were open-ended. Questions were either about the interview subject in general, DevOps or related to security in each CAMS-category. For example, in relation to Culture the subject would be asked **Do you have a security culture in your company?**. This question can be answered even if the subject has no particular DevOps experience. 4.2 shows the questions. Questions are either general or belong to a CAMS-category. Because the interviews are semi-structured questions were sometimes removed or additional ones were added. For example if an informant was asked "**Do you use any tools to automate security testing?**" and they reply with "**We do, but the tools have not been adequate**". Then we ask an additional question, such as **You mentioned the security automation testing tools are inadequate, why?**. An example of this can be seen in Interview A. If the informants, for example, said that they don't do DevOps then questions such as **What tools do you use in your DevOps practice?** are not included. Instead, informants are just asked about tools they use to automate security.

4.1.4 Ethical Implications

As this research involves interviewing people it is especially important to consider ethical implications. Guidelines given by the Norwegian National Research Ethics Committee were followed [37]. The committee writes that subjects should be informed about the purpose of the project and their participation in it. At the beginning of the interview informants were told about what their answers would be used for. Subjects were also told that they could withdraw from the experiment at any time and that they would be granted confidentiality.

Category	Question(s)
General	Can you tell me about your company and your role there?
General	Do you have a project where you say you are using DevOps?
General	What do you think are the benefits of using DevOps principles compared to not using it?
General	Did you have challenges with implementing DevOps on a project level? How did you address these challenges?
Automation	What tools do you use in your DevOps practice? Do you have tools to automate security?
Culture	Do you have a security culture in your company? If so, how do you ensure that is the case?
Measurements	Do you do any measurements? Do you measure security metrics?
Sharing	Do you have any processes for sharing security knowledge in your company?
General	Can you think of any additional tools or process that you would like to have in your company that would help you create even more secure programs?
General	What challenges related to DevOps do you think will come up in your company?

Table 4.2: Interview questions for subjects with DevOps experience

Anonymity goes a step further from confidentiality and would require that not even the interviewer knows the identity of the informants [37]. Only confidentiality was provided, as we conducted the interviews ourselves and therefore knew informant identity. The subjects consented to this. They were also told that no data apart from notes would be stored. No audio-recordings were created. Further, they were told that their identity and company name would not be disclosed. Two of the informants pointed out that this was important to them. One of them disclosed information about a data breach that otherwise would have been kept hidden. In order to respect their wishes we made sure no unauthorized persons will know who provided the data.

4.1.5 Data Analysis

Process Overview

The following steps were used in the data analysis method.

Step One First we quickly browsed through the transcripts and made notes about our first impressions. We did coding (or indexing) by labeling relevant words and sections. Information was labeled as relevant if it touched upon CAMS, or if it was about how informants interpreted what DevOps or DevSecOps means. We also looked at if the same information was repeated several places. We also placed more emphasis on information that conceptualizes underlying themes, rather than superficial facts. For example, even

though we did collect a list of the security tools utilized by the informants, we were more interested in if they found these tools useful.

Step Two There were 114 codes created in step one. These are listed in appendix .6. Therefore, we reduced the number by combing several codes together. Afterwards, the remaining codes were divided into categories.

Step Three In this step the categories were labeled. It was also important to find the connections between the categories. For example by looking at if some categories appeared more important than others. By doing this we created a hierarchy among the categories. The categories and are the final results of the study. In the discussion, relationships between the categories are analyzed.

5 Results

5.1 Identified Themes

Table 5.1 shows the final themes and the codes that went into creating them. In this section the themes are presented with quotes from the interviews. In the discussion-section it is examined what the relationship between the themes are, and how they answer the research question.

5.2 Automating Security

The results in this section are about general automation of security testing. It is about what tools companies that have automated security tests use. This section also briefly touches upon a reason to why setting up these tools can be challenging.

Informant B. who works as a developer for a security company, said that they used tools such as "*Snyk, Retire.js, Hakiri, PVS-Studio, Veracode, Coverity, Sonarqube and Checkmarx*". Informant F, who is a security architect, also mentioned Sonarqube and Checkmarx. He also added that they use OWASP dependency check for Java-projects. A brief description of each tool follows.

Themes	Codes
Automating security	Automating legacy systems, Automating security contradiction, Tool standards
Value of security automation tools	Tool-mindset link, Knowledge/Training more important, Security is abstract
Definitions of DevOps and contradictions	DevOps contradictions, DevOps meaning, DevOps mindset, DevOps is automation, No DevOps standard
Security Measurements	Security metrics, Measuring training, Monitoring time to repair, Monitoring pen-test fails
Developer-security divide	External security teams, Technical Debt, Team structure, Moving security to the left, Security-development merge, Security Champion, Developer Autonomy
Specialist vs. generalist knowledge	Lacking specialists, Final security responsibility
Reluctancy in prioritizing security	Security push-pull, Caring about security, Judging Risk, Management-security divide, Nobody responsible, everyone responsible security contradiction

Table 5.1: Identified Themes

Snyk shows which external libraries contain vulnerabilities [38]. When code is run on the CI-server Snyk gives a warning if new, insecure libraries are used. Snyk is also able to automatically update libraries to new versions if those versions contain less vulnerabilities. Snyk provides certain automatic fixes, such as the mentioned library version updating. However, if programmers already have written a lot of code with an insecure library, they might have to manually rewrite everything.

Retire.js works in a similar way to Snyk, but is created specifically for Javascript [39]. It identifies vulnerable Javascript libraries that are used in a project.

Hakari also looks at third-party dependencies for vulnerabilities [40]. It has an additional feature of analyzing code when a pull-request is created. Hakari does static analysis and checks if code contains security flaws. Hakri is mainly used for Ruby on Rails.

PVS-Studio detects security faults in C, C++ and C sharp code specifically [41]. It requires a pull-request to be made and does static analysis after the code has been written.

Veracode finds vulnerabilities at an earlier stage than the other tools [42]. It scans code in the IDE while a developer is working on it. Veracode supports a variety of programming languages. In addition to giving warnings Veracode shows the level of severity of a warning. In this way developers can prioritize the most serious security risks first.

Coverity is another tool that shows vulnerabilities after code has been created and a pull-request has been made [43]. Coverity is not only used for security, but does additional scans for detecting code-quality. An interesting aspect of Coverity is that it guarantees there will be no false positives in regular quality assurance. However, when it comes to security Coverity websites states that false positive might happen. An example of this is when storing sensitive data. Coverity is not able to determine which data has which level of confidentiality. Therefore, the programmer still has to make decisions about if the warning is valid or not.

Sonarqube provides quality assurance which includes security [44]. While the programmer is writing either C/C++, Java or Objective-C code he or she gets warnings on a separate dashboard. These warnings include potential security faults that are gathered from OWASP Top 10 and SANS Top 25. Sonarqube also states that false positives might occur and that the a human needs to make decisions about if a warning is legitimate or not.

Checkmarx offers different security tools for each stage in a DevOps-cycle [45]. Checkmarx offers a platform that lets developers, operations and security keep track of the current security status. Checkmarx also offers code scanning that can be run on only specific code-parts. Developers don't have to spend time on running security checks for the entire code-base every time.

OWASP dependency-check identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities [46]. The tool supports Java and .NET and mainly gives an overview of if vulnerable third-party libraries are used. This works similar to Snyk.

Several informants show signs of struggle in automating security. Informant D said that they never managed to automate any security nor create tests that permanently checked for vulnerabilities. Similarly, Informant F said that his project is still in the early phases of the automation process. He mentioned challenges in removing some old systems "*like production setting review boards*". He also explained that it is difficult to find developers that know how to use the security tools by saying "Almost all developers know some Docker but if you ask for people who know Checkmarx or something like that it is almost nobody". He says that this is due to a lack of standards. According to Informant F "*If everyone converged more towards 'this is what everyone is using' so that the tools become solid and easy to use that would be an ideal DevOps-scenario for me*". It appears as if automating security can become difficult because there is no one way to do it.

The fact that there is no standard on how to set up automated security testing is apparent from the "*automating security contradiction*" found in Interview C. Informant C says that "*It has been very important for us that everything is automated*". At the same time he says that they only have security tools to monitor logs. Automating security is therefore not seen as a part of "*everything*".

5.3 Value of Security Automation Tools

In this section, results regarding tool utility will be shown. If subjects used security testing automation tools we attempted to find out if they thought the tools were useful or not.

Informant A said that the tools they use "*have not been adequate*". He was referring to the in-house system that created a security configuration and tested if the environment follows this configuration. When asked about why he found the tools inadequate he responded with "*security is difficult. If you want a server, for example, security is more*

abstract. So there is no tool that does all security for you. What you have to test is very different depending on the environment you use. You can get AWS config, and you can do that with Puppet on a system level, but for an overall security management suite this is difficult to generalize". He adds that "*There is some way to go on vulnerability testing on the server level. Right now pen-testing is human-based. There are tools like Metasploit but a lot of configuration needs to be done by humans*".

It seems like Informant A views security as too broad to be taken care of with the available tools. This is contradictory to what the websites of certain security tool vendors such as Veracode promise. For example, Veracode websites say that "*With its powerful combination of automation, process and speed, Veracode seamlessly integrates application security into the software lifecycle, effectively eliminating vulnerabilities during the lowestcost point* " [42]. Veracode might give an impression of that their tool alone will fix most security issues.

Subject B was asked if the various tools he used slowed down work for a programmer. To this he replied that "*After a while you adapt the security mindset so it becomes less and less of an issue concerning speed*". Here, informant B made a connection between tools and a security mindset.

Informant B had an unexpected response when he was asked what new security tools or processes he would like to have. Instead of suggesting something new he said "*I don't know. I would actually not like to add layers, but perhaps remove some of them*". Other informants mentioned that they would like to have tools for "*known fingerprints of attack*" (in relation to ports by Informant C), and automating "*vulnerability testing on the server level*" (informant A). Informant D only mentioned new routines like "*patch Tuesdays*". It was surprising that informants did not have extensive wishes on what new tools they might like to have.

Subject F was asked if he believes that the automation tools they set up will assure sufficient security. To this he replied "*I think mainly it all should start with increasing knowledge and training developers in security. I would say this training and increasing knowledge is the most important part. These tools are not able to identify all mistakes and one problem is that they show a lot of false positives*".

Subject F believes that the tools are useful. He did mention that a benefit of DevOps was "*code scanning and automated security tests, dependency checks and so on*". This can only be achieved with the correct toolset. At the same time he thinks tools in themselves are not as useful without the accompanying knowledge.

5.4 Definitions of DevOps and contradictions

From our findings it appears as if Informants have differing views on what DevOps is. Informant A said that "*A key principle of DevOps is more involvement from the beginning of a project. This is important because developers are often not thinking about monitoring, alerting, scaling and other things that people on the platform team are thinking about. Before DevOps developers would come in on a Friday and tell operations to put the code into production for next week, but operations would often send it back because it could not be scaled*". Informant A emphasizes that DevOps is about cooperation between development and operations. He mainly views DevOps as a way to facilitate mutual understanding in the early stages of a project.

Informant D and E, on the other hand, thought of DevOps as an activity you could do by yourself. Informant D says that "*I implemented the DevOps part such as CI and CD systems*". Similarly, informant E said "*I apply DevOps all the time, I automate all recurring tasks*". What informant D and E point out is the opposite of cooperation. Rather, it appears as if they view DevOps to be synonymous with automation.

Informant A said that the term DevSecOps is challenging as it could lead to further extensions of DevOps such as DevSecTestOps. Informant F appears to indirectly agree that a further extension to the DevOps term is not necessary. This is because he sees DevOps as a movement where security already fits in. He says that an advantage of DevOps is "*that you can apply more mechanisms to improve security*". In this way he does not view DevOps as a hindrance for security. Rather, for him the purpose of DevOps is to enhance security. Therefore it could be discussed if terms such as "DevSecOps" are necessary when security already is part of DevOps.

Another contradiction found in the interviews were opposing opinions on if a standard for implementing DevOps exists. Informant A says that they had challenges in implementing DevOps because "*there is no standard*". Informant D says that a future challenge with DevOps could be that "*we could end up getting a unique setup for every project, for example everyone has their own AWS account where they set up Kybernetes on their own from scratch. So there is no common standard and you have a problem because if the few people who put up the stack leave nobody will have overview of what is happening*".

By talking to the informants it is clear that DevOps is not entirely defined or standardized. This can create challenges for companies. As informant D mentioned, a lack of standards could cause everyone to do DevOps differently and have different setups which might create documentation difficulties and confusion. Informant A implies the same

thing by saying that "*Many people take DevOps to mean 'development anarchy' so DevOps can cause developers to gain more freedom, but it is easy to go too far. Some people think DevOps means that developers can use any system or technology and not what has matured in the company. Especially with containers, developers have been enabled, so they see something works on their machine but don't understand why it does not work in Production*".

5.5 Security Measurements

Informant F worked at a company where they tracked several security measurements. He explains that "We have chosen this because we have seen that previously there has been a lot of security holes that happen because people are in a hurry. So some security flaws would appear as late as user acceptance-testing or right before things are about to go into production". He adds that "We try to check if we are introducing too many new features compared to technical debt that we get". Informant F believes that measurements are important because teams get constant feedback on how they are doing in regards to security. In this way they will not get surprised at a late stage in the project.

He suggests that the following security metrics can be tracked:

- The number of developers that have gone through security-training
- Number of mistakes in different security categories, such as OWASP top 10
- Time spent correcting mistakes in each category
- Which systems are affected by internal and external pen-testing

It is interesting to see that training is one of the security metrics Informant F suggests to keep track of.

The other informants were not keeping track of security metrics. Informant E kept track of other metrics. In his company there were dashboards for different services and warnings were given on Slack if anything critical happened. However, he did not mention any specific metrics in relation to security.

5.6 Developer-security divide

Informant A and B both identified a divide between developers and security professionals. Informant A started by saying that "*I feel like security is often a battle, it is never helpful*

it is always in the way, costs time and gives nothing" when asked about if his company has a security culture. He adds that security is a "*Constant battle between risk and cost*"

When asked about if developers are open to changes suggested by security professionals he responded with "*If you are in a room with people and the security guy says 'I don't like the way you wrote this', it doesn't matter how they say it, they are still judging your code. No matter how you phrase it they criticize the developers work*". Informant A identifies an important issue. Programmers spend time writing their code and then an external security professional comes in and tells them the code is not good enough.

Informant B, who works as a developer at a security company points out many of the same issues as Informant A. Early in the interview Informant B states that "*some security features can be frustrating for developers. For example you can not copy-paste code from the internet because the code you are writing has to be isolated and on a separate network*". In response to questions about security culture he goes on to saying "*Copy-pasting is just one example, but it seems like many of the things security teams suggest are just for show*". Here, informant B is talking about how developers and security have different ideas about what is truly necessary.

When informant B was asked about if they have a security culture in his company he said that "*There is a lot of push to create a security culture. We (developers) are always laughing about it*". He adds that "*We get a lot of propaganda*". After this comment he was asked a follow-up question of "*You are saying that developers sometimes laugh at the changes proposed by the security team. Is there a divide between software engineers and security teams in your company?*". Informant A responded with "*yes*" and went on to talking about the copy-pasting issue and other security-imposed restrictions. In the end he added that "*We would like to decide this with our own opinion. Programmers are good at security also, often times we have more knowledge than the security teams on some topics*". Informant B felt the need to unprompted point out that developers do have security knowledge. This could point towards that developers feel looked down on by security professionals. It is described how developers have "*no choice*" on these matters. Developers might feel like security teams take away their autonomy and do not sufficiently consider their opinions.

Informant F proposed a solution to this divide. He said that his company wanted to "*ensure that there are not separate people who work with security and security is not a separate activity. Instead it is part of development*". He says that having an external security team is "*something we have explicitly tried to avoid both on a project-level and in our company in general*". Basically, he identifies similar issues as informant A and

B in regards to external security teams. When asked about if developers think it is stressful to get feedback from the security team he answered "*the security team can be seen as a bottleneck. Also the security team can be seen as people who force changes on developers. With a separate security team developers often try to avoid them all together or just do the least possible amount of work for it to be approved*". Informant F believes that if developers are commanded to think about security by an external entity they will take shortcuts. Additionally, thinking about security too late is more expensive than considering it from the start. Informant F believes an ideal scenario would be one where security is "*like any other step in the software development process where it is taken care of from the beginning and thought about all the way*". He adds that "*Hopefully in some years we will not have to specify any roles but it will be natural to think about security just like you think about using Docker or Kubernetes in Azure or anything like that. So if you have a Kanbanboard that you consider security instead of doing it as a side-project". However he also says that 'for now things are still happening a bit in parallel'*".

Informant F believes that an ideal scenario would perhaps be one where nobody has a security-role, but rather security is seamlessly integrated into the process. However, he says that for now it is still useful to have so-called "*security champions*". Security champions are programmers who have the most security-training on the team and who care about security. They work as a bridge between programmers and other teams. Security champions are the ones who are called if a security issue is identified externally. In this way the team gets the message from one of their own and not directly from the outside. Additionally, the security champion makes sure that security is not forgotten about in other steps of the process.

5.7 Specialist vs. generalist knowledge

Informant A said that one challenge with DevOps is that "*Cross-functional teams would mean that everyone is an expert in everything. Maybe specialist skills that make work high quality will be lacking. So the challenge is when do you stop integrating and when have you come to the right place*". Informant C makes a similar claim by saying that "*The problem with DevOps is that it is impossible for everyone to know everything, it is difficult to be an expert on all fields so just by the nature of things someone is going to be good at security, someone at development someone at design*". These claims illuminate an important issue. Is it possible for security to become a natural part of the development process that programmers take care of? Perhaps programmers don't have the time to acquire enough security knowledge on top of everything else they need to think about.

Informant F, who was an advocate of cross-functional teams, was asked about this.“ ***You say that everyone should think about security and there should not be a specialized team that takes care of it. Are you worried that we will lose specialists and that everyone will become too general?***”. He says that he sees the point of this claim, but adds “*I have the opposite experience*”. He goes on to saying the following “*The more awareness there is about security the more developers start talking about it and maybe an interest awakens in them that was not there previously. People might suddenly find security to be really interesting and develop into white-hat hackers who check if things are done the right way. My experience with security, which is a very new and not matured field, is that the more people that talk about it the better. Otherwise it can end up downing away in things that seem more ‘cool’*”

5.8 Reluctancy in prioritizing security

When looking through the interviews we saw that several subjects mentioned some form of reluctance in prioritizing security.

Informant D points how management did not want to prioritize security. He starts by explaining that in his company “*there was not much focus on security and nobody worked with it". When asked about security culture he says that “People did not really care enough*”. Informant D worked at a company that had a security breach which cost them 300 000 NOK. He describes how management “*stopped caring and we never managed to figure out why this happened*”. When asked about what could have prevented the breach he responded with “*To hire someone full time who cares about security. The management did not care about security*”. This shows that management also is important in creating a security culture. An interesting aspect of this finding is that even after the breach, management still did not take security seriously. A hacking attack was not enough to create a security-culture shift.

Informants C and E also show signs of the security push-pull. For example, informant C begins by mentioning that “*If someone were using the system and has stolen the username and password and were using the system it would be difficult to detect*”. He refers to how someone could steal credentials to log into the Norwegian Id-system. He recognizes a threat but then says “*the risk is quite low because it is two-factor, so stray accounts are maybe not that dangerous*”. In this sentence he implies that even if someone were to steal an account the two-factor system would stop them. This was coded as the security push-pull where informants constantly need to decide if a risk is large enough to consider. Sometimes they identify a risk, but then go back and say that it is not a big deal after all.

Informant E did a similar thing, he said "*We are providing a free service for downloading wallpapers, but we still think about security but we don't need to certify ourselves on being security engineers, but it is good enough to be reasonable about it*". He says that they are thinking about security, but at the same time makes security seem as not that much of a big deal.

5.9 Degree of results validity

Assessing the quality of qualitative research can be a complicated process [47]. This is because the issue of quality in this type of research is part of a larger debate. Critics might say that qualitative research does not provide valid results as the findings can't be generalized [47]. However, we have attempted to create valid results to the best of our ability. Also, DevOps is not just about automation, but culture and cooperation. Qualitative research is beneficial in understanding how people experience these phenomenons. Therefore, it was seen as necessary to conduct qualitative research despite the controversy in result validity.

THe article *Assessing quality in qualitative research*[47] presents questions that researchers can ask themselves to assess the validity of findings. The questions and answers are given below.

Are the main claims plausible enough to be accepted at face value?

Certain claims in this report are technical, such as "Informant B says their company uses these tools". Such claim do not hold a lot of room for interpretation. It is straightforward to write down the tools used by the company of Informant B. RQ1.1 which is relevant to this part is: "*How can security tests be automated?*". How one company automates their security might not be relevant to how all companies can do it. However, it still provides input on one potential solution.

There are several claims in this report about how there is a divide between developers and security professionals. These claims are more subjective. When an interview subject claims that the security team spreads propaganda this can mean several things. The subject could mean that spreading of propaganda is only the case sometimes, or only in relation to certain tasks. To say that there always is a divide between security and development from these interviews might seem non-plausible. However, as many interview subjects unprompted mention the same points the claims become more valid. Additionally, Informant F who has worked on numerous security projects confirmed the claim. The large, international company he works at avoids separate security teams because

they have the same experience as many of the interview subjects. This again makes the claims more plausible.

If so, is the evidence sufficient, both in terms of strongly implying the validity of the main knowledge claim and in being sufficiently plausible or credible to be accepted?

It is difficult to say if six interviews provide sufficient evidence for all the claims. However, we have attempted to combine the claims with evidence from literature where it is available. As DevSecOps is a new topic material from literature was limited. Therefore, the claims are not backed up by several other researchers with the same findings. There is no strong evidence showing that the findings are universal. Still, they are the subjective experiences of our interview subjects. As not much other DevSecOps literature is available they could be a solid starting point for further research.

6 Discussion

6.1 DevSecOps Definitions

In section 5 it became clear that Informants had differing views on what DevOps is. Some informants referred to DevOps as an activity you can do alone. Informant C said that there are standards for DevOps while B and D said that there is no standard way of doing DevOps. In theory it was presented how individuals in the DevOps community disprove of the title "DevOps Engineer" and the word "DevSecOps".

DevOps Engineer usually means someone who works with automation. Informant D and E applied this definition when they said that they do DevOps on their own. However, DevOps is more than just automation. Even though the definition of the term is not clear the initial literature study shows that most research describes DevOps as more than automation.

If DevOps is more than automation than the term DevOps-tool can also be criticized. As shown in theory Ken Mugrage says that the term DevOps-tool is too broad. What would a DevOps tool be? Something a developer uses, something a security-specialist uses? A tool for automation? If the tool is mainly for automation (such as Puppet) than it is an automation-tool. The term DevOps-tool appears to be too broad.

It is important to be clear about terminology because if there is no definition of DevOps than it becomes unclear how to implement DevSecOps. For some DevSecOps would mean using tools to automate security testing, while for others it would mean having developers do security instead of an external team. It is difficult to talk about how to implement DevSecOps without a clear definition of DevOps.

DevSecOps in itself might be an unnecessary term. Myrbakken and Colomo-Palacios introduce DevSecOps by saying that traditional security methods have struggled to keep up with DevOps agility and speed [6]. This is the opposite of what Informant F is saying. As a security architect informant F says one *purpose* of DevOps is to increase security. He sees DevOps as a way of creating *even more* secure systems. The DevOps Handbook also presents security as a natural part of DevOps. The original idea behind DevOps was

enhanced cooperation [22]. Even though not all areas are mentioned in the "DevOps"-word, they are still not excluded. According to Ken Mugrage if someone is doing DevOps they still need to do security. Creating a word such as DevSecOps "lets DevOps off the hook" as he says [10]. It might also be unnecessary to continuously expand DevOps until the term grows into DevSecTestOps (as Informant A mentioned).

Even though the DevOps term is undefined, it is widely used. In theory it was shown how DevOps principles help organizations as documented by the State of DevOps reports. Therefore, despite lacking clear definitions the DevOps movement has aided organizations. However, it is difficult to create a standard for secure DevOps when the term is undefined. If one organization thinks DevOps means automation, while another thinks it means cooperation, than DevSecOps will not mean the same thing for these two companies. Still, while researching what the term DevSecOps could mean we found several issues and solutions regarding culture, measurements, automation and sharing. We did, however, not find a standard way of doing DevSecOps.

6.1.1 Future Research in Defining DevSecOps

Future research could look into how to define DevSecOps, or if the term is necessary in the first place. Informant F mentioned that he would like a tool for automating security that everyone is using, with a large community. He says that currently there are many tools and practices with no standards. Possibly defining terms such as DevOps or DevSecOps could lead to more standards. Securing a production pipeline and knowing how to organize various teams might than become easier.

6.2 Automating Security

As shown in the result-section several informants indicated that tools alone are not enough to ensure security. Informant F emphasized that while they are trying to set up tools, training of developers and thinking about security from the beginning is still most important. Informant A talked about how security is difficult to automate even when you have tools such as Metasploit. Metasploit can be scripted to perform certain tests automatically, but at the end of the day manual configuration is still necessary.

A security professional suggests to automate task such as patching, malware detection and data protection [48]. However, automating penetration testing turned out to be difficult and inaccurate. The findings from theory are in line with what Informant A said about the difficulties of automating security tests. Informant F also mentioned that while his company are working to set up automated security tests, things are still in an early

stage. Some informants, such as Informant D, never managed to automate any security. Therefore, it appears as if automating security is not an easy task and will most likely differ in nature from company to company.

Additionally, it could be discussed if tools or other third-parties can lead to a false sense of security. In the result-section it was mentioned how Veracode marked their security products as "effectively eliminating vulnerabilities during the lowest-cost point". However, Informant F said that tools often show false positives and that developers still need to know how to deal with the warnings they get. If developers know false alarms can occur they might potentially be likely to skip over too many warnings.

Some informants also implied that they felt a sense of security by moving to the cloud. It was shown in results how Informant D said that you get a lot of security for free by using AWS. On AWS websites it says that "*Cloud security at AWS is the highest priority. As an AWS customer, you will benefit from a data center and network architecture built to meet the requirements of the most security-sensitive organizations.*" [49]. An insecure application being deployed with AWS will still be insecure. It could be beneficial for companies to get an overview of which security features cloud service providers such as AWS offer, but also what the limitations are.

6.2.1 Future Automation Research

There is a lot of room for future research when it comes to security automation. There is no standard for how to automate security or what should be automated. As certain experts claim that patching and malware detection can be automated, future research can look into how to do so. It would also be interesting to see if tools that attempt to automate security truly cause software to be more secure. If this is the case, is it enough to just have the tools or do developers need to go through security training? Also, it would be interesting to see to what extent security is provided by platform such as AWS.

6.3 Creating a security culture

In section 5 there are several interesting findings regarding security culture. Informant F talked about how his company is explicitly trying to avoid having separate security teams. Several informants mentioned that developers don't like being commanded from the outside to fix security flaws, especially late in the coding process. Programmers are the ones who, at the end of the day, need to fix software vulnerabilities. If they feel like they are forced to take considerations they do not wish to take, or that their work is being criticized, this could have catastrophic consequences. Informant A also says that

nobody wants to take responsibility for security because it "*adds nothing*". This was said in an exaggerated manner, but Informant A was still referring to the fact that security does not add new functionality. Informant F also said that security "*can end up downing away in things that seem more 'cool'*".

Unfortunately there is almost no research done on how developers experience security. In section 2 it was shown how traditionally there has been a divide between software engineering and security. However, not much research addresses this issue. In this report it has come forward that the one of the core parts of DevOps is cooperation. DevOps principles are also relevant when it comes to the developer-security divide. However, almost no research looks at the developer-security divide directly. This is unfortunate as we already established that tools alone don't solve all security-related issues. There still needs to be a will from the developers side to create secure code.

Creating this will can be difficult as programmers often think about adding new functionality. Security is also about considering risks that might never happen. As Informant A describes there is a dialogue in his company regarding risk. Informant D describes a situation where a major security breach actually happened. However, this did not cause management to care more about security. Security has an interesting psychological dimension. It is about risks that might not happen and how to still make people consider these risks. As spending time on security does not add new functionality other arguments are needed for developers to consider it. Trying to convince developers by scaring them with potential worst-case scenarios might not be effective. As seen in the company of Informant D, even when one of the worst-case scenarios happened, management did not change their opinion.

Informant F describes how, by integrating security from the beginning, developers might become interested in it. At the same time, informant A said that if everyone needs to be good at everything high quality, specialist work will be missing. This is a fair point. On top of everything developers have to learn should they also be doing security. Even Informant F says that "*everyday working lives of developers and architects are full of impulses where they have to decide what to care about*". It would be interesting to see what the minimum requirements for developer security knowledge should be, or if there should be any requirements at all.

6.3.1 Future Security Culture Research

Informant F said that the most important thing in relation to security is that developers are trained in it. As previously mentioned, two other informants said that not everyone

can be good at everything. When confronted with these claims informant F said that he has the opposite experience. He believed that the more software engineers that think about security for the beginning, the better. He went on to describing a desirable scenario where developers become white hat hackers and think security is fun to work with.

In theory it was presented how software engineering and software security education is separated. In a recent article it was described how volunteers who work with software security teach Norwegian students to hack [34]. They believe that security is often overlooked in school. It is described how this results in developers thinking about security as scary and difficult to understand. Informant A also said that security is not taught in university and that the in-house training they provide is probably not adequate. Informant F, on the other hand, said that the level of security knowledge amongst programers varies a lot. Either way, sharing security knowledge in a company or even at university might be beneficial. Software engineers often want to solve problems by using tools. However, as Informant F pointed out tools can give false positives and knowledge is still required to use them correctly.

From the research we conducted DevSecOps is about several layers. There are security automation tools, but tools can not be used without appropriate knowledge. Appropriate knowledge can not be acquired if a company does not have a culture where they care about security. Measurements can be used throughout the whole process to monitor security status. Figure 6.1 illustrates these points.

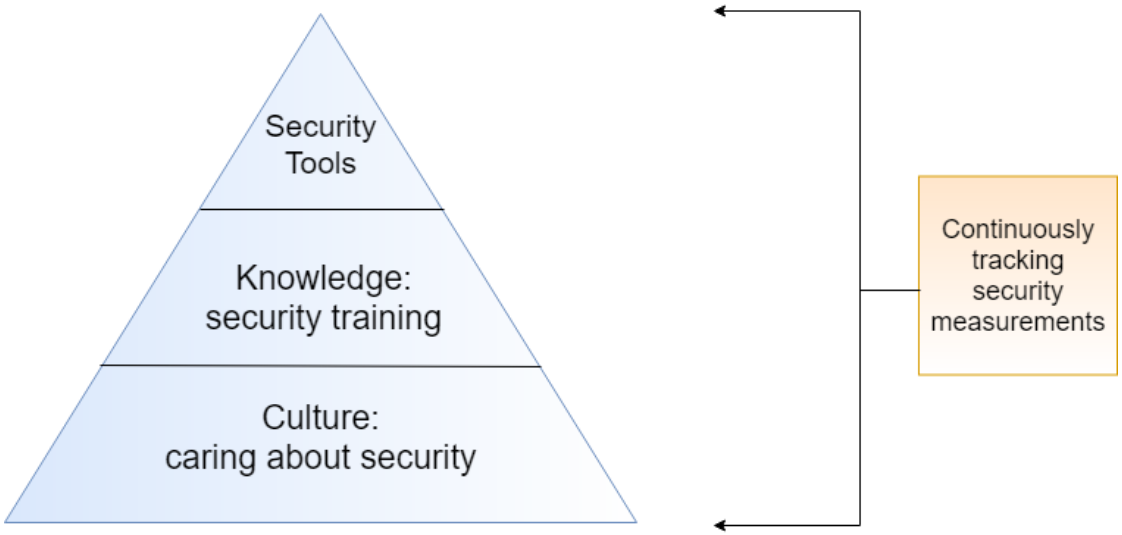

Figure 6.1: DevSecOps Pyramid

Culture is the foundation of the pyramid. Without a solid security culture developers will “take shortcuts” as Informant F said. Another interesting aspect that Informant F pointed out is that his company did a lot of measurements. These measurements were

not only about technical aspects. In theory it was shown which measurements can be used when doing DevOps. Many of these measurements are quite technical, such as memory and CPU percentage usage. However. Informant F pointed out the importance of measuring how many developers that have gone through training.

It is clear from the findings that having a security culture is important. Therefore, some measurements could be conducted on how developers and operations view security. They could be asked on a regular basis if they find security to be important, how much time they spend working on it, and perhaps even if they think security is fun. It appears as if all of these factors are important in creating secure applications.

7 Conclusion

At the beginning of the report the following research questions were asked:

- RQ1: How can DevSecOps enable programmers in writing code fast and secure?
  - RQ1.1: What is the state of art of DevSecOps in relation to the four pillars of DevOps (culture, automation, measurements and sharing)?

Throughout the report we found that the answer to these questions are not straightforward. We found that there is a division between programmers and security professionals. Security is often not mandatory in computer science courses. Programmers who are working might often feel like security teams exaggerate problems. Additionally, programmers might feel criticized when security professionals tell them to change their code. Some interview subjects used strong words to describe this, such as "propaganda" and "being forced" to do things.

DevSecOps was seen as a potential solution to this problem. DevSecOps has roots in DevOps which promotes collaboration between different departments, such as development and operations. DevSecOps is about shifting security to the left and making it a natural part of the development process. We aimed to find out how DevSecOps would enable this shift, and what the state of art is. However, this turned out to be challenging because DevSecOps as a term is controversial. Several people believe that DevOps already includes security. Others hold the opposite view, claiming that DevOps and security don't naturally go together. Therefore, there is no real definition of what DevSecOps is, or if it is even necessary in the first place. Every company has a different way of doing DevOps and maybe even a different name for it.

We still managed to get some key findings from the interviews. Figure 6.1 shows how we categorized the findings. Initially, we expected people to talk about which security tools they would like to have. We thought automation would be what most informants would focus on. However, this was not the case. Informants talked a lot about security culture. A security architect emphasized how important it is that developers care about security. He additionally talked about the importance of knowledge. Security automation tools need to be used by developers who have the required knowledge. However, the

required knowledge will not be generated unless the company has a culture where they care about security. Therefore, we placed culture at the bottom of the pyramid. Security automation tools in themselves will not add value unless the two previous levels are in place. Throughout this whole process it is possible to track security measurements to see the status quoe. The pyramid is the closest answer to how DevSecOps can enable programmers. As in DevOps, culture is a large part of it and tools are useful when the culture has been created.

These findings are quite broad and it was difficult to pinpoint narrow problems. Instead, most informants talked about fundamental ideas. Such as that security is a separate track both through education and in working life. The potential for future research is therefore quite large. From education to how to create a company culture shift, to the actual usefulness of security automation tools.

References

[1] S. Mezak. (2018). Data breaches compromised 4.5 billion records in first half of 2018, [Online]. Available: https://www.sttinfo.fi/tiedote/data-breaches-compromised-45-billion-records-in-first-half-of-2018?publisherId=58763726&releaseId=69844038.

[2] Y. Klijnsma. (2018). Inside the magecart breach of british airways: How 22 lines of code claimed 380,000 victims, [Online]. Available: https://www.riskiq.com/blog/labs/magecart-british-airways-breach/.

[3] S. Kraemer, R. Carayon and J. Clem, 'Human and organizational factors in computer and information security: Pathways to vulnerabilities', *Computers and Security*, 2009. DOI: https://doi.org/10.1016/j.cose.2009.04.006.

[4] A. Raman and S. Muegge, 'An integrated approach to security in software development methodologies', *Canadian Conference on Electrical and Computer Engineering*, 2008. DOI: 10.1109/CCECE.2008.4564898.

[5] N. I. of Standards and Technology. (2002). The economic impacts of inadequate infrastructure for software testing, [Online]. Available: https://www.nist.gov/sites/default/files/documents/director/planning/report02-3.pdf.

[6] H. Myrbakken and R. Colomo-Palacios, 'Devsecops: A multivocal literature review', *Communications in Computer and Information Science Software Process Improvement and Capability Determination*, 2017. DOI: 10.1007/978-3-319-67383-7_2.

[7] B. Chess and G. McGraw, 'Static analysis for security', *IEEE Security*

Privacy Volume: 2 , Issue: 6, 2004. DOI: 10.1109/MSP.2004.111.

[8] S. Mezak. (2018). The origins of devops: What's in a name?, [Online]. Available: https://devops.com/the-origins-of-devops-whats-in-a-name/.

[9] G. Kim, J. Humble, P. Debois and J. Willis, *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution Press, 2017.

[10] K. Mugrage. (2018). You only have to change one thing to do the devops, [Online]. Available: https://www.youtube.com/watch?v=WEuYFq7crdY&t=1504s.

[11] V. Almazova. (2017). Appsec eu 2017 security best practices in azure cloud, [Online]. Available: https://www.youtube.com/watch?v=HJthzZO1D9A&t=125s.

[12] Sonatype. (2018). 2018 devsecops community survey, [Online]. Available: https://www.sonatype.com/2018survey.

[13] Amazon. (2018). What is devops?, [Online]. Available: https://aws.amazon.com/devops/what-is-devops/.

[14] J. Roche, 'Adopting devops practices in quality assurance', *Communications of the ACM. 56. 38-43*, 2013. DOI: 10.1145/2524713.2524721.

[15] J. Smeds, K. Nybom and I. Porres, 'Devops: A definition and perceived adoption impediments', 2015. DOI: 10.1007/978-3-319-18612-2_14.

[16] C. Ebert, G. Gallardo, J. Hernantes and N. Serrano, 'Devops', *IEEE Software*, vol. 33, no. 3, pp. 94–100, May 2016, ISSN: 0740-7459. DOI: 10.1109/MS.2016.68.

[17] L. Riungu-Kalliosaari, S. Mäkinen, L. E. Lwakatare, J. Tiihonen and T. Männistö, 'Devops adoption benefits and challenges in practice: A case study', *Product-Focused Software Process Improvement*, pp. 590–597, Nov. 2016. DOI: https://doi.org/10.1007/978-3-319-49094-6_44.

[18] N. Forsgren, M. C. Tremblay, D. VanderMeer and J. Humble, 'Dora platform: Devops assessment and benchmarking', *Designing the Digital Transformation*, pp. 436–440, 2017. DOI: https://doi.org/10.1007/978-3-319-59144-5_27.

[19] L. Riungu-Kalliosaari, S. Mäkinen, L. E. Lwakatare, J. Tiihonen and T. Männistö, 'Devops adoption benefits and challenges in practice: A case study', *Product-Focused Software Process Improvement*, pp. 590–597, 2016. DOI: https://doi.org/10.1007/978-3-319-49094-6_44.

[20] A. Dyck, R. Penners and H. Lichter, 'Towards definitions for release engineering and devops', *2015 IEEE/ACM 3rd International Workshop on Release Engineering*, 2015. DOI: 10.1109/RELENG.2015.10.

[21] (2001). Principles behind the agile manifesto, [Online]. Available: http://agilemanifesto.org/principles.html.

[22] J. Allspaw. (2009). 10+ deploys per day: Dev and ops cooperation at flickr, [Online]. Available: www.slideshare.net/jallspaw/10-deploys-per-day-dev-and-ops-cooperation-at-flickr.

[23] T. Fitz. (2009). Continuous deployment, [Online]. Available: timothyfitz.com/2009/02/08/continuous-deployment/.

[24] Puppet and DORA. (2013). 2013 state of devops report, [Online]. Available: https://puppet.com/resources/whitepaper/2013-state-devops-report.

[25] R. Wilsenach. (2015). Devops culture, [Online]. Available: https://martinfowler.com/bliki/DevOpsCulture.html.

[26] M. Rouse. (2018). Monolithic architecture, [Online]. Available: https://whatis.techtarget.com/definition/monolithic-architecture.

[27] Puppet and DORA. (2016). 2016 state of devops report, [Online]. Available: https://puppet.com/resources/whitepaper/2016-state-of-devops-report.

[28] B. Fitzgerald and Klaas-JanStol, 'Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices', *Journal of Systems and Software*, vol. 123, pp. 176–189, Jan. 2017. DOI: https://doi.org/10.1016/j.jss.2015.06.063.

[29] M. Shahin, M. A. Babar and L. Zhu, 'Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices', *IEEE Access*, pp. 3909–3943, 2017, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2685629.

[30] TrustRadius. (2018). Build automation tools, [Online]. Available: www.trustradius.com/build-automation.

[31] (2018). Measuring devops, [Online]. Available: https://newrelic.com/devops/measuring-devops.

[32] P. A. Nielsen, T. J. Winkler and J. Nørbjerg, 'Closing the it development-operations gap: The devops knowledge sharing framework', *BIR Workshops*, 2017.

[33] N. Mendes. (2017). Devops maturity model report: Trends and best practices in 2017, [Online]. Available: www.atlassian.com/blog/devops/devops-culture-and-adoption-trends.

[34] T. M. F. Vestheim. (2018). Studenter skal lare å tenke som kriminelle, [Online]. Available: https://www.dn.no/magasinet-utgaven/dn-magasinet-2018-11-03/5.

[35] CloudPassage. (2016). U.s. universities failing in cybersecurity education, [Online]. Available: https://www.cloudpassage.com/company/press-releases/cloudpassage-study-finds-u-s-universities-failing-cybersecurity-education/.

[36] J. Boyer. (2018). Security as code: Why a mental shift is necessary for secure devops, [Online]. Available: simpleprogrammer.com/security-code-secure-devops/.

[37] T. N. N. R. E. Committees. (2018). Ethical guidelines, [Online]. Available: https://www.etikkom.no/en/ethical-guidelines-for-research/guidelines-for-research-ethics-in-science-and-technology/protection-of-research-subjects/.

[38] Snyc. (2018). Use open source. stay secure., [Online]. Available: https://snyk.io/.

[39] Retire.js. (2018). Retire.js, [Online]. Available: https://retirejs.github.io/retire.js/.

[40] Hakari. (2018). Ship secure ruby apps, [Online]. Available: https://hakiri.io/.

[41] P. Studio. (2018). Pvs studio, [Online]. Available: https://www.viva64.com/en/pvs-studio/.

[42] Veracode. (2018). Veracode, [Online]. Available: https://www.veracode.com/.

[43] Coverity. (2018). Coverity, [Online]. Available: https://scan.coverity.com/.

[44] Sonarqube. (2018). Continuous code quality, [Online]. Available: https://www.sonarqube.org/.

[45] Checkmarx. (2018). Manage software exposure at the speed of devops, [Online]. Available: https://www.checkmarx.com/.

[46] J. Long, S. Springett and W. Stranathan. (2018). Owasp dependency check, [Online]. Available: https://www.owasp.org/index.php/OWASP_Dependency_Check.

[47] N. Mays and C. Pope, 'Assessing quality in qualitative research', *BMJ*, vol. 320, pp. 50–52, Jan. 2000.

[48] K. Sheridan. (2018). The best and worst tasks for security automation, [Online]. Available: https://www.darkreading.com/operations/the-best-and-worst-tasks-for-security-automation/d/d-id/1332074?image_number=6.

[49] AWS. (2018). Aws cloud security, [Online]. Available: https://aws.amazon.com/security/.

.1 Appendix A - First Interview

Role: Platform Tech Lead **Company size:** 200 employees **Company Product:** SaaS for smart cities **Type:** Private

Could you tell me about your job and the role you have in your company?

- Technical lead of a platform team
- Platform means infrastructure which are servers and pipeline that pushes code into production

Can you tell me about a project where you are using DevOps?

- Main project is infrastructure as code (IoC)
- Team has developed tools to allow developers to leverage the power of the platform without getting too involved into learning low level code
- We abstract away the platform itself to make tasks such as spinning up servers easier
- We are also involved in spinning up new regions and network structures

So you help developers spin up servers easier, do you have any other problems you solve for the developers?

- One of the main requests from developers is to make a copy of a database, this is time-consuming
- Especially time-consuming if you have some production data and they wish to make a change but don't know how long it will take
- Previously we would have to pull the database through a set of nodes. Data had to be modified, such as GDPR-identifying information had to be removed before being put back into development
- Previously the process would take three days, but can now be done in hours and database is made available in the Amazon development account

So a benefit of what you are doing is that developers don't have to spend a lot of time on tasks that are being automated. Are there any other benefits of using DevOps principles?

- A key principle of DevOps is more involvement from the beginning of a project

- This is important because developers are often not thinking about monitoring, alerting, scaling and other things that people on the platform team are thinking about
- Developers think about how something will work with just a few users and for a demo
- Before DevOps developers would come in on a friday and tell operations to put the code into production for next week, but operations would often send it back because it could not be scaled
- Now operations can get involved from the beginning

Did you have any challenges in implementing DevOps?

- Yes, because there is no standard
- Many people take DevOps to mean "development anarchy" so DevOps can cause developers to gain more freedom, but it is easy to go too far
- Some people think DevOps means that developers can use any system or technology and not what has matured in the company
- Especially with containers, developers have been enabled, so they see something works on their machine but don't understand why it does not work in production

Is this a challenge people are aware of, is anything happening to address them?

- It is difficult because DevOps and start-up culture tends to encourage a flat hierarchy
- In my company we have employed more engineering managers that implement certain standards, so people are allowed to say "let us not do this" because it is not standard
- This leads to more management, but in this case it helps

What tools do you use in your DevOps pipeline?

- The biggest one is Puppet
- Everything else is pretty replaceable
- We obviously use Git and other source control and things that everyone uses

Do you use any tools to automate security testing?

- We do, but the tools have not been adequate
- We have an in-house system that creates a default security configuration and tests if the environment follows the configuration
- The tests usually says that all rules should look like this, and they do
- A company also does log management for us and tells us if something looks suspicious

You mentioned the security automation testing tools are inadequate, why?

- Yes, security is difficult. If you want a server for example security is more abstract. So there is no tool that does all security for you
- What you have to test is very different depending on the environment you use
- You can get AWS config, and you can do that with Puppet on a system level, but for an overall security management suite this is difficult to generalize
- Therefore people make their own tools to manage overall compliance, which is what we also do

Do you have any security metrics that you measure in the projects?

- No, it is not something we enumerate. It is more something we do qualitatively

Who is responsible for security in the project?

- I think the best way to answer this is: someone else, no one wants that job
- we are legally responsible for security, but it is quite distributed
- developers check that code is secure, but physical security is with the operations management team

Do you feel like people have adequate training to deal with this security?

- I find that they don't teach it in university
- A lot of the developers we get have never dealt with security, security is a separate track that they did not take
- We do inhouse training, but I don't know if it is really adequate

- We do inhouse training, but I don't know if it is really adequate
- It probably is not adequate

Do you feel like you have a security culture? Do people care a lot about security?

- I feel like security is often a battle, it is never helpful it is always in the way, costs time and gives nothing
- We have a lot of conversations about if something is secure, and what can be done, but also if it is worth doing something because what is the chance of someone actually exploiting this
- Constant battle between risk and cost
- But there is a lot of opposition to security-related work because it is time consuming

Are developers open to changes suggested by security people?

- If you are in a room with people and the security guy says "I don't like the way you wrote this", it doesn't matter how they say it, they are still judging your code
- No matter how you phrase it they criticize the developers work
- Especially if you spent a few months working on something and the security team tells you in a two hour meeting that it is no good

Do you have a way of sharing security knowledge?

- We have a security council, which is a far grand name for four men who sit in a room and discuss the latest security findings
- So we do talk to each other about security in general

Can you think of any tools or processes that can help you create more secure programs?

- There is some way to go on vulnerability testing on the server level. Right now pen testing is human-based
- There are tools like Metasploit but a lot of configuration needs to be done by humans
- Amazon also has an inspector that checks if installed packages look vulnerable, if you want a more extensive analysis you still have to hire a human to do it

In relation to processes, you mentioned that it would be better if people agreed on security practices in beginning?

- Yes. I am a big believer in frameworks. It is easier to tell someone "you can't do that " if everyone has agreed to a framework
- For example if you are PCI compliant you have to follow certain rules. If you don't have a framework it is much more difficult to convince people
- Like some ISO standards where everyone has already agreed that it is a good idea
- If you try to come with security after you have written code, you have already given yourself a lot of technical debt

What kind of challenges related to DevOps do you think will come up in the future?

- The next step is DevSecOps, which is interesting, but then once you finish that assimilation then you suddenly have DevSecTestOps and so on
- Cross-functional teams would mean that everyone is an expert in everything
- Maybe specialist skills that make work high quality will be lacking
- So the challenge is when do you stop integrating and when have you come to the right place

.2 Appendix B - Second Interview

Role: Software Engineer **Company size:** 11 000 employees **Company Product:** Software Security Services **Type:** Private

Can you tell me about your company and your role there?

- Software engineer at security company
- Company provides hardware box that sniffs for malicious traffic
- My job is to program and make sure the box works

Do you have a project where you say you are doing DevOps?

- Company-wide don't know, large company and some teams might be using DevOps
- Our team does not say we use DevOps, but we want to adopt the mindset of DevOps

- We want to hire individuals with knowledge from different fields
- Trying to move our systems to the cloud

Do you use any tools to automate security? For example to automate security testing?

- Yes, as a security company we have strict security measures
- Running automated security tests
- Using tools such as Snyk, Retire.js, Hakiri, PVS-Studio, Veracode, Coverity, Sonarqube, Checkmarx
- Tools that give overview of library weaknesses, libraries with weaknesses are not released

As a developer, do the security tools stop you from deploying code as fast as you would like?

- After a while you adapt the security mindset so it becomes less and less of an issue concerning speed
- However, some security features can be frustrating for developers. For example, you can not copy-paste code from the internet because the code you are writing has to be isolated and on a separate network

Do you measure any security metrics in your projects? Which tools do you use to do so?

- We do track security metrics, but not as a part of DevOps
- This is done by a large security team, developers do not see the metrics, but only make changes if it is requested by the security team

Who is responsible for creating secure code in your projects? Do you have a dedicated security team?

- We have multiple teams, some security teams are on other projects
- We also have a legal security team that checks legal requirements

Do you think you have a security culture in your company? If so, how do you ensure that this is the case?

- There is a lot of push to create a security culture. We (developers) are always laughing about it
- We get a lot of propaganda
- We have a lot of seminars and have a lot of tools and teams that make sure security is in place
- Developers don't have a choice

You are saying that developers sometimes laugh at the changes proposed by the security team. Is there a divide between software engineers and security teams in your company?

- Yes. Not being able to copy-paste is troublesome and it does not make things more secure. People could still take a picture of the code or just copy the lines manually
- Copy-pasting is just one example, but it seems like many of the things security teams suggest are just for show
- We would like to decide this with our own opinion. Programmers are good at security also, often times we have more knowledge than the security teams on some topics

Do you have any processes for sharing security knowledge in your company?

- We use wiki-pages and Slack, which helps with sharing information
- Also seminars, as mentioned previously

Can you think of any additional tools or process that you would like to have in your company that would help you create even more secure programs?

- I don't know. I would actually not like to add layers, but perhaps remove some of them
- A lot of our code is written in Python, and it is in the box and the customers (or anyone who buys the box) could see it anyway, so it makes little sense trying to hide it like the security teams suggest

.3 Appendix C - Third Interview

Role: Software Engineer **Company size:** 1300 employees **Company Product:** Consulting **Type:** Private

Can you tell me about your project?

- Software consultant, currently consulting at a public department office in Norway
- The Department has been very old-fashioned, you need a license to dig for rocks, earlier that licensing was done on paper and we have a project to modernize the systems
- Making the old processes of the department digital

When did you hear about DevOps and for what project is it used?

- The expression has been big the last five years, but I think a lot of teams have done stuff like that before
- The concepts are not new, but having a system and a description for it is now and on a high. The first project where we had full automation was probably my first project, in 2012
- . In 2010 also, maybe, when I was a student on my part-time job

Do you use any tools for your DevOps project?

- What we have is an automated loop, we have source control
- Then we use Git and all of it is gated through to code review and a build that is done through Jenkins, testing and all is automated here
- When that part is ok the code will be submitted into the database, Jenkins is also responsible for producing our build artifact, in this case docker images
- Docker images are pushed to a repository called Nexus, they are immutable can never be changed and are versioned
- Ansible is responsible for deploying things automatically to Linux servers. Anything with servers is done with Ansible
- Ansible sets up a few other services, like logging and all the output goes to a system with elastic search to logstash, we use Kabana to read logs and give back information
- Also running Monit , for monitoring and active lookup like are you running things on the right ports and it sends a mail if something is wrong
- It has been very important for us that everything is automated

Do you have any special tools for security?

- Other than monitoring logs, not really
- We have security tools running on servers, but not related to DevOps
- . we have firewalls and a tool called “failed to ban”. There are rules if someone has failed an SSH login 3 times (for example) they get blocked by the firewall

How do you ensure security during the development process?

- Everything is connected to our active directory, everything is encrypted, but there are no active probes that say that someone did something in the code-base
- We have some static code analysis, but if someone were to use my password to do something we would not notice

Do you have a special person working with security?

- No, everyone is supposed to take responsibility, but in the end everyone becomes more and more specialized
- Everyone on the team has no problem deploying a new version to the customer

When you do DevOps do you have some measurements?

- No, that is our sin, we do keep an eye on things but have no special measures with response time, and engagements and so on
- We do try to check if anything produces a lot of errors. But we use Kibana to graph it out and display a dashboard

What is the difference between the state and private companies?

- state is much more rigid, there are a lot of rules for what you can buy and when you can buy it
- in private companies everything is easier and faster and more agile and in the public it Is a bit more proper

Do you have any knowledge sharing between Development, Operations and Security?

- Right in this project we have very little operations, but we do have some operations because all of this runs on Linux

- We don't own the physical hardware but Atea has it. We don't really share very much with them, they do their job we do ours
- If they detect Ddos attacks or something in the network they block stuff and contact us

Did you meet some challenges In implementing DevOps?

- This is quite standardized, there is a lot of information, it is not hard to set up
- one thing interesting with security is how do you keep secrets
- We use ID-porten, to do that you need to have certificates, passwords and usernames and how you deal with that is always a challenge
- We have them encrypted and separated in a new file store but when you have your project live you have to decrypt again
- To address this challenge you need to have people who know what they are doing, but if some of the developers wanted to take it with them and do something bad they could but at some point you have to trust your employees

Are there any challenges you think you will meet in your future projects?

- It has not been a challenge for us, but measurements could become a challenge in regards to security
- . If someone were using the system and has stolen the username and password and were using the system it would be difficult to detect
- So maybe we should have had an alarm if a Norwegian user logs in from another continent or something like that
- With using the national logging system the risk is quite low because it is two-factor, so stray accounts are maybe not that dangerous

What is the difference from your first DevOps project and the one you are working on now?

- It is a bit hard to answer because first it was Microsoft and .NET and now I am working with Linux and Java
- people are not using is the measurement and metrics. Not very many people do formal measurements
- If there is one part that is missing, that is the one

- When I talk about it I think we should probably do that but we do not. Luckily, for us there are not too many users so we have not run into any problems when it comes to scale, but it would be nice to know

What do you think DevOps will be like in the future?

- I think containers have been a huge part and it has been very important in making DevOps popular
- People are making microservices with Docker and splitting the application into many units
- There are quite a few alternatives so I think that is a space that is going to be improved

How can we combine security into DevOps?

- The problem with DevOps it is impossible for everyone to know everything, it is difficult to be an expert on all fields so just by the nature of things someone is going to be good at security, someone at development someone at design
- If you have a team without someone who knows security it will be hard to do this
- The container area gives you nice security features and maybe some false sense of security, but you never know, there are exploits to get out of it
- There are a lot of good cloud services that help you with stuff like that now and it is very easy to think about security when you are using the cloud

Are there any security tool you would love to have?

- Probably already exists, but since we have monitoring on port uptime, we should probably have something for known fingerprints of attack
- if we see malicious traffic we get a red flag, our hosting company helps us with that but for our sake it would be nice to have
- These kind of systems would not know what part of our application are prone to danger

.4 Appendix D - Fourth Interview

Role: Software Developer **Company size:** 30 employees **Company Product:** Platform for Web Sites **Type:** Private

Can you tell me about your company and your role there?

- Company delivers web pages to 200 large and small customers
- The biggest customers had a few millions NOK in earnings and the smaller customers a few hundred thousand NOK, so customers are both llarge places and family-businesses

Do you have a project where you are using DevOps?

- Yes, on our main project, I implemented the DevOps part such as CI and CD systems

What do you think are the benefits of using DevOps principles compared to not using it?

- We had no other choice, because we did not have a separate Operations department so people doing ops and dev were always sitting in the same room from the beginning
- There were around 7 - 8 people

Did you have challenges with implementing DevOps on a project level? How did you address these challenges?

- It was difficult to make things correctly
- We had system problems more than practical problems, cooperation was fine but making the systems communicate was difficult
- The boss also did not understand the importance of DevOps

So the boss did not understand the advantages of using these tools?

- Correct

You say it was challenging to set up the tools?

- Yes, because we had to use the tools with undocumented legacy systems
- It was difficult to put up CI/CD with legacy systems
- There were a lot of forgotten dependencies in the system that nobody had documented

How did you address these challenges?

- We spent a lot of time, mainly
- A combination of a lot of time used and moments when we were working slowly because things were not well-documented
- Things were solved in the end
- The boss worked on this system alone for a long time, and things were not created so that others could read them later

Was the effort worth it when you managed to set up the pipeline?

- Yes, the developers thought so and the boss also saw the value of using Jenkins for deployment

What tools do you use in your DevOps practice? Do you have tools to automate security?

- Mainly Jenkins, we used AWS, Ansible for deploying system, and of course Git and those kind of things
- I made some security tests, but we never managed to set up something permanent that tested security
- We never managed to automate any security

Do you measure security metrics in your projects? What tools do you use?

- No, we did not measure any metrics

Who is responsible for security in your projects, do you have a dedicated team?

- AWS controlled some of this like segregating networks and you get a lot for free
- Other than that there was not much focus on security and nobody worked with it

Do you have a security culture in your company? If so, how do you ensure that is the case?

- People did not really care enough
- We actually had a security breach, AWS API root keys that control all the resources were leaked
- We got a fee on 300 000 NOK

- We had to figure out what happened and why they were compromised but we never managed to find the reason. The boss stopped caring and we never managed to figure out why this happened

What do you think could have prevented this?

- To hire someone full time who cares about security
- The management did not care about security

Do you have any processes for sharing security knowledge in your company?

- No formal process, we sometimes talked about it

Can you think of any additional tools or process that you would like to have in your company that would help you create even more secure programs?

- We should have had some routines for updates and following security
- For example having patch Tuesdays where we patch systems
- I don't know how other companies implement such things

What challenges related to DevOps do you think will come up in your company?

- Either it will be that everyone who runs a system is on the environment
- But DevOps could turn out to be project-based where one person sets up a unique stack, and maybe development does a bit operations and operations does some development but the stack is unique
- This is scary because we could end up getting a unique setup for every project, for example everyone has their own AWS account where they set up Kybernetes on their own from scratch. So there is no common standard and you have a problem because if the few people who put up the stack and they leave nobody will have overview of what is happening

.5 Appendix E - Fifth Interview

Role: Software/Data Engineer **Company size:** 40 employees **Company Product:** Mobile App **Type:** Private

What role do you have in your company?

- Software engineer
- Now more a data engineer
- ensure that systems are running when event logs are received

What does your company do?

- Make an application where people can get new wallpaper or games for the phone
- Provide number one platform in customizing phones

When did you first notice the concept of DevOps?

- I call it a buzzword, because it kind of is, but it is an old one
- Most companies have some degree of DevOps
- Don't think many companies have done automation of everything
- In our company we do all the assembling of application packages automatically, unit tests, CI, build server
- Lately we are always focused on continuous delivery, but there are still manual steps involved
- But another thing that I noticed in 2015 was when Kubernetes arrived it helped with one really key point, and that is how to describe your resources and application in a declarative way

Do you have any projects where you are using DevOps concepts?

- I apply DevOps all the time, I automate all recurring tasks. If you do a task more than two times then you automate stuff or find ways to make it less of a burden

Do you do any measurements in your work?

- yes we use measurements, all alarms and paging and stuff are based on metrics
- We use a "time-series" database, this one is called Prometheus
- Time-series database is a very efficient way of saving data
- If we make a service or something we try to make a dashboard, each service has a different dashboard

- The key rule is to set up warnings we get them in Slack so we are able to react on them the next service days
- We define our own metrics, but we have a goal of certain metrics that are global across all services
- We have a goal of moving our whole platform on the cloud, it is more or less done, but not yet

You say you have many metrics for measurements, do you have any metrics for security?

- We do think of security, and after moving to the cloud we get the benefit of that they have utilities for scanning images
- The software scans Docker images and points out security
- Security is always a battle between usability and security, and we do think of security

Who is responsible for security in the project?

- Every code contribution goes through review process
- We are such a small company and can check each other, so nobody in particular is responsible

Do you think it is necessary to have a special person for security?

- It makes sense, but depends on how large the company is
- We are providing a free service for downloading wallpapers, but we still think about security but we don't need to certify ourselves on being security engineers, but it is good enough to be reasonable about it
- Every developer should be aware of security issues, it never helps to put a burden on just a person or group of persons, but if the company has many employees you can add them as an extra guard
- That is wrong because you should test certain things yourselves. And same for security if you run a root or superuser account you should be aware of the security issues that can arise

Do you often share knowledge between development, operations and security?

- Yes, that is quite normal in agile. Maybe it is just me but we hold daily standups
- We started holding like tech-Friday tech-talks in the company. We had it for five months, because that was supposed to be a way of sharing knowledge

What DevOps-related challenges have you run into and think you will run into in the future?

- Where should we start, there is always issues. We are moving to a cloud environment, so we basically swapped out our whole infrastructure
- We are still working on nice tools to do like semantic versions of all our services and those kind of things
- We are definitely missing system tests, we have plenty of unit tests some integration tests but no system tests

.6 Appendix F - Interview Six

Role: Security Manager/Architect **Company size:** 42 000 employees **Company Product:** Consulting **Type:** Private

Could you tell me about what you work with?

- At the moment I am employed in a company that does development as a security manager or security architect

Do you work on projects where you say that you are using DevOps?

- Yes, at our current project we are trying to implement DevOps
- Some development teams have come further than others at the project
- We try to create autonomous teams so that each team does CI and CD and can deploy their part of the project
- The teams try to do frequent deployment and testing
- We have not gotten so far in the process yet. The platform that would enable these processes has not yet been put up

Do you get any benefits of using DevOps principles?

- The main benefit I as a security specialist see is that you can add more mechanisms to improve security for example code scanning and automated security tests, dependency checks and so on

- This way we can fail earlier and to be able to see where things are completely wrong or where there are smaller problems
- In many projects I have been in there are many dependencies and when one subsystem goes down it influences other systems

Did you have any challenges in implementing DevOps? You say some teams have come further than others. How did you address these challenges?

- Challenges in removing some old systems like production setting review boards
- Previously you finished parts of a system and handed that over to test
- However now you don't want a system to have to go through stages and potentially have bottlenecks
- Especially when you wish to get many things into production you want to have parallel mechanisms

Do you mean that for example security testing should happen at the same time as development, instead of things going in phases?

- That is correct, and to think about secure design from the beginning
- This ensures that there are not separate people who work with security and security is not a separate activity. Instead it is part of development
- We wish to use tools that can early see that security is affected. We wish to automate this as much as possible

Do you use any tools to automate security testing?

- We use something called OWASP dependency check on Java-projects
- We have some proof of concept with Coverity and Checkmarx
- Checkmarx is used a lot on projects outside of Norway for code scanning but also some integration towards dynamic scans, but I don't know all the details there. They have their own DevOps platform where you get access to these tools
- Most of our tools go together with a standard Docker-Kubernetes setup

Do you have faith in that these tools will assure sufficient security?

- I think mainly it all should start with increasing knowledge and training developers in security

- For example when working with web-applications people often get sent to OWASP Top 10 courses and secure development process awareness and also teaching people about risk analysis
- I would say this training and increasing knowledge is the most important part
- These tools are not able to identify all mistakes and one problem is that they show a lot of false positives
- So tuning these tools, especially on projects that are not very large might not pay off

You said security training is important. Do you get the impression that developers know a lot about security before they start on a project?

- This varies a lot
- Some developers have a lot of knowledge about security while sometimes security is not even a topic because the information we dealt with has not been sensitive

Do you have any security metrics that you measure in the projects?

- Yes, we do a lot of measurements
- In our projects we check how many of the core-developers have done how much training
- All mistakes are measured in different security categories, like in OWASP top 10 where we categorize mistakes and check if any of them stand out
- We check how much time we spend in correcting mistakes in different categories per team. We check if some teams for example have a lot of checked in code but also a lot of mistakes
- In the next phase we check how many systems are affected by external and internal pen-testing
- We have chosen this because we have seen that previously there has been a lot of security holes that happen because people are in a hurry. So some security flaws would appear as late as user acceptance-testing or right before things are about to go into production
- We try to check if we are introducing too many new features compared to technical debt that we get

You said that teams should be autonomous and everyone should take some responsibility for security. So you don't want to have an external security team that gives instructions?

- This is something we have explicitly tried to avoid both on a project-level and in our company in general
- Instead we try to have a security champion which is the person who has most security training
- We give these people the responsibility to lift the security level for all developers

It seems like developers sometimes think it is stressful to get feedback from an external security team, especially late in the process. Do you think this is true?

- Yes, the security team can be seen as a bottleneck. Also the security team can be seen as people who force changes on developers.
- With a separate security team developers often try to avoid them all together or just do the least possible amount of work for it to be approved
- The way we see it is that the more people have security-training the less vulnerabilities will be created, at least in theory
- The better tools we have in every step of the process the more mistakes we can pick up
- With more knowledge people can also judge the usefulness of tools more
- DevOps perhaps says that there should be no security champion but we still chose to have one because then we know who to call if something happens

Do you believe the other on the team listen to the security champion?

- I don't have so much experience with this yet, but my impression so far is that this is a person others appreciate. Especially if they care about security. Because then this person can give feedback about what tools to use, for example
- It is important to know that this is not an administrative role. The security champion just ensures that teams have the tools they need and that security is taken care of

Do you feel like you have a security culture? Do people care a lot about security?

- Yes, it is at least better than before when you had an external security group that just was a "show-stopper"
- This group would just come in from the outside and tell the team to make changes because it says in a document somewhere, to exaggerate iit a bit
- It is at least better that those who take care of security are on the team
- Security should be like any other step in the software development process where it is taken care of from the beginning and thought about all the way. Just like designing prototypes where you have a feedback-loop
- Hopefully in some years we will not have to specify any roles but it will be natural to think about security just like you think about using Docker or Kubernetes in Azure or anything like that. So if you have a Kanban-board that you consider security instead of doing it as a side-project, but for now things are still happening a bit in parallel

You say that everyone should think about security and there should not be a specialized team that takes care of it. Are you worried that we will lose specialists and that everyone will become too general? For example everyone thinks about security, but because nobody is specialized in it the level is not too high?

- I see the point, but I have the opposite experience
- The everyday working lives of developers and architects are full of impulses where they have to decide what to care about
- The more awareness there is about security the more developers start talking about it and maybe an interest awakens in them that was not there previously
- People might suddenly find security to be really interesting and develop into white-hat hackers who check if things are done the right way
- My experience with security, which is a very new and not matured field, is that the more people that talk about it the better. Otherwise it can end up downing away in things that seem more "cool"

Can you think of any tools or processes that can help you create more secure programs?

- Yes, what I feel is missing is that right now there are so many tools that are available. I wish there was some kind of "Maven" for security or "Docker" for security. What I mean is that this would be the one tool that everyone would know they should use

- I wish there was a standardized bundle for things that have a large community that everyone knows they should use for security
- Some of the security tools today are for too specialized systems and they are expensive and not many people have competence in using them
- If everyone converged more towards "this is what everyone is using" so that the tools become solid and easy to use that would be an ideal DevOps-scenario for me
- Almost all developers know some Docker but if you ask for people who know Checkmarx or something like that it is almost nobody

.7 Appendix G - Codes

Below are the initial codes retrieved from each interview.

Codes - informant A

- Abstract away platform, Developer operations involvement, Automate privacy, Automation, Early involvement, Developer tunnel vision, Code volleyball, No DevOps standard, Developer anarchy, Flat hierarchy, Replaceable Tools, Tools inadequate , In-house security, External security team logs, Security is abstract, No security metrics, No one wants security responsibility, Lack of security knowledge, In-house training inadequate, Security battle, Security push-pull, Risk-cost battle, Criticize Developers, Talking about security, Amazon security feature, Convincing people to care about security, Technical Debt, Moving security to the left, Lacking specialists

Codes - informant B

- Adapt DevOps mindset, Moving to cloud, Automated security tests, List of tools, Security Mindset, Frustrating security, Tool-mindset combo, External security team metrics, Developers laughing at security, Security propaganda, Developers lacking autonomy, Security-developer divide, Security just for show, Talking about security, Removing security voluntarily

Codes - informant C

- DevOps not new, Having a standard for DevOps, DevOps tool list, Automating everything, No security automation, Everyone responsible, but everyone specialized in something else - contradiction, No measurements, Security push-pull, Lacking specialists, Cloud gives security, Nice-to have tools, not necessary

Codes - informant D

- Solo DevOps, Management non-understanding, DevOps legacy difficulty, Dependency Management difficult, DevOps time-consuming to set up, DevOps deployment benefit, Obvious tools, No security automation, Nobody works with security, Security caring, Talking about security, No DevOps standard, Unique stacks

Codes - informant E

- DevOps continuum, DevOps = automation, Solo DevOps, DevOps related to cloud, Cloud providing security, Security battle, Nobody responsible for security, Security push-pull, Nobody responsible, everyone responsible contradiction, Talking about security, DevOps = cloud, Lacking tests

Codes - informant F

- Autonomous teams, Early in change-making process, DevOps improves security, Finding mistakes earlier, Difficulty managing dependencies, DevOps = parallel mechanisms, Moving security to the left, Secure design, Thinking security early, Security-development merge, Tools not enough, Knowledge/Training most important, Developer in-consistent security knowledge, A lot of measurements, Measuring training, Monitoring time to repair, Monitoring pen-test fails, Hurrying ruins security, No external security team, Security champion, Champion = most training, Security bottleneck, Developer autonomy, Avoiding security, More training, Tools not enough, Caring about security, Not compelling security reasons, Security natural part of development, Interest in security, Security awareness, No standard, One ultimate tool, Lacking competence