Universiteit
Leiden
The Netherlands

Executive Master Cyber Security Thesis

DevSecOps practices in CI/CD pipelines –

from reactive practice to proactive handling

Sylvester Klijnstra

Leiden University

Institute of Security and Global Affairs (ISGA)

2023/2024

Page Intentionally left blank

Preface

From 2022 to 2024 I have with intense pleasure participated in the Executive master's program Cybersecurity at Leiden University, the Netherlands. During this course I have met interesting people, and gained new knowledge and experiences that I can use in my daily work within the cybersecurity sector. This master's thesis is the cap stone of the program, but before continuing with the thesis I would like to express my gratitude towards a few individuals that helped me throughout the writing process. At first, I would like to thank my supervisor Dr. T. van Steen for his tremendous help, and great guidance during the writing of the thesis. Furthermore, I would also express my gratitude towards the other professors, and guest speakers for sharing their knowledge and interesting topics throughout the program. Also, within the cohort, I was fortunate enough to get to meet like-minded people from a variety of organizations/sectors.

I would also like to thank my previous manager, <redacted>, and the organization of SSC-ICT for making it possible to study at Leiden University. Lastly, I would also express my gratitude to my family and friends for their support during the study and thesis writing, especially <redacted> for his great feedback.

Abstract

At the heart of this comprehensive study lies the central research question: "*What are the key security-related obstacles that organizations encounter when integrating DevSecOps/Secure DevOps, and how can these security challenges be effectively resolved or mitigated?*" The primary achievement of this work is the identification and categorization of these challenges into four interconnected domains: People, Tools, Values, and Governance & Processes. This identification has been done via literature reviews and by employing interviews with members of the development-, security-, and operations teams, within a specific Dutch governmental IT organization. The categorization of the identified challenges has been performed via a cyclical coding process known as the Grounded theory from Strauss and Glaser. By shedding light on these four critical areas, this study provides a roadmap for organizations to navigate the complexities of a DevSecOps implementation.

In the realm of people, cultural transformation emerges as a critical need, demanding a profound shift in mindset and behavior. This transformation encompasses the cultivation of trust, the nurturing of transparency, and an unwavering commitment to continuous learning. However, it extends beyond abstract ideals to encompass tangible actions. Team skill development becomes paramount, equipping personnel with the expertise needed to bolster the security posture. Concurrently, the cultivation of security awareness takes center stage, ensuring that every team member understands their role in safeguarding the organization. Ownership and accountabilities are clearly defined, reinforcing the importance of fostering a culture deeply rooted in security.

As we delve into the Tools domain, we find that the choices made here have far-reaching consequences for the DevSecOps journey. Thoughtful tool selection is not merely a matter of preference; it is a strategic imperative. Overreliance on tools can lead to vanity and missed vulnerabilities, making it essential to strike a balance between automation and human judgment. Simultaneously, proper access controls must be carefully enforced to prevent unauthorized access to sensitive resources. Addressing legacy systems and grappling with technical debt necessitates incremental improvements, gradually modernizing the technological landscape while preserving security integrity. Continuing with the next domain, Values serve as the compass guiding organizations through the complex DevSecOps terrain. Continuous monitoring is an unwavering commitment, offering real-time insights into the security posture and identifying potential threats promptly. Striking the delicate equilibrium between speed and security highlighting the importance of avoiding undue haste within the team that might compromise safety. Embracing a culture of continuous improvement propels organizations forward, encouraging iterative enhancements and the evolution of security practices. Within this framework, DevSecOps control takes root, fostering a disciplined and principled approach to security integration.

In the Governance & Processes domain, the foundation for a secure and collaborative culture is laid by securing management buy-in. Leadership commitment paves the way for organizational alignment and support, making security a top priority. The intricacies of compliance and regulations are diligently addressed through the integration of compliance experts and automated compliance checks. This ensures that DevSecOps practices are not only effective but also adhere to legal and industry standards.

In essence, the challenges within these domains are not isolated silos but are indistinguishably intertwined. Success in the DevSecOps journey hinges on fostering a culture of security (People), making strategic choices about tools (Tools), upholding core values (Values), and establishing governance and processes (Governance & Processes) that collectively form a resilient and robust framework for secure software development and delivery.

Keywords

DevOps, DevSecOps, SecDevOps, Secure Development, Operations, Security, Best practices, Challenges, Continuous Integration and Continuous Delivery, CI/CD Pipeline, Agile, Shifting Left, Shifting Right, IT, Workspace, Software Development Life Cycle (SDLC).

Contents

1. Introduction

In our highly technologized society, there is an increased need for rapid software development and adoption of systems, appliances, and applications. This rapid software development/adoption requires a flexible and Agile approach, and as such in recent years the Agile approach has become increasingly more popular in the software development sector over the more generally linear Waterfall method [1]. Both Agile and Waterfall are two unique project management methodologies with the aim of completing projects or work items within a project. Agile is an iterative method that operates in collaborative cycles, also known as sprints. While Waterfall is a historic sequential method, that can also be performed in a collaborative way, the tasks however are normally more done in a linear manner. The need for flexibility and agility is the reason that in modern software development, the Agile way of working has mostly predominantly replaced the Waterfall approach [1].

1.1 Agile

Since the Agile way of working enables (new) processes such as program increment (PI, adaptive) planning, early delivery, and continuous improvement, it also introduces new problems that requires solving to support continuous delivery in a fast and short software development life cycle, in which demanding customers/end users requires high quality of the software provided. These problems and the gap between continuous development and the current production IT environment can be addressed with the DevOps collaborative framework.

Figure 1 DEVOPS framework [2]

1.2 DevOps

DevOps is a way of working, in which Developers and IT operations do collaborate and no longer work separately from one another, with the goal of providing faster and better software with the aid of automation tools [2]. While for a functional perspective, this fast and better delivery of software seems great, from a security perspective this raises concerns about the overall security of organizations in relationship to the tools and processes used by both teams from end to end. Also outdated security can lead to exploitable weaknesses and subsequently risks for both the software supplier and its customers.

As IT security must play an integrated role in the entire life cycle and security is a responsibility that must be shared, security as a foundation must be anchored in the DevOps framework. We call this integration and shared security responsibilities, “DevSecOps” or “Secure DevOps,” in which security as an important mindset becomes an integral part of the

framework. In short DevSecOps is the theory or philosophy of adopting security practices like security by design, privacy by design, and compliancy by design within the current DevOps processes. This theory/philosophy is also used to describe the Software Development Life Cycle (SDLC), in which security and continuous delivery are the core focus.

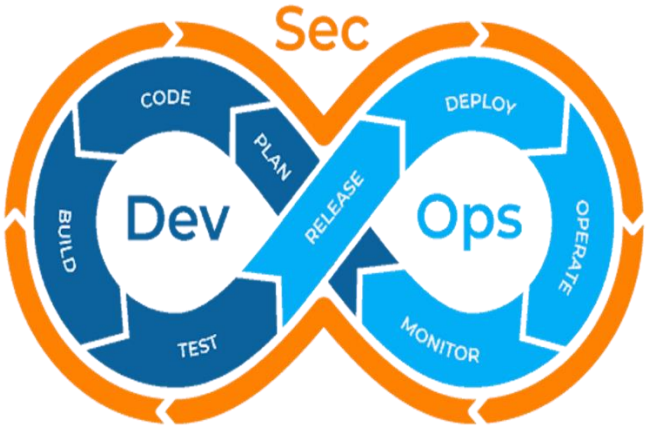

Figure 2 DevSecOps [3] – Security as an integral part of the DevOps framework

1.3 Motivation

What is noticeable, is that in DevOps practices, security often is treated as a secondary system, as security is seen as a burden towards flexibility, functionality, and productivity. Security comes often at the end of the SDLC, when the developers must require approval from security to deploy the newly created software or (pre-)packaged software to the production environment. As security vulnerabilities could be present in the (source)code of for example an application, it can be very frustrating to discover these vulnerabilities at the very end of the Software development life cycle. These frustrations will surface, because security vulnerabilities do introduce risks, and it is often very difficult to correct these vulnerabilities when the software has already been deployed to production. In return this makes it also a costly endeavor to fix these risks afterwards. To overcome these frustrations there have been processes introduced in the general DevOps processes, like Continuous Integration and Continuous development (CI/CD), better known as the CI/CD pipeline. CI/CD ensures continuous testing and verification of code correctness during the entire sprint [4] [5].

1.4 Goals

Since there are many cyber threat actors, security becomes increasingly important for the protection and resilience of the business services provided. As current security practices within some organizations are often reactively present, while a pro-active security posture is required to combat present- and future cyber threats, we are aiming to find a solution to go from reactive security practice to proactive handling within DevSecOps [6]. To make this work the DevSecOps mindset/philosophy must become an integral part of the way of working, since this helps organizations to follow the principle of security-by-design, a proactive approach. To pave the way for a strategy and to understand DevSecOps, we will research what the biggest challenges are for organizations when adopting security in a Dev (Sec)Ops approach. This thesis therefore aims to answer the following research question:

"What are the key security-related obstacles that organizations encounter when integrating DevSecOps/Secure DevOps, and how can these security challenges be effectively resolved or mitigated?"

This research is based on a qualitative approach, in which two primary methods of data collection have been utilized, such as interviews and literature analysis.

1.5 Sub questions

To answer the main research question, the following sub questions must be answered such as:

- **Do DevSecOps/Secure DevOps practices impact the collaboration between security and development teams?**
- **What are the differences between a shift left approach and a shift right approach in an Agile way of working?**
- **What type of shift approach would be the most beneficial for increased security?**

1.6 Hypothesis

Beside the main research question, since DevSecOps is the theory or philosophy of introducing security within the current DevOps practices, we are curious if the adoption of security is only a change of mindset/culture or that it is more than that. According to different sources, DevSecOps, like Agile and DevOps is only a cultural issue [7] [8]. We believe that culture is of major importance within the DevSecOps approach, but it is most likely not the only issue when adopting security within current DevOps practices.

Therefore, the hypothesis is that cultural alignment when adopting/improving DevSecOps is important, but it will most likely not be the only challenge to face.

1.7 Contributions

In this research endeavor, we begin on an exploratory journey into the realm of DevSecOps, to address the pressing question of what key security-related obstacles organizations encounter when integrating DevSecOps/Secure DevOps and how these challenges can be effectively resolved or mitigated. This study adds several contributions that extent a deeper understanding of the DevSecOps landscape. To start with the categorization of challenges: This research provides a comprehensive categorization of challenges organizations face when adopting DevSecOps. These challenges are segmented into four interrelated domains: People, Tools, Values, and Governance & Processes. By organizing the challenges into these domains, we offer a structured approach for organizations to identify and address specific issues in their DevSecOps journey.

Furthermore, another contribution of this thesis is giving a holistic perspective, as the study shows the interconnected nature of these challenges. It emphasizes that success in the DevSecOps journey requires addressing all four domains simultaneously, recognizing the need for a holistic approach that integrates cultural transformation, tool selection, value alignment, and governance into a cohesive framework. The third important contribution lies therefore, in the guidance for a cultural transformation. Via the exploration of the *people* domain, we emphasize the significance of cultural transformation, and in such this study

provides insights into fostering trust, transparency, and continuous learning within teams, along with strategies to cultivate security awareness. This research also contributes actionable steps for leadership buy-in, skill development, and shared ownership, which are essential for nurturing a culture rooted in security.

Fourth, progressing in the strategic tool selection: In the *tools* domain, we highlight the strategic importance of choosing the right tools for the DevSecOps journey. By advocating a balanced approach between automation and human judgment, this research guides organizations in avoiding the pitfalls of tool overreliance. It also emphasizes the importance of proper access controls and managing legacy systems and technical debt. Continuing with the fifth contribution of value alignment: The study provides insights into the Values domain, emphasizing the significance of continuous monitoring and balancing speed with security. It contributes to the culture of continuous improvement, promoting iterative enhancements in security practices. DevSecOps control is introduced as a guiding principle in this journey. As DevSecOps requires an organizational culture that receives a structured governance, and support from its upper management, Management Buy-In and Compliance Expertise is therefore seen as the sixth contribution: In the Governance & Processes domain, this research signifies the importance of management buy-in, which paves the way for organizational alignment and support for security. The integration of compliance experts and automated compliance checks ensures that DevSecOps practices are not only effective but also compliant with legal and industry standards.

Overall Guidance: This study offers organizations a comprehensive roadmap to navigate the complexities of DevSecOps implementation, contributing to their success in secure software development and delivery. It serves as a vital guide for organizations seeking to thrive in the ever-evolving landscape of DevSecOps. These contributions collectively form a valuable resource for organizations, guiding them in achieving the highest standards of security, efficiency, and collaboration in their DevSecOps endeavors.

1.8 Reading Guide

The first part of this thesis will focus on the details of the DevOps approach. This consists of, the purpose, and origin of DevOps, the main characteristics and how these practices leverage on the Agile way of working. Also, the CI/CD pipeline, and several identified common challenges of the DevOps approach will be described. The second part will be focusing on the overarching security within DevOps/DevSecOps-cycle as described. First the Software Development Lifecycle (SDLC) will be mentioned, as this model has been used to further research the security elements that are part of DevSecOps. Furthermore, the supportive pillars of DevSecOps, and the three lines of defense model will be pointed out. Finally, to round this security chapter up, the concepts of shifting left, and shifting right will be described as these are part of the continuous monitoring parts of DevSecOps. Subsequently in chapter 4, will we focus on the results of the literature research, and in chapter 5 are the results from the interviews described. These results do provide the foundation for answering the research questions as mentioned earlier. The answering of the research questions will be done in the discussions and conclusions section of this thesis. Also, proposals for future work are part of this last chapter.

2. Methodology

This chapter outlines the methodology used within this research to investigate the challenges faced by organizations when adopting DevSecOps and to propose viable solutions to these challenges.

2.1 Research approach

This research has been performed via a qualitative research approach, that employs two primary methods of data collection. The researcher has chosen a literature analysis and semi-structured interviews as data collection methods. By conducting interviews and analyzing literature, qualitative data has been collected to gain valuable insights into DevOps/DevSecOps and the challenges faced when trying to adopt these practices within organizations.

2.2 Data collection methods

Two primary methods of data collection have been used within this research:

Literature analysis

A comprehensive literature analysis has been carried out on existing literature, books, whitepapers, and reports related to the DevSecOps practices. To help structure the research, the funnel technique from E. Hofstee [9] for a structured literature review has been used.

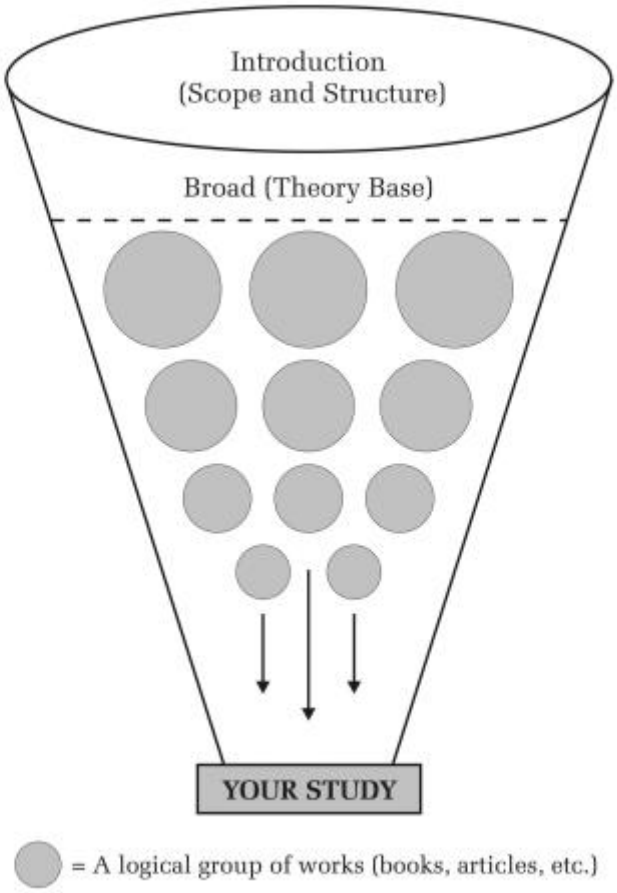

Figure 3 Funnel method - to help structure literature review [9]

The funnel technique starts from a broad perspective, in relation to the main research topic, and subsequently works towards a more specializing approach of researching the main components of both DevOps, and DevSecOps.

Therefore, to start with the research for this thesis, the concept of DevOps has been researched to create a broad foundation. It is important to first gain broad knowledge about DevOps, since DevSecOps is a deepening of DevOps. To gain this knowledge, the first step is to gain in depth knowledge of the core principles of DevOps, and what the goals and benefits of DevOps practices are. After that, the researcher needed to explore with what means a team can work in a DevOps way of working, resulting in automation technologies and CI/CD practices. Finally, based on this knowledge several challenges of implementing DevOps have been identified.

To deepen the understanding of embedding security, within DevOps, the DevSecOps approach has been explored in depth using four supportive pillars (Organization, Process, Technology, Governance), and the security components of the software development lifecycle (SDLC) (figure 8). A few examples of these security components are Security Monitor, Security Audit, Security Patch, Secure coding, Security as Code, and others [10]. Again, just like the approach employed for DevOps, we start by founding out, what the DevSecOps concept defines, and which pillars do support this concept. Also, to extend the reach of the pillars, the 3LoD model have been described, to shape more context around the different layers of governance, and the differences between responsibilities and accountabilities. Furthermore, as both Continuous Operation and Continuous Monitoring are important to understand why there is a need for continuous testing, we will also cover Application security Operations.

Interviews

In chapter “Results of Interviews” will we discuss the used interview method and results of all interviews combined.

3. Background

In this chapter, both DevOps and DevSecOps approaches will be researched and described in detail. In addition, the relationships between the two approaches and the (common) challenges of adopting DevOps/DevSecOps are described in this chapter. This has been done with the goal of shaping the foundation for this research, so that well-considered answers to the research questions can be given.

DevOps

The history of DevOps can be traced back to around 2007 [11], when the separate communities of software development and IT operations raised concerns regarding traditional software development practices [12]. These concerns were raised because a dysfunction was found in the model used for traditional software development, because developers who wrote the software source code, entirely worked separately from IT operators who needed to implement and support the software and its source code. Therefore, the term DevOps is the abbreviation of the combining of the two different disciplines “development and operations”. DevOps can be seen as a set of procedures, tools and a cultural philosophy that tend to integrate and automate the processes between the development team and the IT operators responsible for the production environment [12]. The goal of DevOps is to create a unification culture that, in turn, results in better and faster delivery of software by emphasizing team empowerment, collaboration, and communication between the developers and IT operators, as well as the technology automation used. Other benefits of DevOps do involve “increased trust, easier releases, team efficiency, increased security, the possibility of solving critical problems quickly and greater management towards unplanned work” [12]. This leads to higher-quality products, more satisfied customers, and both teams more involved in the entire process from end-to-end.

To achieve this, the human factor or creating a collaborative culture in which employees that trust each other and share the same objectives, instead of conflicting objectives is considered as the biggest success factor of DevOps [13]. In an open and transparent culture in which feedback between teams can be freely shared, this makes members of for example the development team more aware of how their performed activities affect their colleagues engaged in the process of releasing software. If there is a lack of visibility between teams, or if the goals of both teams do not align with one another, this can cause different priorities, finger pointing and in result mentally leading to poorer quality and lower velocity of creating software and implementing it [12]. By changing the comprehensive approach of only looking at the development processes, DevOps does greatly contribute to overcoming obstacles between both developers and IT operators at an early stage.

3.1 Core DevOps practices

As seen in figures 1 and 2, DevOps is not focused on a single area of software development. DevOps is a dual-purpose way of working, that embodies a systematic and scientific approach to software development and delivery. Within this paradigm, DevOps orchestrates a harmony of practices across the eight interrelated dimensions of the development- and operations pipeline stages [14].

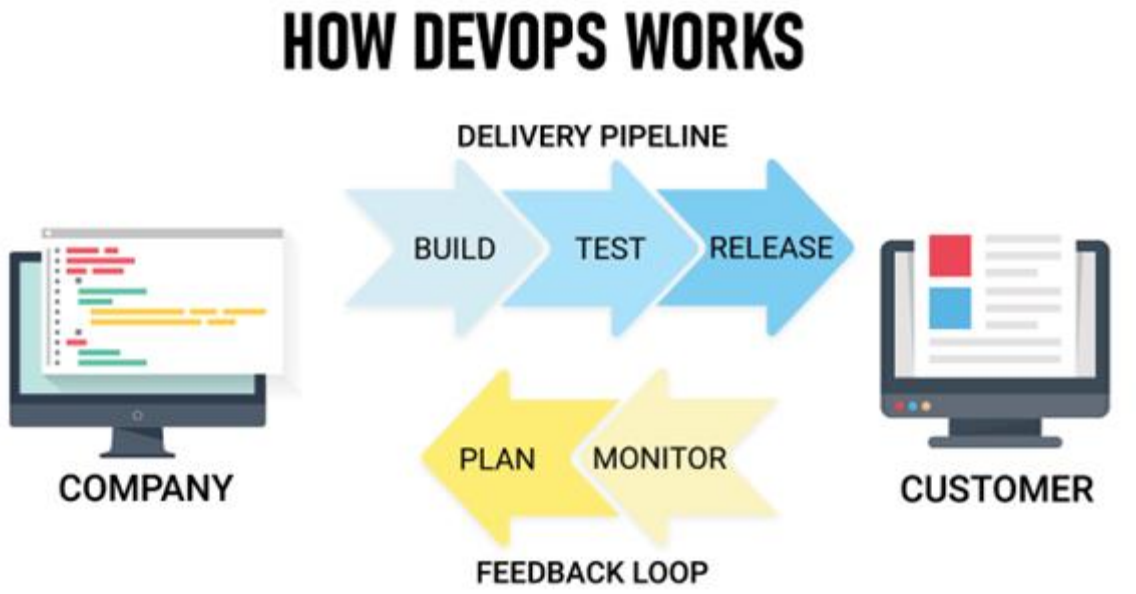

Figure 4 How DevOps works in a technical sense - [15]

Development Pipeline Stages

Each development project starts with a plan, and as such *planning* serves as the foundational phase of DevOps [14]. Planning is like creating a roadmap, in which project planners lay out project goals, technology needs, and architectural plans. Decisions about software and tools are like choosing the right tools for a job. The product owner's role is setting clear goals to guide the developers throughout the development process. When the goals are clear and concise, then in the *Coding* phase this is where the actual work happens. In an iterative way, piece by piece, feature by feature. The development team gets creative, and this is where automation tools can help make things faster, more efficient, and repeatable.

As soon as the code build is ready, then in the *building* phase it is time to put all pieces of the puzzle together. In this phase the code components get assembled in a separated testing area, and any problems found during this phase are like discovering coding mistakes. And finally, in the last development pipeline stage of *Testing*, will the software build have subjected to rigorously experimental testing. Automation tools assume the role of scientific instruments, ensuring adherence to expected behavior while detecting deviations, analogous to observing experimental results. Tools like Loadgen and Ranorex are instrumentations for precise measurements, including assessments of system performance under varying conditions.

Operations Pipeline Stages

When the software build passes the rigorous *testing* phase, then collaboration between development and the operations team is of most significance. During the operations pipeline stages, both the software build, and the production environment are getting ready for adoption and integration. During the *release* phase the software build transitions into the peer-review phase by making sure that the work of the development team is reviewed via

the 4-eyes principle. The operations team checks if the software is ready for use, like making sure the research is approved by peers. They also make sure it follows all the rules and standards. When the peer-review is over, then in the *deployment* phase the findings of the operations team will be shared. This is also the phase in which software becomes available for people to use. However, before the software is shared there are some operational checks performed to make sure everything is okay for adoption of the new software.

When the software has been adopted in production and shared to end-users then in the operating phase, will the software be managed after it's out in the world.
The operations team looks after the software in its working environment, a bit like how technicians take care of machines in a factory. They use automation tools to help with this. And finally, in the last phase of DevOps we have the *monitoring* phase. Monitoring is like keeping an eye on things, to ensure normal operations. After the software is being used, it's important to gather information about its performance, but also how the software is experienced by the end-users. Feedback of end-users can be used to continuously improve the process. DevOps is therefore a methodical way of developing software. It's a structured process, that helps to make sure the software keeps getting better to meet the changing demands of end-users.

3.2 The Agile approach

In a fast-changing world, DevOps must be highly flexible to accommodate sudden and imminent changes requested by users. This flexibility is incorporated via the Agile approach, an approach built on the Waterfall approach. One of the drawbacks of Waterfall is that there is no support towards iterative and incremental development of features and functions incorporated. As customers and other stakeholders are becoming increasingly demanding, also their demands are constantly changing, this requires an incremental and iterative software development that can be highly flexible and Agile. Incremental development is an approach in which the product gets separated into fully working bite-sized chunks, called increments. Every increment gets subsequently released on top of other increments and existing functionalities [16].

While iterative development is the approach in which teams do tend to build on functionalities and features, they will however not wait for all features and functions to be complete, prior to releasing the software. Software as a minimum viable/basic product gets released and improved with future releases. The Agile way of working does strengthen the DevOps principles, because both the developers and IT operators can rely on the before mentioned pre-determined steps and validations such as continuous delivery and continuous integration, by testing the software from end-to-end via automation tools. Both Agile and DevOps do fill the gap between both teams, caused by the lack of communication and collaboration. As it is imperative that IT operations and the development teams, do frequently communicate this ensures increased security and stability of the IT environment. As a result, this leads to higher customer satisfaction and less disturbances when releasing software [17].

3.3 Goals and benefits of DevOps

Several benefits and goals of DevOps have been found and assessed, to assess the benefits of DevOps practices even further, we draw insights from the Atlassian Survey conducted in 2020, which sheds light on the trends in DevOps. The findings are particularly revealing, with over 99% of respondents attesting to the affirmative impact of DevOps on their respective organizations [12]. These benefits span several key dimensions, including “the acceleration of software development and delivery processes, the enhancement of collaborative efforts, heightened operational efficiency, the assurance of software quality and reliability, and the fortification of security protocols”. When successfully implemented, these advantages improve operational excellence and elevate customer satisfaction.

One of the notable advantages of DevOps is the acceleration of software development and delivery. In the context of DevOps, speed is characterized by the ability to expedite software releases while simultaneously maintaining high standards of quality and stability. This phenomenon aligns with findings from the Google-backed "DevOps Research and Assessment team" in 2019, which demonstrated that proficient DevOps teams achieve remarkable deployment rates, with software deployment being 106 times faster compared to slower performing teams [12]. If we take look at a fundamental characteristic of a well-established DevOps culture, this can be found in the optimization of collaboration within and across teams. In such a context, responsibilities are shared, and work processes are structured to ensure seamless end-to-end workflows. The outcome of these collaborative efforts is increased efficiency, translating to time-saving benefits, particularly evident during activities like coding and development. In result, the DevOps practices have a direct impact on the pace of software implementation. A structured approach to frequent software releases ensures a competitive edge by rapidly delivering new features and promptly addressing bugs to end-users. To accommodate all these benefits, the cornerstone of DevOps is the so-called Continuous Integration/Continuous Deployment (CI/CD) pipeline. The CI/CD pipeline is a practice that upholds the uncompromised quality and reliability of software. CI/CD mechanisms guarantee that changes to software are rigorously evaluated to anticipate any disruptions to its functionality. Continuous monitoring further enhances software quality, providing continuous feedback and allowing for the prompt identification and resolution of any issues that may arise.

Finally, within a DevOps pipeline, security assumes a significant role in the form of DevSecOps, a specialization within DevOps, that ensures that security practices are embedded into the very fabric of the product development process. This entails (enforcing) security by design, incorporating security controls, executing security audits, and conducting extensive testing across Agile development and DevOps workflows. In essence, security is not a mere afterthought but an integral aspect of the entire development lifecycle. The research will go into depth about security in the DevSecOps subchapter.

3.4 CI/CD DevOps practices

If we delve further into this DevOps pipeline, there are three distinguished continuous practices that all serve software development needs in a unique way. These continuous practices are “Continuous Integration (CI)”, “Continuous Delivery” and finally, “Continuous deployment” [18]. These practices play a vital role in modern DevOps methodologies, bridging the responsibilities of both development and operations teams.

Continuous Integration (CI)
To start with CI, CI revolves around the establishment of a central "main branch," often referred to as "Master" or simply "main [19]." Developers continually merge new changes via a subbranch into this main branch. The crux of CI lies in running automated tests on all code commits, such as code builds or packages (e.g., MSI/MSIX). This proactive approach mitigates integration challenges that may otherwise surface during release, a scenario often termed "integration hell." Nonetheless, the downside of CI entails manual labor in crafting automated testing scripts and maintaining them for new features, fixes, or alterations introduced by developers. A CI server is also indispensable for all three practices, requiring continuous monitoring of the main repository and automatic test execution whenever new code commits are made. Developers must, however, merge their changes frequently, ideally at least once a day. The primary gains of CI lie in significantly reduced automation testing costs. CI servers can literally execute hundreds of different tests within seconds, resulting in fewer reoccurring bugs and defects reaching customers [18]. This also allows the QA/testing team the opportunity to focus on more critical aspects, as they spend less time manually conducting tests.

Continuous delivery (CD)
To continue with an extension of CI, Continuous delivery (CD) can perform tasks automatically, such as the automatically deployment of code changes towards various environments (Development, Testing, Acceptance, and Production - DTAP*)*. Beyond automated testing, CD allows developers to release and deploy applications with a simple mouse click, and the release frequency can align with business needs, whether daily, weekly, or another preferable cadence. Furthermore, small, incremental releases simplify the identification and troubleshooting of incidents and issues. Like CI, CD demands a robust CI foundation that adequately covers the codebase. It also requires a test suite and may necessitate the use of feature flags to accommodate changes in team workflow. While the deployment process must be automated, manual initiation remains necessary, with subsequent steps executed automatically. The key advantages of CD include the elimination of complex software deployment preparations and the ability to gain more customer feedback by increasing release frequency.

Continuous deployment (CD)
If we look at another CI/CD practice, Continuous deployment is somewhat the same process as Continuous delivery, however the deployment to production is done differently in this practice as all changes that do pass each stage of the CI/CD pipeline, do gets released towards customers without human intervention of the development team or operations team. Only a failed test in one of the other stages will prevent the actual deployment of the software to the production environment. Continuous Deployment offers a significant boost to the feedback loop with customers and relieves the development team by eliminating traditional release days. Code can be rapidly integrated into production within minutes. However, due to the fully automated nature of CD, the quality of required tests must be exceptionally high.

The primary advantage over Continuous Delivery lies in the potential for further accelerating the development and release process, as there are no interruptions caused by manual checks and releases [18].

CI/CD practices, how do they relate to one another.
In figure 5, a graphical overview of the differences of the three CI/CD practices is shown. In this figure it is notable that continuous integration is part of both the continuous delivery and deployment practices. While the difference between continuous delivery and deployment, is the fact that continuous deployment is basically like continuous delivery, however in a continuous deployment the releases do happen to be performed automatically, instead of manual deployment that takes place in continuous delivery.

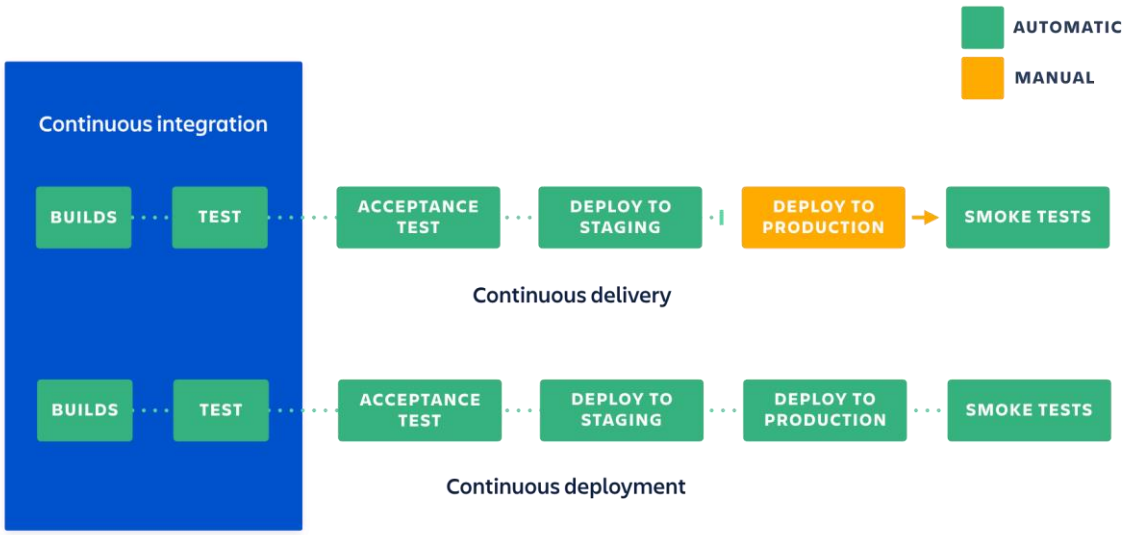

Figure 3 CI/CD relationships and differences [18]

Automation technologies
To suit the CI/CD needs, DevOps requires automation technologies to be fully utilized, and a multitude of tools do facilitate this automation [20]. The following tools are some examples of these automation tools. For example, for Version Control needs, Git (Gitlab, GitHub, Bitbucket) is a well-known tool that is widely used by developers and Git is characterized with dynamism and collaboration via a development community. When it comes to building code, Apache Maven, often referred to simply as Maven, steps into the limelight. Beyond its coding functions, Maven takes on an expansive role encompassing reporting, documentation, distribution, releases, and software dependency management. The term "Maven" itself carries a profound implication, translating from Yiddish as an "accumulator of knowledge." This tool streamlines the intricate process of feature migration by simplifying development endeavors.

An example of a CI tool that defines its domain is Jenkins. Jenkins's distinguishing feature is its adaptability, catering to both internal- and plugin extensions. Given that CI/CD forms the backbone of DevOps, Jenkins emerges as an indispensable ally. It boasts an extensive library of over 1500 plugins, bestowing customizable functionalities that seamlessly integrate into

the development process. This extensive compatibility renders Jenkins a preferred choice for DevOps practitioners [20].

For the orchestration needs of project configurations, developers frequently turn to Ansible, Chef, and Puppet. These open-source tools wield versatility and are harnessed for diverse deployment, automation, and orchestration tasks. They also possess intrinsic cloud capabilities, such as the concept of "Infrastructure-as-Code (IaC)," streamlining the automation of infrastructure configuration. The native support for cloud environments further amplifies their appeal, rendering them favorites among DevOps teams. Other useful tools are virtualization and containerization platforms, like Docker and Kubernetes, that are indispensable assets within the DevOps ecosystem. These tools play an important role in enhancing the efficiency of application development. Containers, encapsulating libraries, source code, and more, furnishing comprehensive runtime environments tailored to the needs of specific applications. Beyond efficiency, these platforms augment security and governance, casting a protective mantle over entire software projects. However, container technologies also introduce new security risks that requires decent risk analysis, thus using such platforms does not suggest that security considerations are no longer required.

In addition to these instrumental tools, the DevOps landscape is loaded with an array of project management tools, as illustrated in Figure 6: DevOps Automation Tools. Platforms like Jira, Confluence, and Trello facilitate streamlined project coordination and collaboration, enriching the DevOps experience with enhanced project management capabilities.

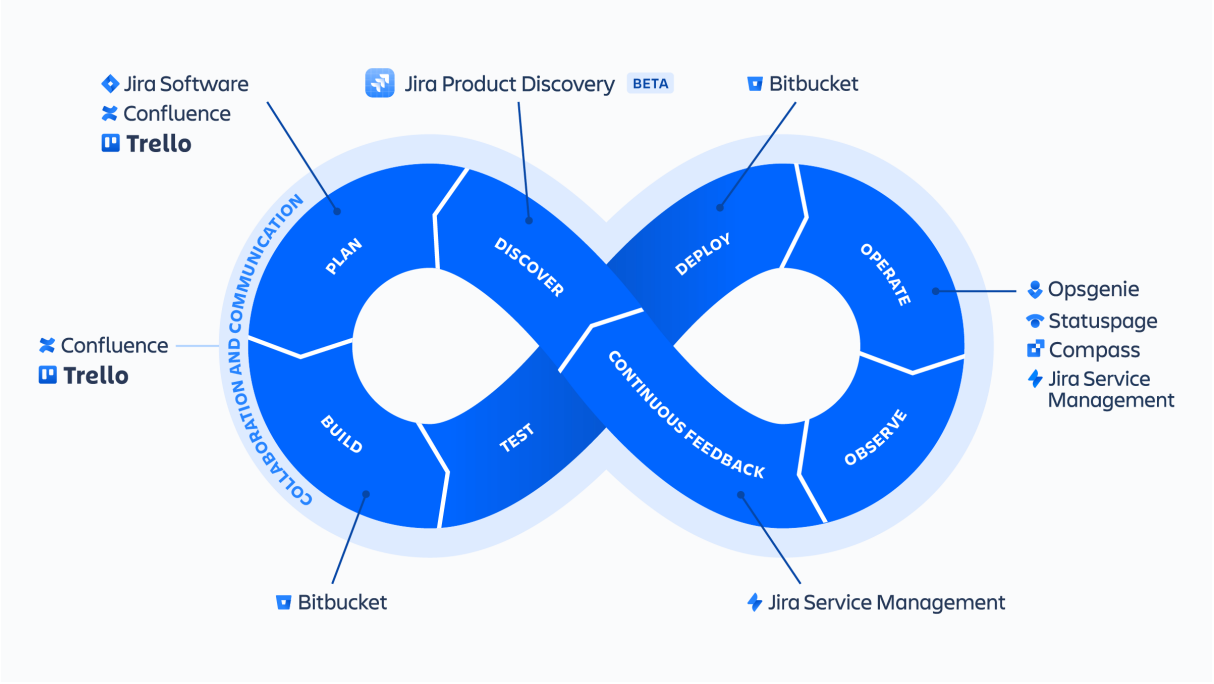

Figure 6 DevOps- automation tools [12]

3.5 Challenges of implementing DevOps

According to Robert Krohn, the Head of Engineering for DevOps at Atlassian, DevOps isn't the responsibility of a single individual; it's a collective duty shared by all members of both development and operations teams [12]. However, breaking established habits, routines, and values can be exceptionally challenging. Teams accustomed to working in isolation may encounter difficulties when adapting to structural changes or may even resist the adoption

of DevOps practices [12]. It's important to note that DevOps isn't merely about incorporating new tools; it's a fusion of three fundamental elements: people, tools, and culture. Each member of a DevOps team must possess a comprehensive understanding of the value stream that encompasses the entire DevOps flow, from ideation through development, involving activities, individuals, systems, and the flow of data and information, all with the ultimate objective of delivering an exceptional end-user experience [21]. While DevOps certainly involves the utilization of tools, it's essential to establish a solid foundation around core DevOps stages, including automation, configuration management, and CI/CD procedures. An excessive reliance on DevOps tools can however divert the focus of DevOps teams from building this essential foundation [12]. Therefore, the challenge lies in reshaping the organization's culture, restructuring development teams, and adapting internal processes to align with the requirements of the new DevOps model.

Another significant challenge in DevOps integration is the absence of a robust security mindset and approach. When security isn't seamlessly integrated into DevOps practices, it can introduce substantial security risks. To address this, the security team must be an integral part of the DevOps process, from start to finish, ensuring the delivery of high-quality, reliable software suitable for integration into the production environment. However, since DevOps teams often prioritize speed and flexibility over security, there's a risk of neglecting essential scans and code verification checks. This negligence can lead to the introduction of vulnerabilities and security holes, which may remain unnoticed unless the security team is actively engaged [12] [22].

Cloud first strategy

Transitioning legacy systems and modernizing outdated applications within the framework of DevOps architecture in a cloud environment presents its own set of challenges. Embracing a "Cloud-First" strategy, whether through Infrastructure-as-Code (IaC) [23], or other cloud services like Platform-as-a-Service (PaaS), can introduce hurdles for organizations striving to implement DevOps effectively [22]. These hurdles often arise from skill shortages, legacy architectural constraints, resistance to organizational changes, and limitations within automation tools or a lack thereof.

CI/CD challenges

CI/CD practices, driven by the desire for speed and efficiency, necessitate careful considerations of how to automate repetitive tasks with minimal human intervention and minimal error margins. In practice, CI/CD presents a complex landscape with unique complexity. The pursuit of agility and rapid development must therefore be balanced with the imperative for robust testing and validation, all while ensuring seamless integration into the broader DevOps framework. These difficulties make CI/CD a particularly challenging endeavor, demanding careful planning and execution to attain its full benefits [24].

Summary of DevOps

In the theoretical framework, this research delves into DevOps, the foundation of DevSecOps. Exploring the details and the challenges associated with adopting DevOps. DevOps emerged in response to the need for improved collaboration between software development and IT operations, which traditionally worked in isolation. DevOps integrates procedures, tools, and cultural philosophies to automate and harmonize the processes

between development and IT operations. It emphasizes collaboration, communication, and technology automation, aiming to accelerate software delivery while enhancing quality and security. The human factor, by fostering a collaborative culture with trust, is a key success factor in DevOps. A transparent culture with open feedback channels helps align the goals of development and operations teams, reducing conflicts and improving software quality. DevOps encompasses a range of practices across eight dimensions, starting from planning and extending through coding, building, testing, releasing, deploying, operating, and monitoring. These practices involve iterative development, automation, and continuous integration/continuous delivery (CI/CD) pipelines. The Agile approach complements DevOps by enabling flexibility and incremental development. Agile focuses on delivering software incrementally and iteratively, ensuring alignment between development and IT operations and improving security and stability. Other DevOps benefits, include faster software delivery, enhanced collaboration, operational efficiency, improved software quality, and strengthened security.

Within the CI/CD pipeline, Continuous Integration (CI) involves regularly merging code changes into a central branch and running automated tests, while Continuous Delivery (CD) automates deployment to various (DTAP) environments. Continuous Deployment (CD) takes automation a step further, deploying changes automatically to production. These practices contribute to software quality and faster delivery, via automation tools like Git, Maven, Jenkins, Ansible, Chef, Puppet, Docker, and Kubernetes. These tools facilitate version control, build management, continuous integration, configuration management, containerization, and orchestration. However, implementing DevOps presents several challenges. Cultivating a DevOps culture requires overcoming resistance to change and fostering collaboration between traditionally siloed teams. Furthermore, transitioning to a cloud-first strategy poses skill shortages, legacy system constraints, resistance to change, and automation tool limitations as challenges. Additionally, CI/CD introduces complexities due to the need to automate repetitive tasks with minimal human intervention while maintaining rigorous testing and validation.

In conclusion, DevOps offer substantial benefits but requires a careful cultural transformation, skill development, and tool implementation to address the challenges and realize their full potential in accelerating software delivery and enhancing quality and security. Integrating security into DevOps practices is crucial to mitigate security risks, but teams sometimes prioritize speed over security, leading to potential vulnerabilities to be exploited. In the next section, will we deep dive into the realm of security within DevOps, or rather DevSecOps. In such, security related challenges within DevSecOps will be explored.

3.6 DevSecOps

In this subchapter, we embark on an exploration into the realm of DevSecOps, an abbreviation that combines, “Development,” and “Operations,” together with “Security”. DevSecOps is a paradigm that tightly weaves security into the fabric of DevOps. This exploration is part of the broader spectrum of DevOps, where efficiency, collaboration, and automation reign supreme. The red thread that runs through DevOps and DevSecOps is the relentless pursuit of excellence in software development. DevOps, as we know it, is all about breaking down silos, fostering collaboration between development and operations, and automating processes to achieve faster and more reliable software delivery. But it became evident that within this framework, security was often treated as an afterthought, lurking in the shadows, ready to strike when vulnerabilities were discovered. This disconnection between DevOps and security gave rise to the need for DevSecOps.

Incorporating the security by design principle from the very beginning, or "shifting left," as we briefly discussed earlier, is the essence of DevSecOps. It's about recognizing that security is not an impediment to speed but rather an enabler of it [25]. By addressing security concerns at the outset in the entire CI/CD pipeline, organizations can prevent costly and time-consuming issues down the line. This shift left mentality runs parallel to the broader DevOps philosophy, where early collaboration and automation accelerate delivery [26]. This exploration of DevSecOps will be done via the four supporting pillars of DevSecOps—Organization, Process, Technology, and Governance, as illustrated in figure 7. Each pillar is an integral component of DevSecOps, which corresponds to the areas of DevOps wherein challenges have been found.

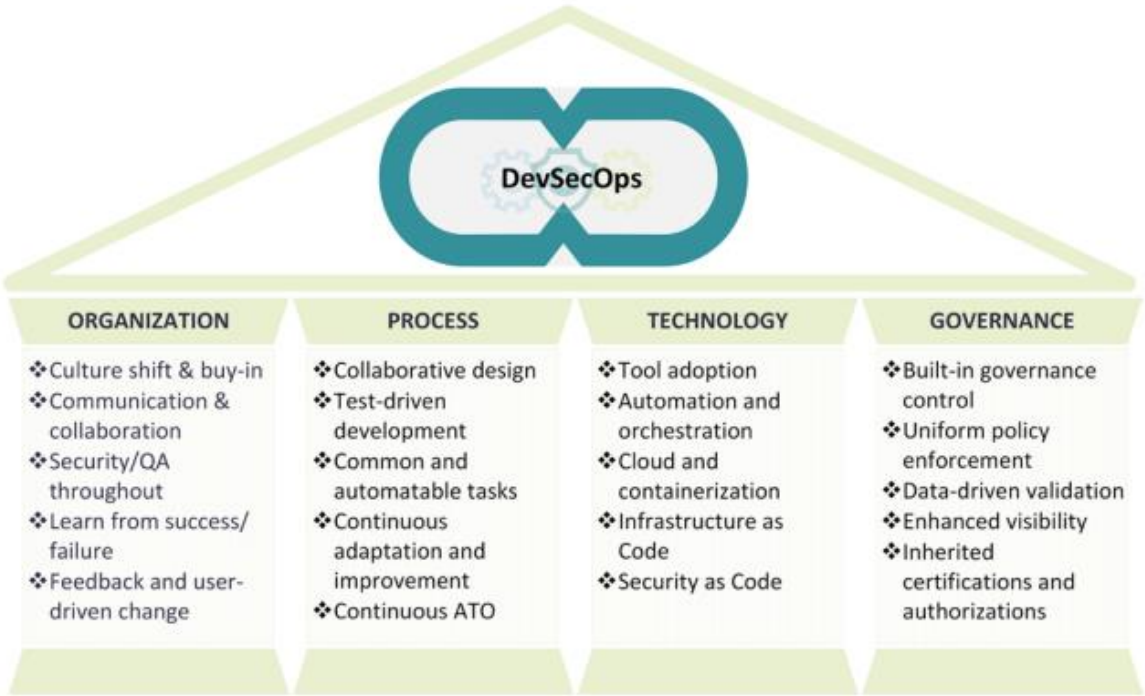

Figure 4 DevSecOps pillars [10]

DevSecOps pillars

The exploration into the world of DevSecOps starts with a significant step: securing the buy-in of senior management within an organization [10]. This essential commitment from the

upper echelons of leadership sets the stage for a transformative shift in culture, but also for the philosophy, and mindset, which is fundamental to the DevSecOps principles. DevSecOps is a journey that not only reshapes the way processes unfold but also incorporates the adoption of innovative technologies and tools for the automation needs of the development process. In essence, senior management's alignment with the right philosophy allows for a culture shift that paves the way for the development of new collaborative processes, underpinned by cutting-edge technologies.

Organizational Pillar

At the very core of adopting DevSecOps practices lies the *organizational* pillar, which necessitates a profound evolution in an organization's day-to-day activities and software development life cycle (SDLC) processes [10].

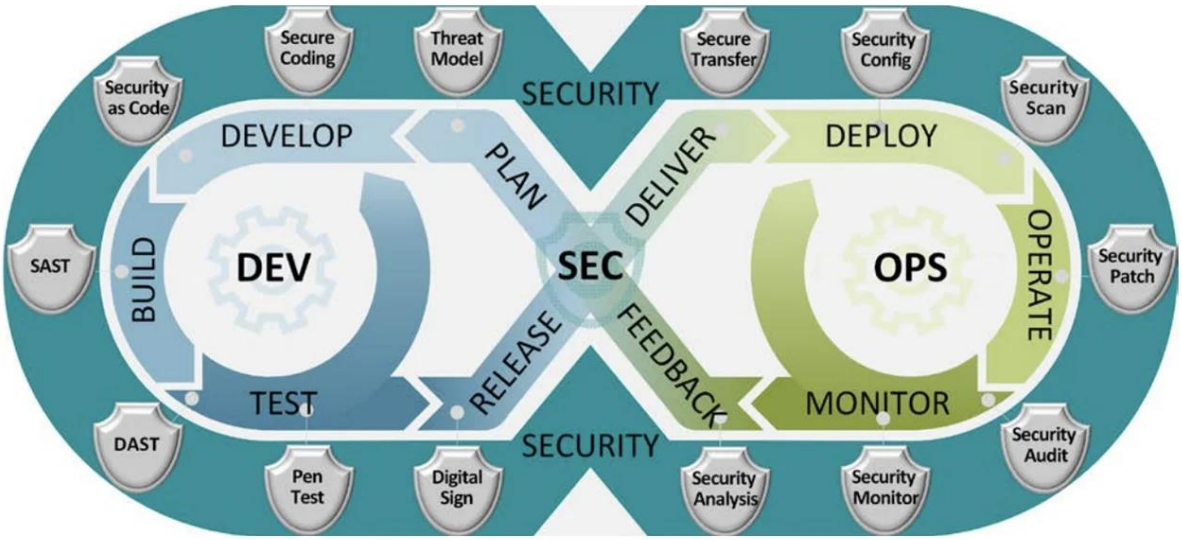

Figure 8 DevSecOps lifecycle [10]

First and foremost, as mentioned before it calls for a culture that employs a holistic view, and that encourages the sharing of responsibilities across software development, security, and operations teams. Achieving this requires dedicated training and education, so that gradual buy-in from all stakeholders involved arises. To achieve this, it is imperative that organizations should start breaking down the silos that may have once isolated these three separate teams. Communication and collaboration should become the lifeblood that courses through every phase of the software lifecycle. To facilitate this, critical information regarding security and quality assurance (QA), such as security alerts and product QA reports, must flow seamlessly and automatically to all stakeholders at each stage of the DevSecOps lifecycle. This information sharing is the bedrock upon which collaborative activities can thrive.

In the journey towards a mature DevSecOps culture, an organization must also nurture a safe environment. Team members should feel empowered to share both their triumphs and setbacks through after-action reports. These narratives of success and learning opportunities should, in turn, fuel the refinement of system design, the hardening of implementations, and the enhancement of incident response capabilities within the DevSecOps process.

The preference for small, incremental changes over infrequent, large-scale alterations becomes evident, as releasing small changes limits the scope and simplifies management, while concurrently fostering a tighter feedback loop with customers. It's essential to embrace the feedback loop and the insights gained from customers. By allowing users to steer change, the development team can more adeptly respond to emerging requirements, even those that might have been unforeseen.

Lastly, a fundamental principle known as “the boy scout rule”, often expressed as "leave the playground cleaner than you found it", is also imperative [27]. In the context of DevSecOps, this translates to continuous refactoring. Continuous refactoring involves the ongoing process of restructuring code to enhance readability and simplify complexity [28]. It ensures that the codebase remains sustainable, facilitating collaboration among developers. However, this practice requires careful planning and budgeting to systematically reduce the technical debt accumulated throughout the development process. In DevSecOps, the *organizational* pillar is thus the initial thread that sets the tone. Highlighting the need for a cultural metamorphosis, mirroring the collaborative spirit that defines the broader DevOps landscape. Together, these principles and practices form the fabric that integrates security seamlessly into the software development process.

Process pillar

At the heart of DevSecOps lies the *process* pillar, a foundational element that forms the bedrock for the entire DevSecOps transformation. Embarking on this journey requires a deliberate, step-by-step approach, starting with small, manageable tasks that lend themselves to automation. The progression occurs in iterative phases, allowing the organization to gradually enhance its DevSecOps capabilities while refining the process over time. Recognizing that each software lifecycle is unique, shaped by factors like the mission environment, system complexity, architectural choices, and risk tolerance, it's prudent to start small by automating a Continuous Build pipeline [10]. In this initial phase, automation revolves around the build process, triggered by a developer's code commit. As experience accumulates and confidence grows, the organization can advance towards implementing other Continuous practices, such as Continuous Integration, Continuous Deployment, and Continuous Delivery.

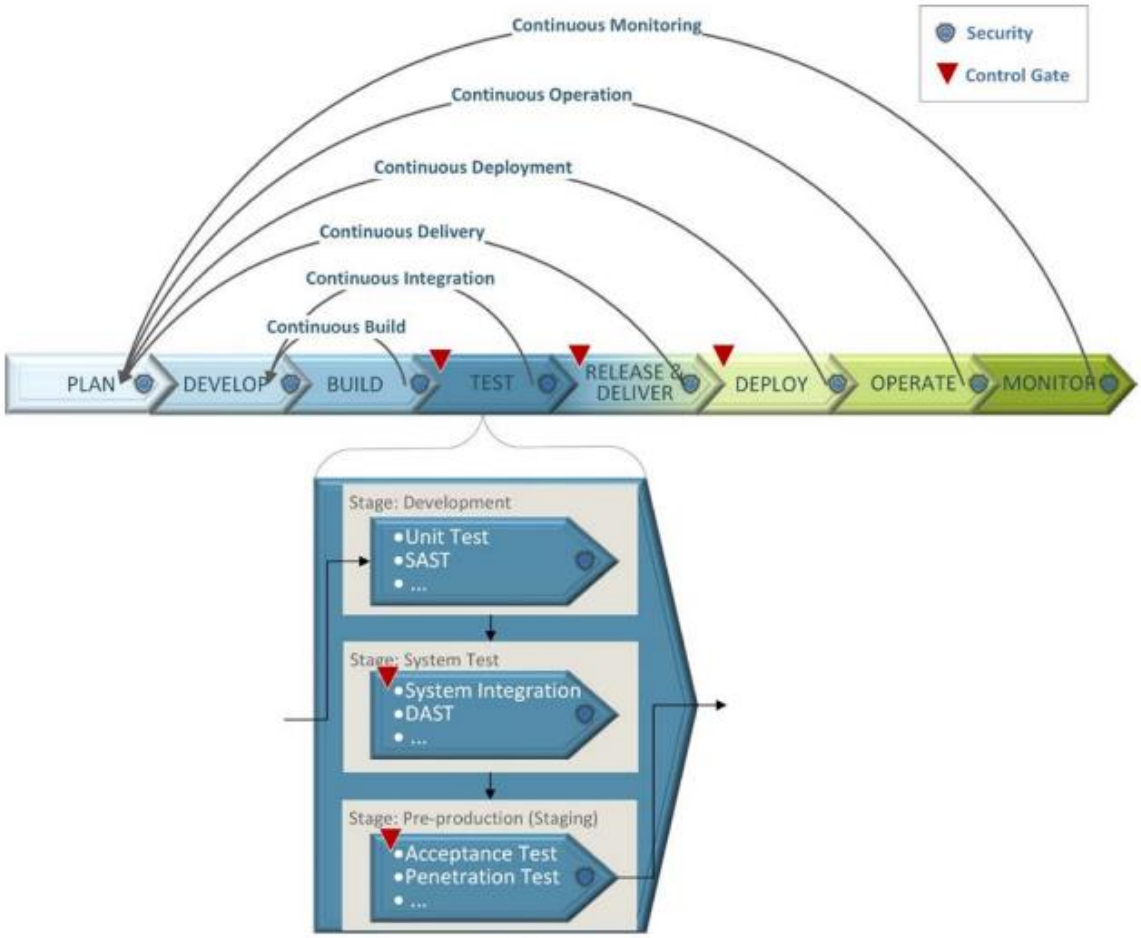

Figure 9 DevSecOps processes [10]

However, achieving a closed loop within DevSecOps, as depicted in Figure 9, necessitates the inclusion of Continuous Operation and Continuous Monitoring as integral components of the security and assurance framework that surrounds the CI/CD pipeline. These elements are paramount in establishing a streamlined and secure operational environment, a topic that will be explored later in this thesis.

It's important to note that there is no one-size-fits-all approach to management processes within DevSecOps. Each DevSecOps team operates with its unique set of requirements and constraints. Therefore, the process design should embody collaborative efforts, involving all multidisciplinary teams. Furthermore, automation takes center stage as a driving force for efficiency, with a variety of tools and technologies available to facilitate various phases of the software lifecycle, as discussed before. In such, DevSecOps, by nature, adopts an iterative and closed-loop approach, while, starting conservatively, it gradually progresses toward continuous improvement. Initially, more control gates may be established to allow for human intervention, but as DevSecOps teams gain confidence and maturity, the number of these control gates can be gradually reduced.

DevSecOps maturity model

Part of the continuous improvement within the DevSecOps process is the use of a maturity model. The American Department of Defense has developed a good to use DevSecOps maturity model, with the goal of helping organizations in improving their DevSecOps

processes and capabilities. This maturity model has been presented in the DevSecOps playbook of September 2021 [29]. By understanding the current maturity level, a DevSecOps team can take improvement actions if needed.

Technology pillar

Now that we have covered the core, and the heart of DevSecOps, the strength of DevSecOps is automation. Automation minimizes manual intervention while maximizing efficiency [10]. Creating an Integrated Development Environment (IDE), equipped with various DevSecOps plugins and tools, including static application security testing tools, allows for orchestration through configuration files, DevSecOps tool configuration scripts, and application runtime scripts. This approach, collectively referred to as Infrastructure as Code (IaC), streamlines processes and eliminates the need for manual setup.

In the realm of security, a similar approach, known as Security as Code (SaC), prevails. Security policies are encoded within configuration code, accompanied by the implementation of security compliance checks and auditing as code. Both IaC and SaC necessitate software traversing all phases of the software development process, including design, development, version control, peer review, static analysis, and rigorous testing. In essence, technology and tooling play a significant role in enhancing the efficiency of DevSecOps practices and shortening of the software development lifecycle. These tools enable automation of software production and orchestration of security processes and operations.

Governance pillar

In the expansive landscape of the DevSecOps frame, the Governance pillar serves as the guardian that oversees the entire software lifecycle. Throughout the software lifecycle risks can be associated within a project, and it's within this pillar that they are actively identified, assessed, and managed. However, governance in DevSecOps transcends the conventional notion of periodic evaluations and approvals by only authority figures. Governance activities do remain a constant presence throughout the entire DevSecOps lifecycle, for which everyone is responsible. Responsibilities should be pushed or delegated to the lowest level in the organization. In Information security, the Information security governance model, or the Direct-Control cycle of von Solms (*see figure 10* [30]) is often used to describe the three levels of Information Security governance as: Direct, Execute and control. These three governance levels are dispersed across the strategic-, tactical-, and operational levels of the organization. With the operational level to be the lowest level of governance.

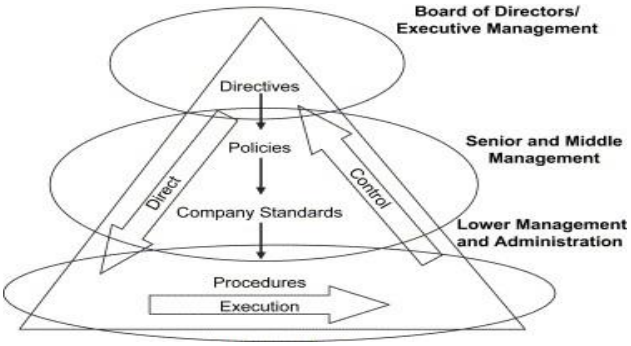

Figure 5 Information security governance framework [30]

As previously emphasized, this extended purview encompasses both Continuous Operations and Continuous Monitoring practices, critical components that ensure the robustness and security of the software pipeline. Continuous operation comes in action when a successful deployment towards production has been made. This phase places the burden on the operations team, which shoulders a range of responsibilities crucial for the system's stability and security. Among their core duties, the operations team takes charge of system patching, ensuring that the software is up to date and fortified against vulnerabilities. Compliance scanning is another vital task, ensuring that the system aligns with regulatory requirements and industry standards. Data backups are performed meticulously to safeguard against potential data loss, with a robust restoration plan in place should the need arise.

Automation of Governance

One of the remarkable features of DevSecOps is its capacity for automation, and governance is no exception. Just like other important aspects of DevSecOps, governance activities can also be seamlessly integrated into automated workflows. This automation not only expedites processes but also enhances their reliability and consistency. In essence, the *governance* pillar in DevSecOps is a dynamic and ever-present force that actively manages risks, maintains a watchful eye on operations, and can be harnessed to enhance efficiency through automation. It embodies the essence of the DevSecOps philosophy, which seeks to embed security and governance into every facet of the software development lifecycle.

Three lines of defense (3LoD) model

Within the framework of DevSecOps, the delegation of responsibilities to the lowest governance level is a fundamental need as beforementioned. However, while this approach empowers teams to make autonomous decisions, it's crucial to recognize that this level cannot be held solely accountable for all day-to-day business-related risks and choices. This is where the Three Lines of Defense (3LoD) model comes into play, providing a structured approach to risk management and control [31].

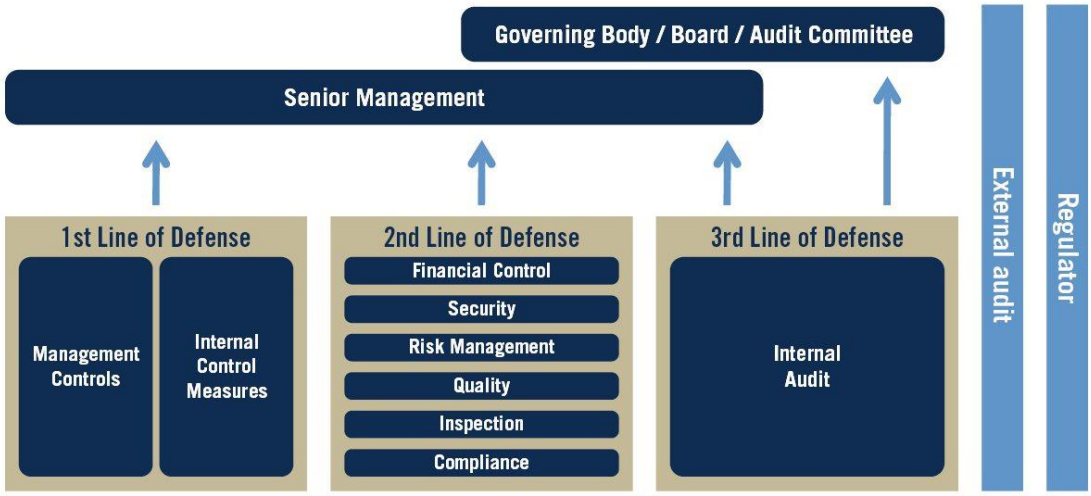

Figure 6 Three Lines of Defense (3LoD) model [31]

The first line of defense in the 3LoD model encompasses the business itself, including senior management. This line assumes accountabilities for all the risks faced by the organization. Embracing a dual management structure, which combines top-down and bottom-up perspectives, allows the business to engage in a continuous risk dialogue. This dialogue

extends to operational levels, facilitating the sharing of insights about incidents that impact DevSecOps processes. By fostering an understanding of the organization's dynamics and taking proactive measures to mitigate risks, this approach is widely regarded as a robust risk management strategy. In the second line of defense, a team of specialist's steps in to provide support to the business. These specialists cover diverse domains such as security, financial control, risk management, compliance, and more. Often, these specialized roles are necessitated by the evolving landscape of laws and regulations. This layer also houses critical components like the Information Security Management System (ISMS) and other process-related systems essential for effective risk management and control.

The third and final line of defense is occupied by internal audit functions. Internal audit teams offer management invaluable assurances regarding the quality of risk management processes and control mechanisms. In essence, they act as the capstone of the Deming cycle (Plan, Do, Check, Act – PDCA) [31]. While the third line doesn't bear direct responsibility or accountability for inconsistencies in business processes, its role is to thoroughly analyze, validate, and report on the design, operation, and existence of implemented processes, controls, and other measures.

These three lines of defense align therefore closely with an integral Governance, Risk, and Compliance (GRC) approach. Compliance is intrinsically tied to Continuous Monitoring, covering the entire CI/CD pipeline, and plays a significant role in maintaining the integrity- and security of DevSecOps practices.

Continuous Monitoring

Continuous Monitoring, an extension of the earlier mentioned CI/CD phases, is a vital link in the DevSecOps chain, turning the pipeline into a complete closed loop. Stretching from the secure production environment all the way back to the planning phase, this step entails several critical monitoring activities. First and foremost, it involves the continuous creation and update of an asset inventory, compiling essential information about all system components. Key Performance Indicators (KPIs) are employed to gauge the performance and security of these components, providing valuable insights. However, since KPIs primarily offer retrospective insights, Key Risk Indicators (KRIs) step in to take a forward-looking stance, helping organizations anticipate and address potential threats, and risks [32].

Moreover, monitoring activities also encompass continuous tracking of application logs and system events. These logs are often aggregated and forwarded to designated platforms, such as Security Information and Event Management (SIEM) clusters, for streamlined log management. In non-containerized environments, aggregating and storing logs may necessitate more manual labor, along with the integration of security tools for comprehensive coverage. Analyzing the data stored in log databases can be time-consuming but is essential for maintaining security. Containerization and virtualization platforms bring further benefits to Continuous Monitoring, by simplifying compliance tests. Furthermore, tests against mandatory hardening baselines, like the STIG or CIS benchmarks, can be conducted to bolster system security. These baselines provide blueprints for applying security hardening measures, reducing the attack surface by, for instance, closing unused services and ports.

Additionally, Continuous Monitoring also encompasses software license compliance checks, a critical aspect to prevent legal issues and breaches of contract arising from unlawful software usage. For codebases utilizing open-source software, automated software composition analysis (SCA) tools come into play. They delve into security, license compliance, and code quality, ensuring that open-source components are used within the confines of licensing limitations and obligations. In essence, Continuous Operation and Continuous Monitoring are the twin pillars that help sustain a secure and agile DevSecOps pipeline, continuously fortifying the production environment and maintaining vigilance for potential risks and incidents.

Security test methods

Another significant aspect of Continuous Monitoring is the performance of extensive automation tests providing a level of assurance that major security vulnerabilities and exploits are not present in the code prior to release. Employing security testing methods, and penetration testing frameworks this offers proactive enhancements of security for the entire DevSecOps process [33]. To aid organizations in performing security tests, a variety of free (open source) and commercially available penetration testing frameworks and other security testing methods are available, including: "Whitesource," "Aqua Security," "CheckMarx," "Veracode," and many more [34]. These tools and frameworks encompass various comprehensive application security testing (AST) capabilities, such as "Static Application Security Testing" (SAST), "Dynamic- and Interactive Analysis Testing" (DAST/IAST), and "Software Composition Analysis" (SCA). DAST, for instance, adopts the perspective of an attacker to detect vulnerabilities and security gaps through threat modeling exercises. SAST examines software code to identify security flaws without the need for code execution, while IAST combines elements of both DAST and SAST. In IAST, the goal is to monitor application performance during tests. Additionally, the "Runtime Application Self-Protection" (RASP) method uses real-time data to detect and address attacks on the application, enabling automatic intervention when an attack occurs.

Secure coding is not a static aspect of software development; it can be application-specific and requires interactive documentation, including best practices, coding standards, and guidelines. Examples include the NIST Secure Software Development Framework (SSDF) and the OWASP Top Ten. The OWASP Top Ten addresses various web application security risks, such as input validation to mitigate SQL injection, - or Cross-Site Scripting attacks [35]. Other areas of focus within OWASP include authentication and authorization controls, session, and file management, and more. The secure coding context must be well understood by software development teams, who should diligently adhere to secure coding guidelines, standards, and practices established by the organization.

Shifting left vs Shifting right

In addition to the beforementioned testing methods, the so-called “shift left”, and “shift right” concepts, are part of the core testing concepts of the DevOps/DevSecOps approach [36]. These two fundamental concepts, play a remarkable role in shaping the core testing methodologies. These concepts are integral to ensuring the effectiveness of the development and security processes [36].

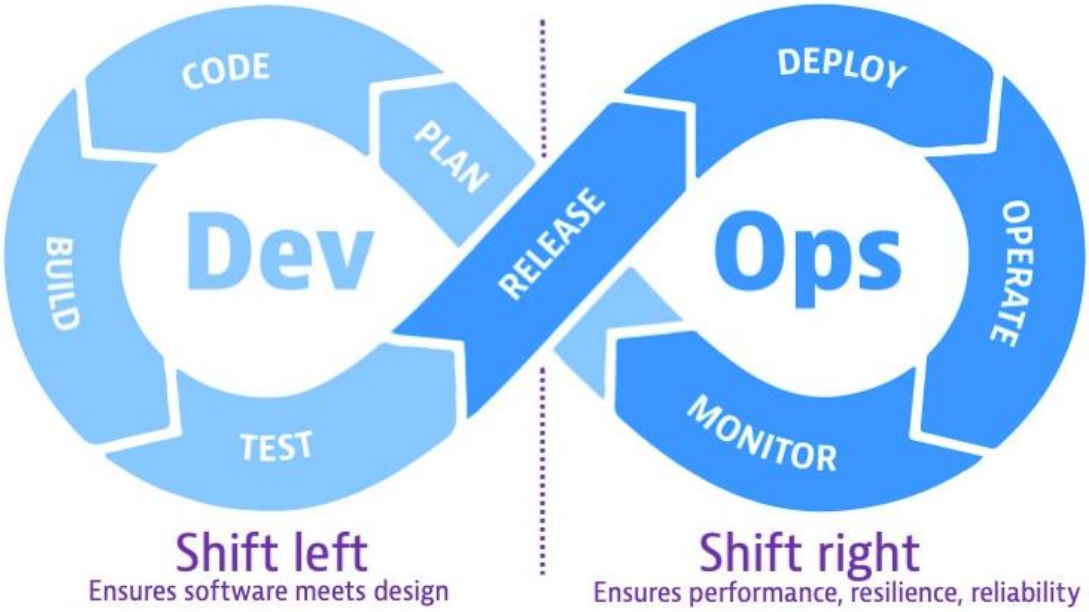

Figure 7 Shift left vs Shift right DevSecOps [36]

"Shift left" involves the strategic relocation of critical activities within the DevSecOps framework to the early stages of the software development process, often even before a single line of code is written. These activities encompass testing, quality assessment, and performance evaluation. By shifting left, organizations aim to proactively address potential issues and vulnerabilities right from the outset, ensuring that the software aligns with the expected design criteria. This approach emphasizes the importance of a security-by-design mindset, as introduced earlier.

Conversely, "shift right" serves as the counterpart to shift left. In shift right, the primary focus shifts towards critical criteria like "software reliability," "performance," and "resilience " [36]. The objective here is to rigorously assess whether the newly developed software can operate effectively in a production or similar environment that simulates real-life user loads. This transition is crucial because development environments often lack the necessary user data or conditions to replicate actual usage scenarios. The shift-right approach entails identifying and addressing any potential issues or performance bottlenecks before releasing the software into the production (Operations) environment.

Both shift left and shift right come with their own set of testing methods tailored to their respective needs. Shift left encompasses various testing methodologies, including: "Traditional shift left testing," "Incremental shift left testing," "Agile/DevOps testing," and "Model-based shift left" [36]. On the other hand, shift right employs diverse testing methods designed to suit its objectives. These include: "A/B testing," "Synthetic monitoring," "chaos engineering," "Canary releases," and "Blue-green deployment" [36]. While there are distinctions between these shift left and shift right testing methods, the thesis emphasizes the overarching concept of "shifting both ways". Therefore, we won’t focus onto the specific details for these testing methods. This approach entails conducting both shift left and shift right tests, together with security testing methods outlined before. This comprehensive approach significantly enhances application security. Shifting left helps in identifying vulnerabilities at an early stage, simplifying the remediation process while it's still relatively straightforward. From a shifting right perspective, testing can uncover hidden components

within, for example, a container or repository, even if these components are obscure. A relevant real-world example is the discovery of the 2021 zero-day vulnerability "Log4shell/Log4j", a specific component within applications and dependencies. It took several weeks, with the collaboration of world-wide cybersecurity communities, to identify in which applications the Log4shell/Log4j Java library was embedded, and therefore which application were potentially vulnerable. Shift left and shift right testing methods can play a crucial role in detecting these vulnerable components in an early state.

In summary, shifting left focuses on reducing software defects by addressing issues early in the development process. Meanwhile, shifting right ensures that the software can operate reliably in a production environment. The best practice advocated in this thesis is to embrace both shift left and shift right concepts. This approach, combined with other mentioned tests, contributes to the advancement of the DevSecOps organization toward successful Continuous Integration and Continuous Delivery (CI/CD), ultimately resulting in a more agile and efficient organization.

Summary of DevSecOps

DevSecOps emerges as a transformative approach, intricately weaving security into the very fabric of DevOps. However, security often lingers as an afterthought, addressed reactively when vulnerabilities surface. This disjointedness between DevOps and security introduced the need for DevSecOps as a concept. DevSecOps can be understood through four foundational pillars. The Organizational pillar requires a transformative shift, that begins with senior management's commitment to embrace a holistic view and shared responsibilities among development, security, and operations teams. It demands breaking down silos, fostering communication, and nurturing a safe environment where learning from successes and setbacks is encouraged. Next comes the Process Pillar, the core of DevSecOps. It entails a step-by-step approach, starting with small, manageable tasks that are ready for automation. Continuous practices like Integration, Deployment, and Delivery become key components of this pillar, and it evolves iteratively over time.

Technology, the third pillar, plays a significant role in automation and collaboration. Tools and integrated development environments (IDEs) equipped with DevSecOps plugins to facilitate orchestration, streamlining processes, and eliminating manual setup. This approach, known as Infrastructure as Code (IaC), simplifies workflows and enhances efficiency. Lastly, Governance, the fourth pillar, acts as the guardian overseeing the entire software lifecycle. It identifies, assesses, and manages risks throughout the process. Continuous Monitoring and Continuous Operation are critical components, ensuring a robust and secure operational environment. However, the path to implementing DevSecOps is not without challenges: DevSecOps culture necessitates breaking down traditional silos, often met with resistance to change. Convincing teams to adopt the "shift left" mentality of addressing security early in the development process can be a significant challenge. The same applies to skills and training, as embracing DevSecOps often requires employees to undergo training and upskilling, with skill shortages posing a roadblock to successful integration. Furthermore, as people must work seamlessly with various tooling, this seamlessly integrating of tools for automation and security testing can be a complex technical challenge. Especially, implementing and managing effective security testing methods across the development pipeline requires expertise and attention to detail.

Moreover, meeting regulatory requirements and compliance standards, particularly in regulated industries such as government, healthcare, finance etc., is a constant challenge. DevSecOps must align with these standards without compromising speed and agility. However, another burden lies often within legacy systems. Retrofitting security into older systems can be complex, often hindering DevSecOps adoption.

In conclusion, DevSecOps seamlessly integrates security into software development, fostering a culture of collaboration, efficiency, and continuous improvement while minimizing risks and vulnerabilities. It supports the notion that security is not a hindrance but an enabler, ensuring faster and more reliable software delivery. However, overcoming cultural, technical, and regulatory challenges is essential for successful DevSecOps implementation.

4. Results Of literature research

While the DevOps approach has been around for more than a decade, it still does have a huge potential of streamlining current software development processes and to increase team collaboration between the operations-, and development teams, and other stakeholders such as customers/end-users. However due to the many (cyber)security threats, vulnerabilities and risks involved within modern software development, security can no longer be ignored and must be an integral part of software development. Security must be the concern and proper mindset of everyone involved, from the business to the developers and IT operators. Although most software development organizations do understand the usefulness and necessity of applying security, security is often still treated as an afterthought, or organizations do not know how to prioritize security during software development and other phases of the CI/CD pipeline. Also, security is often seen as a restriction towards progress, and requires continuous commitment and investments of the organization. Because a clear return on investment (ROI) is often not directly visible, this might hinder buy-in from management and members of the DevOps/DevSecOps teams.

During research there have been several common challenges identified, in important areas, such as people, tools, values, and governance/processes. These challenges do impact either the adoption of security within DevOps, or the security practice all together.

4.1 People

While the human factor is the main success factor towards a successful software development life cycle, it is also one of the biggest challenges of the DevOps/DevSecOps approach. Since DevOps builds on the Agile culture/way of working, the addition of security does however require a change of mind, and therefore a cultural change is necessary. Security must change the rituals, habits, and how the team thinks about how secure the software is in all stages of the software development lifecycle. This also includes the functionalities, and availability of, for example, a provided application and the platform underneath.

Both DevOps, and DevSecOps have the same values of preserving and embracing the Agile culture, with the goal of making better, and faster software deliveries. While in addition DevSecOps does incorporate the security by design mindset, within the core of software development.

Team skills & (security) awareness

In DevSecOps, a new skillset for all team members is required. While security is an extremely broad concept, it is not mandatory for all team members to have a deep understanding of security, but a minimum understanding of security at a certain level is a must. To refer to a skilled team member, this means that at the horizontal- and vertical level of the skillset, this team member must understand every DevOps stage, including security, while at the same time this team member must have a very deep understanding of their own area of expertise (*e.g., development, or testing*). As security is everyone's responsibility, this implies that everyone (*not only security specialists*) must be capable of seeing and understanding the threats and security weaknesses, however that can also lead to risks for the application.

As collaboration- and communication is key in a DevSecOps team, no isolation silo should be created within the team. To avoid this, a change in team composition can be necessary, such as ensuring that a security specialist is a full-time member of the team. However, having a dedicated security specialist, does not release the organization from its security obligations. Security remains an integral discipline, which must be integrated into the structure of the organization. The organization must therefore have good policies and procedures, so that even if there is no security specialist present, the DevSecOps team take the right steps and ask themselves the right questions. If in doubt, a security specialist can always be asked to give advice.

Continuous learning
As DevOps/DevSecOps is a highly dynamic, flexible/Agile way of working, this requires a continuous investment in learning opportunities for all team members. As security is a generic concept, an example of a great investment, is offering training opportunities within secure coding best practices. It is a respected rule that security by design is much cheaper, and more effective, than trying to implement security measures after the software has already been written and deployed [37].

Ownership & Accountability
Accountability and ownership in DevOps/DevSecOps approaches do go hand in hand. For example, in the beforementioned interactions of increments (*microservices/building blocks*), the entire team takes on the ownership of the service, which gets worked on via these increments. The idea behind this, is to make the team responsible from end-to-end, this does include not only the building, and delivery/deployment of the software, but also the operations aspects, such as maintaining functionalities as well. In DevSecOps, this also includes the integral security, within all phases of the software development lifecycle, for which everyone is responsible. In a matter of speaking, this makes every team member, also accountable for their own individual actions performed. If an issue has been detected, during any phase of DevSecOps, the entire team is thus responsible of resolving this issue together.

4.2 Tools

In a Dev (Sec)Ops pipeline, automation- and the usage of different tools is paramount to suit the goal of ensuring better, and faster, but also more secure software. DevSecOps tools do help in creating consistency throughout the process. Also, these tools can perform a variety of tasks, and this allows the team to be highly flexible, with the aid of automation for repeated tasks. Repeated tasks can be best executed via tools, as over time a human being cannot guarantee the same accuracy and consistency. However, these tools at first can be overwhelming, and do require many investments such as expenditures of purchase, resource costs, and the training time of team members. A wrongful selection of these tools, however, could lead to counterproductivity. A tool should therefore only be used when it can accommodate the organizational needs. For example, a small development organization, would not improve, if an automation tool is too complex for anyone to handle. The harder it is to configure a tool, how easier it is to introduce security flaws and risks to not only the organization, but its customers as well since the tool will be misused because of lack of knowledge. Organizations must understand that DevSecOps is not just the installation of automation tools. Organizations must give themselves the time to adjust, and overtime gain

experience, and proper knowledge to work with these tools to successfully implement automation. Therefore, the best practice is to start small, with for example automating only the build process within DevSecOps, and easy to automate tasks. Additional, for each tool, there must be proper access controls, and audit logs in place. When it is unclear who has access to these systems, it might be the case that misconfigurations can be made, which negatively affect the security of the entire system. In result this might open possibilities for outsiders, and insiders to perform unwanted activities, or even attacks. To gain insights in access controls, auditing purposes are required to implement detection capabilities to detect if any unwanted behavior takes place within these systems. Furthermore, improper separations of duties, or roles, can affect the production environment in a negative way. A developer should not be allowed to deploy software to production, and an IT operator, should not be able to make adjustment to a software build. With other words, while collaboration, and communication is key within DevOps/DevSecOps approaches, there must be separated environments for development, testing, and operational needs.

The lack of proper automated testing capabilities
As aforesaid, in the lack of proper access restrictions, it is mandatory to have separate environments, that offer (automated) testing capabilities. Automated testing is the preferred solution, above manual testing. As manual testing is often more susceptible of making mistakes, due to human error. The same applies to obligatory security tests, these tests must also be automated as much as possible. Via automation, the organization can ensure that a security baseline will be met, during each iteration of DevSecOps.

Cloud vs on-premises solutions
Working in, or via cloud solutions, like the beforementioned Infrastructure as Code (IaC), does require another security approach than on-premises solutions, as this does introduce new security risks for the organization. Examples include exposed systems on the internet, and the usage of a malicious code repository. As cloud solutions, like containers might not receive the same security controls as on-premises/local servers, this could result in a variety of damages such as the unwanted leakage of data, and unauthorized access to components of the cloud environment. Or worse, the cloud environment can be misused as a steppingstone to access other networked assets. It can be extremely dangerous when untrusted inputs, from untrusted sources are being used within the software development processes. Therefore, in secure standards it must be mandatory to determine and describe what sources are safe to use, and what not. Also, the security testing capabilities, must be up to date, to detect any undesirable security weaknesses.

Legacy systems, and technical debts
Many organizations, if not all organizations do have existing (legacy) systems that were once built, without a proper security mindset. Integrating security within these legacy systems or rebuilding these systems can be a lengthy time-consuming process. It can also stress, many resources of the organization.

4.3 Values

Continuous monitoring is as we know part of the entire DevOps/DevSecOps approach and overlooks all CI/CD phases. Continuous monitoring should be for example performed via measurements such as Key Performance indicators (KPIs). Via KPIs, the entire DevSecOps

approach can be measured, such as understanding how much software failures happened across the total number of increments. By knowing how the efficiency of the DevSecOps team is, the team can make necessary adjustment that bolster any weaknesses in the daily processes. A KPI is therefore a backwards looking indicator. However, since security is an integral part of DevSecOps, it is not enough to only look backwards into process metrics. DevSecOps also require the embracing of key risk indicators (KRIs). A KRI is a forward-looking, metric that can make potential risks, with negative impact visible for the organization [32]. Both KPIs and KRIs should be put in effect, at the start of the planning for new incremental pieces of software, and not just at the end of the entire software development lifecycle. In the upcoming NIS2 directive, for example, the focus will change towards resilience, this also includes the business continuity, and recovery capabilities of for example an application. Making measurements periodically, can help by achieving this.

Velocity over complexity and security

When the organization strain the DevSecOps processes too much, by making more deployments, than it could manage, this impacts the overall security in quite a negative way. While a faster delivery of software is one of the core ideologies of DevOps, organizations must however not forget to initiate proper security tests. By only focusing on velocity of the software development process, this could lead to a negligence of performing proper testing activities. When testing activities are not performed, or in a less consistent manner, this could lead to security flaws and gaps. Therefore, the organization should create a delicate balance between frequent delivery and security testing. With other words this includes the risk management process of the organization. As the organization must decide for themselves what risks are acceptable, and what not. Also investing in (new) security measures, should be considered carefully.

Continuous improvement

Another challenge is the lack of a long-term vision that does not cope with changing circumstances that require continuous adaptations, such as new technologies and/or legal/regulatory alterations. Also, customer demands can change over time, therefore embracing continuous improvement, preferably with each iteration of the CI/CD pipeline/Software development lifecycle, results in preventing major failures.

Since iterations can go back- and forth, making proper lessons learned contribute to the security, and other important aspects. If these lessons learned are ignored, it can however lead to a negative impact for the entire software development lifecycle.

DevSecOps control

Insourcing of security within the DevSecOps team, has huge benefits over seeing security as only a separate department. Seeing security as a separate aspect, is a blind spot that gives a DevOps team less control over their activities, and in return creates organizational dependencies, which have a negative impact. An example of a negative impact is reducing the flexibility of the entire team. When security is part of the team, from end-to-end this, removes these dependencies and do help increase the flexibility and velocity of the entire software development lifecycle.

4.4 Governance & Processes

Security must receive proper attention from everybody, including management. When the management does not feel itself responsible for security, or security it not a priority, this can also seriously affect the overall security of DevSecOps. Management buy-in is therefore crucial, to ensure the security of the organization, and its customers. The more buy-in there is, the less resistance the organization most likely faces. Also buy-in, does contribute to a cultural shift, and create the right environment for the DevSecOps team to thrive in. As it gives a common goal for all disciplines within the DevSecOps team.

Compliance and regulations

Several sectors such as government, are subjected to strict regulatory requirements in accordance with security- and data protection laws, policies etc. Ensuring that the DevSecOps practices are aligned with such regulatory/compliance requirements can be a continuous demand for the organization.

5. Results of Interviews

For the interviews, the researcher has applied the semi-structured interview methodology, with the aid of a set of questions (See Appendix 1). The interview questions have been created based on the results from the literature analysis, in which distinct challenges of adopting DevSecOps have been found around important areas. These interview questions have been created with the goal of cross-examining the theory, with results in practice. The questions have been thoughtfully structured via the grounded theory, to create a logical flow for an exploration in the DevSecOps practices, covering a conceptual understanding, practical experiences, challenges, but also a potential strategy on how to tackle these challenges.

In the interviews, we start by gauging the interviewee's familiarity and experience with DevOps-related concepts. This initial question serves as a foundational inquiry to understand their level of expertise. Once we establish their experience, we move on to a more conceptual exploration. We ask them to define DevOps and DevSecOps in their own words. This step helps us clarify their understanding of these fundamental terms. With the definitions in place, we delve into their knowledge of DevSecOps principles. This is a critical area as it allows us to assess which principles, they are aware of and how they might apply them in practice. For those who have practical experience, we explore the real-world implementation of DevSecOps or security within a DevOps framework. This gives us insights into their hands-on experience and the strategies they've employed.

Understanding that practicality often comes with challenges, we inquire about the difficulties they've faced during implementation. More importantly, we want to know how they've addressed these challenges, revealing their problem-solving abilities. Furthermore, as collaboration and communication are central to DevSecOps, we emphasize their importance and explore if the interviewees also agree on this. We also ask about strategies for promoting collaboration and communication within a DevSecOps culture.

Shifting our focus to the development workflow, we explore how security is prioritized. This sheds light on whether security is integrated at all stages of development or if it's seen as an afterthought. Next, we check their familiarity with the shift left and shift right concepts and inquire about their preferred shifting approach within DevSecOps.

Moving into the realm of tools, we ask about their knowledge of common security tools used in DevSecOps. We also want to know if they have personal experience with these tools and their perceptions of their benefits and drawbacks. Additionally, we inquire about specific security testing methods. Considering the crucial role of monitoring and auditing, we seek to understand their views on the importance of logging within DevSecOps. Furthermore, regulatory and compliance aspects are paramount in many organizations. We, therefore, inquire about how interviewees ensure compliance within their DevSecOps environments.

Given that Continuous Integration and Continuous Delivery (CI/CD) are foundational in DevOps, we check if the interviewee is familiar with these concepts. We also ask about their understanding of how security fits into the CI/CD pipeline. Besides that, since the organization is currently making a transition from on-premises to the cloud, we explore any

unique requirements or challenges specific to cloud-based DevSecOps. Additionally, we touch on the concept of Infrastructure as Code (IaC) in this context.

Recognizing the role of automation, we seek to understand the balance between automation and manual tasks within DevSecOps. We also invite interviewees to share additional aspects or components they believe contribute to improved security in DevSecOps and CI/CD pipelines, drawing from their experiences. To wrap up, we ask if interviewees perceive any challenges not previously addressed during the conversation. This open-ended question allows them to share any valuable insights we may have missed.

Selection of Research Participants

Since DevSecOps is the abbreviation of Development, Security, and Operations, we have interviewed individuals from these three different work areas that possess relevant knowledge and experience in the field of DevOps/DevSecOps.

We have chosen the participants based on the following criteria:

- Individuals must be involved in DevSecOps practices within our organization or have experience with DevOps/DevSecOps.
- Varied roles and responsibilities related to DevSecOps adoption/implementation.

As such in total eight interviews have been held, resulting in two or three colleagues per work area participating in the interviews with the aim of achieving a balance of knowledge between the three teams.

Ethical Considerations

Informed consent has been obtained from all participants, also consent has been provided to create audio recordings of all interviews. Furthermore, pseudonyms have been used in the transcripts, to ensure the anonymity of the participants.

Interview transcripts

For each interview, a transcript has been made. To structure the transcripts, the grounded theory has been applied to allow the interviews to be analyzed thematically with the purpose of determining themes and similarities within the interview data. This has been achieved using an iterative process of coding and categorization to help derive meaningful insights and to identify recurring challenges across all eight participants. During the open coding phase of analysis, several challenges have been identified, related to the adoption of DevSecOps across the eight interviews. Below are the open-coded challenges along with brief descriptions, and several interview fragments. The open coding elements are included in Appendix 2 Grounded theory.

5.1 Open coding

Below are the open coding elaborations of applying grounded theory to all interview transcripts.

Challenge	Interviews (Numerical)	Descriptions

Lack of DevOps/DevSecOps Culture	1, 2, 3, 5, 7	Establishing a culture that fosters collaboration and shared responsibility between development, security, and operations teams.
Security Integration	1, 2, 3, 5, 7	Seamlessly integrating security practices into the DevOps pipeline from the design phase onwards.
Mindset and Culture Change	1, 2, 4, 5, 7	Shifting the mindset and culture of teams to align with DevSecOps principles and redefine roles and responsibilities.
Communication and Collaboration	1, 2, 3, 5, 6, 7, 8	Ensuring effective communication and collaboration among the development, operations, and security teams.
Continuous Testing and Monitoring	1, 3, 4, 5, 6, 7, 8	Ensuring comprehensive testing and monitoring throughout the DevSecOps lifecycle, including automation.
Tool Selection and Integration	1, 2, 3, 4, 5, 6, 7, 8	Selecting and integrating appropriate tools for various DevSecOps processes, including automation and security testing.
Security Awareness and Training	1, 3, 5, 6, 7, 8	Raising security awareness and providing training to ensure that team members understand security risks and best practices.
Adoption of Infrastructure as Code (IaC)	1, 3, 5, 6, 7, 8	Shifting to Infrastructure as Code (IaC) and the associated mindset change and tooling required, especially from on-premises environments to cloud-environments.
Automation of Manual Tasks	1, 3, 5, 6, 7, 8	Automating manual tasks such as deployments and security checks to enhance speed and reliability in DevSecOps.

Compliance and Regulatory Requirements	1, 3, 5, 7, 8	Meeting compliance and regulatory requirements in DevSecOps, including the need for policies, (quality)control gates, and reporting mechanisms.

5.2 Axial Coding

Now that all interviews have been open coded, we can perform the axial coding phase, in which we organize the open-coded challenges into broader categories based on their common themes and relationships. These categories provide a more structured view of the challenges associated with DevSecOps adoption, emphasizing the importance of cultural transformation, effective collaboration and communications, security integration, and automation and tooling. This grounded theory approach helps to identify the core challenges that organizations face when implementing DevSecOps, offering insights into the interconnectedness of these challenges and their impact on the overall DevSecOps approach. The challenges mentioned in the before discussed Open Coding section, have been categorized via the verbatim quotes of all interviewees in the following categories:

Cultural Challenges: *Lack of DevOps/DevSecOps Culture, Mindset and Culture Change.*
Collaboration and Communication: *Communication and Collaboration.*
Security Integration: *Security Integration, Continuous Testing and Monitoring, Security Awareness and Training, Compliance and Regulatory Requirements.*
And finally, **Automation and Tooling:** *Automation of Manual Tasks, Tool Selection and Integration, and Adoption of Infrastructure as Code (IaC).*

5.2.1 Cultural challenges

In the ever-evolving realm of DevSecOps, where the fusion of development, security, and operations converges, a profound cultural transformation is required to navigate its complex landscape. It's not just about adopting new tools or processes; it's a shift in mindset and a transformation that extends from individuals to entire teams. As aptly stated in Interview 1 “DevOps is a culture, an integration between development and operations." This culture shift goes beyond acquiring new skills; it necessitates a fundamental change in the way people think and approach their work. This integration between development and operations teams sounds simple, in practice it is seen as a challenge as interviewee 3 states: “This sounds simple, but the special aspect about this is, that both dev and ops are two separate worlds and one of the main challenges is to be letting them grow towards each other." When asked about security and cultural challenges, interviewee 3 also mentioned that: “security must be embedded within different functions. However, now is security often spread out across these functions. However, also spread out across security functions, and this is a weakness." Resulting in the need for an integral security mindset that is intrinsically available by all members of the team. This last statement gets reinforced by interview 1: “[…] it must be an intrinsic aspect of not only the individual, but also the entire team."

Continuing with the shifting left and shifting right approaches, interview 4 introduces the concepts of "shift left" and "shift right," highlighting the importance of a comprehensive security approach: "Shift left, we just talked about that. Security by design and shift right is

that you ensure that before you release something to production, for example, that you have actually tested in production that an application or a component, you name it, can continue to run." Emphasizing the need for security to be an integral part of the design process, while "shift right" ensures rigorous testing in the production environment before releasing any application or component. This holistic security approach demands a significant cultural shift within organizations. Interviewee 8 strengthen this notion by adding security, embracing DevSecOps ”DevSecOps, what it says. Developing software, with security and administration. But with security. From end-to-end." Grasping back to the shift left (security by design) practice a necessity within DevSecOps.

Furthermore, interview 6 adds another layer of complexity to the cultural challenges within DevSecOps—privacy. The interviewee stresses that "the most complex challenge is around privacy." This accentuates the nuanced considerations and complexity introduced by privacy issues into the DevSecOps culture, further emphasizing the need for cultural adaptability. The transformation of mindsets and organizational culture is, perhaps, the most significant hurdle on the journey to embracing DevSecOps principles. As interview 1 puts it, " The mindset, the culture... They must know for what they are responsible […]”. It's not merely a superficial adjustment but a profound shift that alters how individuals and teams perceive their roles and responsibilities. This magnitude of the cultural challenge gets reinforced by interview 7: “Culture is really the biggest challenge for me, a hurdle to take. A developer must really feel responsible for production, and administrators must delve into the designs and the future of how something is put together and how do you build something." Embracing that DevSecOps goes beyond the surface; it demands a metamorphosis in how people approach their work. This cultural shift signifies a change in the very DNA of organizations, a departure from traditional, siloed approaches toward a culture that is collaborative, security-conscious, and extremely aware of the implications of their actions. In the intricate landscape of DevSecOps, cultural challenges are the bedrock upon which the transformation rests. It's a journey that requires not just the adoption of new practices, but a fundamental change in the heart and soul of organizations, ultimately leading to a more secure and collaborative future in the world of software development and operations.

5.2.2 Collaboration and Communication: Bridging the Gaps

Continuing with the second category, DevSecOps is a journey that hinges on collaboration and communication. As Interview 1 stresses, "Communication is important; you must involve people into this." The significant challenge here is bridging the gap between development, security and operations, a sentiment echoed also by Interview 7: "You can make a nice figure eight, but if people stay in their own role […]” This holistic approach ensures that all voices are heard, and all perspectives are considered in the pursuit of secure and efficient software delivery. Furthermore, as interviewee 8 mentioned, organizations should: “be focused on rewards. Not only verbal rewards, and don’t have a blame culture. Allow mistakes to happen and to learn from each other. You must have a safe haven for this." While this also relates back to nurturing a safe culture, its notions the importance of having a buddy system, in which DevSecOps specialists can work together, and learn from each other. Another notion is that team members should feel empowered to share their triumphs and setbacks, as beforementioned.

5.2.3 Security Integration: Security by Design

From the literature results, we know that DevSecOps isn't merely about bolting on security measures; it's about the very essence of security into the very fabric of development. Interview 1 indicates this need for early security integration: "Security aspect, from the design phase (security by design)." Interview 5 reinforces this by emphasizing the importance of minimizing security divergencies in later phases: "The less security divergencies you have within follow-up phases...". Basically, the interviewees recognizes that the earlier security is integrated, the more effective and efficient the process becomes. At the heart of DevSecOps is its people, and raising their security awareness is of paramount importance to reduce such security divergencies. Interview 1 highlights the significance of continuous awareness efforts, emphasizing that they significantly contribute to the security of the process: “Awareness... If you can stick with continuous awareness aspects, this will tremendously contribute to the security of the process." However, interview 3 adds the following notion: "DevSecOps needs people who want to learn different things every day, and not just people who want to do their trick." Suggesting that for security awareness to succeed, DevSecOps requires people that want to learn new things, instead of just following the workflow. The consensus among interviewees is that DevSecOps is a culture of perpetual growth and knowledge acquisition.

Moreover, navigating the intricate maze of compliance and regulatory requirements is another challenge in the DevSecOps landscape. Interview 8 states that compliance: “begins with setting the right frameworks, metrics, processes, etc. In every phase of the lifecycle, and not solely afterwards. Evidence is important for compliance requirements”. This statement highlights the need for organizational enforcement in meeting compliance requirements, as interview 1 is stating, "The organization should enforce within the DevOps/DevSecOps process that compliancy requirements get met." This requires aligning processes and reporting mechanisms with the rigorous standards demanded by regulators. Interview 7 reinforces this by stressing the importance of having evidence at every phase, not just afterward, to meet compliance requirements. Interviewee 7 also mentions the fact that you can save time by having the required evidence ready during audits: "Because if you have all your file formation ready during audits, you can save time." This highlights the need for a proactive approach to compliance, ingrained in the very fabric of DevSecOps processes. In the multifaceted world of DevSecOps, these principles of early security integration, automation, continuous learning, and proactive compliance are the cornerstones of success. They represent not just a set of practices but a cultural shift towards a more secure, efficient, and adaptable approach to software development and operations.

5.2.4 Automation and Tooling: Streamlining the Journey

Lastly, automation and tooling emerge as critical DevSecOps components that streamline the journey towards more secure and efficient development and operations. Interview 8 caption the significant role of automation, advocating for the replacement of manual efforts with automated processes, particularly in the management of vulnerabilities: “There will always be some manual activities. However, the deployment must be done via automated means as often as possible. "Interview 6 however, also states that for automation to work, the process an organization wants to automate must be correct: "[…] You need to be sure that your process is correct, and don't iterate about automating, but iterate about the

process you are trying to automate." The underlying idea is that automation enforces consistency and reduces the margin for error.

Interview 3 takes this concept a step further, emphasizing the importance of integrating automated testing and monitoring from the very inception of a project. This means conducting static and dynamic security tests, along with threat modeling exercises, right from the project's outset: “You have places, where you can perform static, dynamic security tests and threat modeling from the starting phases." This approach ensures that security is not a one-time event but an ongoing, automated process. However, the interviewees have also reached consensus, that not everything can be automated, as interview 1 highlights "You can automate many tasks... Automation is really important." For automation you need a variety of tools, and as such additionally, within the vast landscape of available tools, Interview 5 points out the challenge of tool selection and integration: "The only problem is, I noticed tons of tools. The list is endless." The true test lies therefore not just in embracing automation but also in making informed decisions about which tools to incorporate into the DevSecOps pipeline. The goal is to select and integrate the right tools that best serve the organization's unique needs.

Another significant aspect of DevSecOps is the adoption of Infrastructure as Code (IaC), which is seen as a harbinger of the future by interview 2: “[…] it will become really important”, as IaC is a cloud platform that incorporates the DevSecOps principles from the start. Interview 1 also acknowledges the importance of IaC by emphasizing the necessity for structured and repeatable processes: " […] you need regularity and structure within the entire process." IaC entails defining and managing infrastructure through code, ensuring consistency and reliability—a crucial element in the world of DevSecOps. While most interviewees see the benefits of automated solutions, Interview 7 however pointed out that even with IaC, organizations should think about contingency and disaster recovery-like scenarios for when the cloud platform is sudden unavailable: "So you will also have to think about contingency and disaster recovery-like scenarios for automated solutions. I will soon have scripts if my automation does not work." This approach allows organizations to treat infrastructure provisioning and management with the same level of automation, version control, and testing as they apply to software development. In the DevSecOps journey, automation and the considerate selection of tools are integral to success. These practices not only enhance efficiency but also contribute to a more secure and reliable development and operations environment. Furthermore, the adoption of Infrastructure as Code (IaC) paves the way for a future where infrastructure provisioning and management align seamlessly with the principles of DevSecOps, fostering a culture of regularity, structure, and security throughout the entire process.

In conclusion, DevSecOps represents a holistic transformation, encompassing cultural, collaborative, security-integrated, compliant, and automated aspects. Embracing these principles leads to a more secure, efficient, and adaptable approach to software development and operations, setting the stage for a future where security is ingrained in every facet of the process.

6. Discussions and Conclusions

In this section, we will delve into the findings from both the literature review and the interviews, shedding light on the similarities, differences, and limitations of our research. Furthermore, we aim to provide answers to the main and sub-research questions that have guided our investigation. Also, areas for future research and the conclusions derived from the entire research will be discussed.

6.1 Overview of findings

To start with the similarities between the literature and interviews, we again will use the four domains as mentioned in the results of the literature research: People, Tools, Values, and finally Governance & Processes. With regards to people, both the insights drawn from the interviews and the findings in the literature converge on the crucial role of cultural transformation within the realm of DevOps and DevSecOps. The integration of security necessitates a profound shift in mindset and organizational culture, an aspect highlighted in both sources. Additionally, both the interviews and the literature accentuate the importance of building a foundational understanding of security among team members, even if they do not possess specialized expertise. Continuous learning and opportunities for training, particularly in secure coding practices, are recognized as essential for achieving success within DevSecOps. The concept of ownership and accountability for security is another shared emphasis.

When we look at the second important area of tools, then both the interviews and the literature recognize the paramount importance of automation and tools in the context of DevOps and DevSecOps. Automation is acknowledged as indispensable for achieving a more secure and efficient software development process. However, both sources sound a note of caution against over-reliance on tools, emphasizing the need for a well-judged selection and a thorough understanding of the complexities surrounding (the implementation of) such tools. Furthermore, robust automated testing capabilities are mentioned in both contexts, with automated testing being favored for consistency benefits. Also, the introduction of new security risks associated with cloud solutions is a point of discussion in both sources. Taken in account the third domain of values of DevSecOps, Continuous monitoring emerges as a recurring theme in both the interviews and the literature, highlighting the significance of measuring and monitoring the DevSecOps process to drive ongoing improvement. There is a shared concern about striking the right balance between speed and security, as an excessive emphasis on velocity can compromise the thoroughness of security practices. Both sources emphasize the importance of continuous improvement and the value of learning from experiences. The integration of security into the team's purview, or DevSecOps control, is also highlighted as beneficial in both contexts.

To summarize the similarities, with regards to the Governance & Processes domain, Management buy-in for security within the DevSecOps process is a point of agreement in both the interviews and the literature. The commitment and prioritization of security by senior management are seen as foundational to achieving success.

Differences

Although, beside the similarities and distinctions, several differences between the two results are noticeable. To start with a difference, the interviews mark the importance of

team members acquiring new skills and attaining a minimum level of security awareness. This is regarded as critical to ensuring that security becomes a shared responsibility. While the literature acknowledges the need for a new skill set, it does not delve as deeply into the granular details of individual team members responsibilities. Moreover, regarding legacy systems and technical debt, the literature references the challenge of grappling with legacy systems and technical debt, a time-consuming and resource-intensive endeavor. The interviews allude the need for change and adaptation but do not explicitly delve into the complexity of challenges related to legacy systems.

Furthermore, as velocity is often impacted by complexity and security requirements, the literature review stress the significance of striking a balance between velocity and security testing. It warns against overextending the DevSecOps process, potentially neglecting security testing. This trade-off is not as explicitly articulated in the interviews. Another difference goes in the direction of DevSecOps Control, wherein the literature findings emphasize the advantages of internalizing security within the DevSecOps team, affording greater control over security aspects. This aspect is not explicitly discussed in every interview. To continue with the differences, compliance with regulation is important within DevSecOps, both sources acknowledge the importance of compliance and regulations, but the interviews do not explicitly mention sectors subject to strict regulatory requirements, whereas the literature results highlight this challenge, especially in government and other regulated sectors. This is most likely caused by the fact that there are limited law-and regulations solely going into DevSecOps practices. Also, within the organization there are not yet standards and other documents written to adhere to the compliance of the upcoming DevSecOps practices.

To finalize the differences, we know that the utilization of tools is paramount to the successes of DevSecOps, however giving anyone full access rights to such tools is not recommended as briefly mentioned in the literature results. In the interviews, the need of access controls has not been mentioned as something important to uphold the security of the DevSecOps processes.

What Might Be Left Out or Less Emphasized

Besides the similarities, and differences, there are also several areas that are less emphasized by the interviewees. For example, User Experience and Customer-Centricity: The knowledge and information provided by the interview participants primarily focus on internal processes and technical aspects. Information regarding how DevSecOps impacts user experience and customer satisfaction was not extensively covered. If we dive further into the area of measurements and metrics, as mentioned in the chapter values, only one interviewee mentioned the importance of metrics, in such a more detailed discussion on specific Key Performance Indicators (KPIs), and Key Risk Indicators (KRIs) for DevSecOps success could be beneficial.

Furthermore, while collaboration was mentioned, the concept of cross-functional teams (cross-disciplinary), where individuals from different disciplines work together on a project, wasn't explicitly discussed in every interview. This is somewhat peculiar, as collaboration and communication between teams are seen as one of the greatest success factors of DevSecOps. The same applies for Continuous improvement, a fundamental aspect of

DevSecOps, but it might not have received enough attention from the interviewees. The organization should work on strategies for continuously enhancing DevSecOps processes, as this could add significant value.

6.2 Answers to research questions

The purpose of this section is to give an answer to the main research question. This main research question was: "What are the key security-related obstacles that organizations encounter when integrating DevSecOps/Secure DevOps, and how can these security challenges be effectively resolved or mitigated?" However, to give an answer to this research question, it is first important to answer the three sub research questions. As these sub questions, do support the possibility of giving an answer to the main research question. Finally, also an answer to the written hypothesis will be given.

Do DevSecOps/Secure DevOps practices impact the collaboration between security and development teams?

To give an answer to the first sub-research question, we must conclude that DevSecOps practices have a profound impact on the collaboration between security and development teams. DevSecOps represents a significant shift in how these teams work together. It promotes a cultural transformation that encourages shared responsibility, open communication, and the breakdown of traditional silos. In the past, security was often seen as an external gatekeeper, a separate function that sometimes-hindered development progress. In contrast, DevSecOps integrates security seamlessly into the entire software development lifecycle. This integration fosters a collaborative environment where security is considered everyone's responsibility, not solely that of a dedicated security team. It captions that security should be an intrinsic part of the development process, and not a separate, sometimes adversarial, function.

What are the differences between a shift left approach and a shift right approach in an Agile way of working?

In an Agile context, "shift left" and "shift right" are two distinct phases with different objectives. The "shift left" approach centers on moving activities and considerations that were traditionally performed later in the development process, to an earlier stage. In the (cyber)security sector this is better known for its more common distinction as security by design. Shift left strives to identify and address issues, including security vulnerabilities and defects, as early as possible in the development cycle. This approach encourages proactive testing, code reviews, and security checks during the development phase.

The "shift right" approach occurs after the software is deployed, primarily in the production environment. The focus is on real-time monitoring, feedback, and continuous testing in a live environment. It is designed to detect and respond to issues, such as security incidents, as they occur in real-time. It's a reactive approach aimed at improving the quality and security of software in the operational phase.

What type of shift approach would be the most beneficial for increased security?

The choice to implement a "shift left" approach, a "shift right" approach, or a combination of both is contingent on the specific project's objectives and the organization's overarching DevSecOps strategy. Several factors should be considered: Implementing a "shift left" approach is critical for identifying and mitigating security vulnerabilities, defects, and other issues at the earliest possible stages of development. This approach helps in preventing many issues from reaching the production environment, thus reducing risks and costs associated with post-deployment problems. On the other hand, a "shift right" approach is indispensable for real-time monitoring, feedback, and prompt issue resolution within the operational phase. It contributes to enhancing the security and performance of software in a live environment.

In practice, a combination of both approaches is often the most effective strategy. This combined approach enables organizations to reap the benefits of proactive measures taken during development (shift left) while maintaining continuous monitoring and testing in the operational phase (shift right). This approach provides a holistic DevSecOps strategy that addresses security and quality throughout the software's lifecycle. To fully benefit from the infinity circle, organizations should therefore employ a holistic approach that combines proactive measures during development (shift left) with continuous monitoring and testing in the operational phase (shift right). This approach provides a comprehensive DevSecOps strategy that addresses security and quality throughout the software's lifecycle. The choice of approach or combination should ultimately align with the organization's priorities, resources, and risk management strategy.

Main research question: "What are the key security-related obstacles that organizations encounter when integrating DevSecOps/Secure DevOps, and how can these security challenges be effectively resolved or mitigated?"

The main research question guiding this study is centered on understanding the most significant security-related challenges faced by organizations when adopting DevSecOps/Secure DevOps and exploring potential solutions for these challenges. The challenges have been categorized in the beforementioned four key domains: People, Tools, Values, and Governance & Processes.

People

Cultural transformation stands out as a fundamental challenge in the DevSecOps adoption process. This transformation entails a shift from traditional, isolated silos to a culture of collaboration, shared security responsibilities, trust, transparency, and continuous learning. Achieving this cultural shift requires several key strategies. First and foremost, leadership buy-in is vital. Executives who endorse the DevSecOps approach set the tone for creating a secure and collaborative environment. They provide essential resources and support necessary for the teams to thrive. Furthermore, team skills and security awareness play a critical role. Management must provide regular training opportunities to help team members develop the skills and knowledge essential for DevSecOps. Encouraging mutual learning within the team, such as pairing experienced mentors with newer members, promotes knowledge sharing and skill development. To further complement this, an effective communication and collaboration must be central, however this is often another challenge within DevSecOps. To overcome this challenge, regular meetings and Agile/Scrum stand-ups offer spaces for discussions about ongoing projects, security challenges, and other

pertinent matters. By implementing collaborative platforms such as Jira and Confluence this facilitates the communication and knowledge sharing between teams.

However, as DevSecOps can be a complex and technical paradigm, gaining a common terminology is a challenge that must be faced to ensure clear communication. Teams must use common language and terminology to ensure effective communication within the DevSecOps environment. Moreover, the lack of security awareness can be addressed by providing regular security training sessions, covering various security-related topics. These sessions help team members understand the importance of working securely and recognize their roles in maintaining security. Ownership and accountability issues can be overcome by recognizing and rewarding team members, stimulating incentives, and communicating the benefits of DevSecOps in terms of enhanced security, faster software development, and reduced security weaknesses.

Tools

Delving in the realm of tools, selecting the right tools for DevSecOps practices is a crucial but challenging task. Organizations should conduct thorough evaluations to choose tools that align with their objectives and integrate seamlessly into their development processes. To make sure that the teams have access to the right tools for the job, a periodic evaluation and open risk dialogues with DevSecOps team members are essential for ensuring that tools remain effective and up to date.

Furthermore, automated security testing tools are critical for streamlining security practices and identifying vulnerabilities within the software code. However, organizations should start small and allow time for adjustment when adopting automation tools. Although, to ensure the security of the entire process, proper access restrictions for these tools, such as Role-Based Access Control (RBAC), are essential to ensure security. The organization must choose an access control method that aligns with its needs and preferences. And finally, legacy systems often introduce security risks that cannot be quickly mitigated. To address this, organizations should focus on incremental improvements, starting with the most critical components and gradually updating or rebuilding them. Organizations should also think of compensating controls to reduce the risk to a temporarily acceptable risk level.

Values

Continuous monitoring is an essential component of the DevSecOps approach. It requires various activities, including real-time detection capabilities, anomaly detection using machine learning or artificial intelligence, integration of threat intelligence feeds, automated alerts, workflow tools like Jira, and a continuous improvement mindset. Automated remediation responses can also be explored and implemented to intercept unwanted behaviors and incidents and take immediate remedial actions to mitigate vulnerabilities or threats.

Governance & Processes

Progressing in the final domain of Governance & Process, we know from the people domain that management buy-in is crucial for fostering a collaborative and security-conscious culture in the DevSecOps environment. Management buy-in must lead to the establishment of proper governance structures and processes, essential for the success of DevSecOps

practices. Continuing with compliance and regulation needs, these mandatory activities can be addressed by hiring compliance (legal) experts and integrating them into the DevSecOps team. Furthermore, automated compliance checks are essential to ensure code correctness and regulatory compliance.

In summary, to address the main research question, the challenges organizations face when adopting DevSecOps are multifaceted. These challenges encompass cultural transformation, skill development, continuous learning, ownership and accountability, thoughtful tool selection, compliance with regulatory standards, and balancing the need for speed with security. Importantly, these challenges are interconnected, and their extent varies depending on the unique context of each organization. Above several solutions for overcoming these challenges have been given, in short, these solutions involve embracing cultural evolution, fostering collaborative harmony, refining skills, and committing to continuous improvement. Through this comprehensive approach, organizations can navigate DevSecOps hurdles and realize heightened security, enhanced agility, and efficient software delivery in a changing landscape that is DevSecOps.

Hypothesis

Our initial hypothesis, which suggested that cultural alignment might not particularly represent the predominant challenge, is substantiated by our findings. While cultural transformation undeniably is seen as a significant hurdle, the specific challenges manifest diversely across organizations. Factors such as tool selection and adherence to industry-specific regulatory frameworks can emerge as formidable barriers, the extent of these barriers however are also dependent upon the unique context of each organization.

6.3 Limitations

It is essential to acknowledge potential limitations of this research. Limitations include a limited generalizability because only individuals of one specific organization have been interviewed. The results may therefore not represent the entire DevSecOps landscape. Another limitation is time constraint, as the duration of this research may limit the depth of the data collection. And, finally there might also be a self-reporting bias from the interview participants as their responses might be influenced by their perceptions and in such may not always reflect an objective reality of common challenges and/or solutions. However, the results of this research open the way forward for a strategy to successfully implement DevSecOps.

6.4 Areas for future research

As we are almost ready to conclude our comprehensive exploration of DevSecOps, it's important to look forward to future research directions that can further enhance the DevSecOps landscape and address evolving challenges. One crucial area for future work is the development of automated artifacts of evidence to support continuous monitoring and auditing purposes throughout the Software Development Life Cycle (SDLC) and the Continuous Integration/Continuous Deployment (CI/CD) pipeline. These artifacts can encompass a wide range of security-related data, including risk registers and threat/vulnerability reports. The automation of evidence generation has the potential to significantly improve security, elevate organizational maturity, and support information-

driven decision-making. Automated evidence gathering is also a huge benefit for compliance reasons. This becomes especially relevant considering forthcoming regulations such as NIS2, which demand a thorough understanding of an organization's security posture. Another useful artifact of evidence is the so-called Software Bill of Materials (SBoM), another promising avenue for future research. SBoMs provide comprehensive evidence of all software components used within a specific code build, aiding in supply chain dependency awareness and vulnerability assessment [38].

Investigating how to effectively implement continuous monitoring and testing capabilities within enterprise organizations is another area of interest, ensuring that security remains a dynamic and proactive aspect of software development. Furthermore, establishing a safe DevSecOps culture within an enterprise organization and integrating it into existing ITIL processes is an area ripe for exploration. Creating a culture aligned with DevSecOps principles is vital for long-term success, and harmonizing this culture with established ITIL practices can promote synergy and efficiency.

6.5 Conclusion

In this study, we've explored the realm of DevSecOps, uncovering a transformative paradigm that is reshaping software development. This shift merges development, security, and operations, making security a core element at every development phase. DevSecOps isn't merely an extension of DevOps with added security layers; it represents a cultural metamorphosis within organizations. It thrives on collaboration, shared responsibilities, and an unwavering security commitment.

Our investigation has delved into the profound impact of DevSecOps practices on the collaboration between security and development teams. DevSecOps breaks down traditional silos, fostering communication and cooperation. Teams collaboratively share security responsibilities, reducing the risk of vulnerabilities in the production environment. We encourage the integration of shift-left and shift-right testing approaches, combining early security by design with post-development production testing.

Addressing our main research question, we've unveiled a web of challenges in DevSecOps adoption, spanning People, Tools, Values, and Governance & Processes. These challenges encompass cultural transformation, skill development, tool selection, access controls, automation, legacy systems, cloud security, monitoring, balancing speed and security, continuous improvement, management support, and compliance. In summary, DevSecOps represents a transformative journey towards secure and efficient software development. Organizations must adopt a holistic approach that focuses on cultural evolution, collaborative harmony, skill refinement, and a commitment to continuous improvement. By embracing this comprehensive approach, organizations can navigate DevSecOps challenges and reap the rewards of heightened security, enhanced agility, and efficient software delivery. The interconnectedness of these challenges accentuate that DevSecOps isn't just a methodology; it's an organizational and cultural shift shaping the future of software development, leading to a more secure and agile future.

Bibliography

[1] J. Towner, "What is the Difference Between Agile and Waterfall?," Forecast, 2022 6 27. [Online]. Available: https://www.forecast.app/blog/difference-between-agile-waterfall#:~:text=Agile%20and%20waterfall%20are%20two,in%20a%20more%20linear%20process.. [Accessed 28 7 2023].

[2] Delta-N, "Wat is DevOps?," Delta-N, [Online]. Available: https://www.delta-n.nl/devops/wat-is-devops/. [Accessed 28 7 2023].

[3] Q. Anx, "The DevSecOps Cultural Transformation," PagerDuty Blog, 18 3 2023. [Online]. Available: https://www.pagerduty.com/blog/devsecops-ops-guide/. [Accessed 29 7 2023].

[4] G. Sign, "Understanding DevSecOps and its Role in CI/CD," Xenanstack, 31 5 2023. [Online]. Available: https://www.xenonstack.com/insights/devsecops-in-ci-cd. [Accessed 31 7 2023].

[5] Redhat, "What is CI/CD?," Redhat, 11 5 2022. [Online]. Available: https://www.redhat.com/en/topics/devops/what-is-ci-cd. [Accessed 31 7 2023].

[6] ITSupplychain, "Reactionary approach to security is hindering business according to new study," ITSupplychain, 30 3 2023. [Online]. Available: https://itsupplychain.com/reactionary-approach-to-security-is-hindering-businesses-according-to-a-new-study/. [Accessed 25 10 2023].

[7] medium.com, "Understanding and aligning culture for high performance DevSecOps," Medium.com, 31 5 2023. [Online]. Available: https://medium.com/devsecops/aligning-culture-for-high-performance-devsecops-b47ec2f54a64. [Accessed 7 8 2023].

[8] J. Kent, "Study: 71% of Decision Makers Believe Culture is a Top Barrier to DevSecOps Success," Progess.com, 8 12 2022. [Online]. Available: https://www.progress.com/blogs/study-71-decision-makers-believe-culture-top-barrier-devsecops-success. [Accessed 7 8 2023].

[9] E. Hofstee, "Extract from Constructing a Good Dissertation," Exactica, 2006. [Online]. Available: https://www.exactica.co.za/dn/exactica-book-literature-review.pdf. [Accessed 8 2023].

[10] "DoD Enterprise DevSecOps," Department of Defense, 19 8 2019. [Online]. Available: https://dodcio.defense.gov/Portals/0/Documents/DoD%20Enterprise%20DevSecOps%20Reference%20Design%20v1.0_Public%20Release.pdf. [Accessed 16 8 2023].

[11] I. Buchanan, "History of DevOps," Atlassian, [Online]. Available: https://www.atlassian.com/nl/devops/what-is-devops/history-of-devops. [Accessed 7 8 2023].

[12] Atlassian, "DevOps - DevOps aligns development and operations to optimize quality and delivery.," Atlassian, [Online]. Available: https://www.atlassian.com/nl/devops. [Accessed 7 8 2023].

[13] R. Roman, "10 SUCCESS FACTORS FOR THE DEVOPS CULTURE," Nasscom, 19 6 2019. [Online]. Available: https://community.nasscom.in/communities/it-services/10-success-factors-for-the-devops-culture.html. [Accessed 8 8 2023].

[14] A. Ozanich, "Understanding the DevOps Pipeline & How to Build One," HubSpot, 14 9 2022. [Online]. Available: https://blog.hubspot.com/website/devops-pipeline. [Accessed 11 8 2023].

[15] R. Mohana, "What Is DevOps? Definition, Goals, Methodology, and Best Practices," Spiceworks, 11 8 2021. [Online]. Available: https://www.spiceworks.com/tech/devops/articles/what-is-devops/. [Accessed 12 8 2023].

[16] Agility.im, "Incremental VS iterative development?," Agility.im, [Online]. Available: https://agility.im/frequent-agile-question/difference-incremental-iterative-development/. [Accessed 11 8 2023].

[17] M. R. Chrissy Kidd, "Agile vs DevOps: A Full Comparison," BMC, 16 7 2021. [Online]. Available: https://www.bmc.com/blogs/devops-vs-agile-whats-the-difference-and-how-are-they-related/. [Accessed 11 8 2023].

[18] S. Pittet, "Continuous integration vs. delivery vs. deployment," Atlassian, [Online]. Available: https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment. [Accessed 13 8 2023].

[19] Theserverside, "Why GitHub renamed its master branch to main," [Online]. Available: https://www.theserverside.com/feature/Why-GitHub-renamed-its-master-branch-to-main. [Accessed 13 8 2023].

[20] A. Butch, "30 Best DevOps Tools to Learn and Master In 2023: Git, Docker & More," Simplilearn.com, 9 8 2023. [Online]. Available: https://www.simplilearn.com/tutorials/devops-tutorial/devops-tools. [Accessed 13 8 2023].

[21] "Development Value Streams," Scaledagileframework, 14 11 2022. [Online]. Available: https://scaledagileframework.com/development-value-streams/. [Accessed 12 8 2023].

[22] E. Poulson, "DevOps Hits Speed Bump in Race to Cloud Native," Nutanix.com, 20 8 2021. [Online]. Available: https://www.nutanix.com/theforecastbynutanix/news/devops-adoption-stalling-due-to-cloud-native-strategies. [Accessed 12 8 2023].

[23] I. Buchanan, "Infrastructure as Code," Atlassian, [Online]. Available: https://www.atlassian.com/nl/microservices/cloud-computing/infrastructure-as-code. [Accessed 12 8 2023].

[24] True.nl, "Continuous Integration, Continuous Delivery en Continuous Deployment (CI/CD) uitgelegd," true.nl, 20 7 2020. [Online]. Available: https://www.true.nl/blog/ci-cd-uitgelegd/. [Accessed 18 8 2023].

[25] K. Zettler, "DevSecOps Tools," Atlassian - DevSecOps, [Online]. Available: https://www.atlassian.com/devops/devops-tools/devsecops-tools. [Accessed 15 8 2023].

[26] A. Sugandhi, "DevOps vs DevSecOps: Top Differences," knowledgehut.com, 14 7 2023. [Online]. Available: https://www.knowledgehut.com/blog/devops/devops-vs-devsecops. [Accessed 15 8 2023].

[27] S. Moreels, "Continuous Refactoring," Codit.eu, 28 2 2017. [Online]. Available: https://www.codit.eu/blog/continuous-refactoring/. [Accessed 17 8 2023].

[28] N. Rosier, "Refactoring," Globalorange.nl, [Online]. Available: https://www.globalorange.nl/refactoring/#:~:text=Refactoring%20(of%20refactoren)%20is%20het,jouw%20product%20meer%20future%2Dproof.. [Accessed 27 8 2023].

[29] D.O.D., "DevSecOps Playbook," D.O.D., 9 2019. [Online]. Available: https://dodcio.defense.gov/Portals/0/Documents/Library/DevSecOps%20Playbook_DoD-CIO_20211019.pdf. [Accessed 18 8 2023].

[30] R. v. S. &. S. v. Solms, "Information Security Governance: A model based on the Direct–Control Cycle," Elsevier, 9 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404806001167. [Accessed 18 8 2023].

[31] R. '. Hart, "Naris GRC, Governance, Risk en Compliance software," robertthart.risicomanagement.nl, [Online]. Available: https://robertthart.risicomanagement.nl/2019/03/23/three-lines-of-defense-risicomanagement-model/. [Accessed 18 8 2023].

[32] V. Vicente, "How to Develop Key Risk Indicators (KRIs) to Fortify Your Business," Auditboard, 8 5 2023. [Online]. Available: https://www.auditboard.com/blog/how-to-develop-key-risk-indicators-kris-to-fortify-business/#:~:text=Key%20risk%20indicators%20are%20metrics,monitor%2C%20manage%20and%20mitigate%20risks.. [Accessed 26 8 2023].

[33] G. Alvarenga, "DEVOPS VS. DEVSECOPS:," Crowdstrike, 15 9 2022. [Online]. Available: https://www.crowdstrike.com/cybersecurity-101/cloud-security/devops-vs-devsecops/. [Accessed 19 8 2023].

[34] L. Rath, "9 Best DevSecOps Tools," iTT Systems, 18 11 2023. [Online]. Available: https://www.ittsystems.com/best-devsecops-tools/. [Accessed 7 8 2023].

[35] Owasp, "OWASP Top Ten," OWASP.org, [Online]. Available: https://owasp.org/www-project-top-ten/.

[36] S. Gunja, "Shift left vs shift right: A DevOps mystery solved," Dynatrace, 7 2 2023. [Online]. Available: https://www.dynatrace.com/news/blog/what-is-shift-left-and-what-is-shift-right/. [Accessed 23 8 2023].

[37] Softwareimprovementgroup, "Security by design in 9 steps," Softwareimprovementgroup, 27 2 2018. [Online]. Available: https://www.softwareimprovementgroup.com/security-by-design-in-9-steps/. [Accessed 26 8 2023].

[38] NCSC, "Software Bill of Materials (SBoM) en cybersecurity," NCSC, [Online]. Available: https://www.ncsc.nl/onderzoek/onderzoeksresultaten/software-bill-of-materials-sbom-en-cyber-security-map. [Accessed 25 8 2023].

[39] Redhat, "What is DevSecOps?," Redhat, 10 3 2023. [Online]. Available: https://www.redhat.com/en/topics/devops/what-is-devsecops. [Accessed 25 7 2023].

[40] N. Barney, "Microsoft App-V (Microsoft Application Virtualization)," TechTarget, 4 2023. [Online]. Available: https://www.techtarget.com/searchvirtualdesktop/definition/Microsoft-App-V-Microsoft-Application-Virtualization. [Accessed 12 8 2023].

[41] Statista, "Number of data records exposed worldwide [...]," Statista.com, [Online]. Available: https://shorturl.at/erBLP. [Accessed 15 8 2023].

[42] A. V. a. H. Yasar, "Achieving authority to operate and risk management," Federal News Network, 24 7 2019. [Online]. Available: https://federalnewsnetwork.com/commentary/2019/07/achieving-authority-to-operate-and-risk-management/. [Accessed 18 8 2023].

[43] P. Putz, "Kubernetes vs Docker: What's the difference?," Dynatrace, 2 3 2023. [Online]. Available: https://www.dynatrace.com/news/blog/kubernetes-vs-docker/#:~:text=Docker%20is%20a%20suite%20of,application%20code%20and%20dependencies%20inside.. [Accessed 18 8 2023].

[44] S. Ingalls, "10 Best DevSecOps Tools," esecurity planet, 17 3 20222. [Online]. Available: https://www.esecurityplanet.com/products/devsecops-tools/. [Accessed 19 8 2023].

[45] "What is software composition analysis?," Synooposis, [Online]. Available: https://www.synopsys.com/glossary/what-is-software-composition-analysis.html#:~:text=Definition,source%20license%20limitations%20and%20obligations.. [Accessed 20 8 2023].

[46] E. D. Frauenstein, "A Framework to Mitigate Phishing Threats," 1 2013. [Online]. Available: https://www.researchgate.net/publication/267512601_A_Framework_to_Mitigate_Phishing_Threats. [Accessed 8 2023].

[47] L. Smits, "How do you transcribe an interview? | Examples & software," Scribbr, 23 12 2021. [Online]. Available: https://www.scribbr.nl/onderzoeksmethoden/interview-transcriberen/. [Accessed 17 9 2023].

[48] Gartner, "Customer Centricity," Gartner.com, [Online]. Available: https://www.gartner.com/en/marketing/glossary/customer-centricity#:~:text=Customer%20centricity%20demands%20that%20the,customer%20satisfaction%2C%20loyalty%20and%20advocacy.. [Accessed 17 9 2023].

Appendices

Appendix 1 Set of interview questions:

1. Do you have experience with Devops, DevSecOps, secure DevOps? If yes, how much years of experience do you have?
2. How do you define, DevOps and DevSecOps?
3. Are you familiar with the DevSecOps principles? What principles do u use?
4. Were you able to implement DevSecOps, or security in DevOps within an organization?
    a. If yes, how would the effectiveness of this DevSecOps implementation be assessed?
5. What challenges did you face, when implementing DevSecOps?
        i. How did you address these challenges?
6. Do you think, collaboration and communication within DevSecOps practices is important? If yes, how so?
    a. Can you also explain how you would promote collaboration and communication within a DevSecOps culture?
    b. If no, please elaborate?
7. In a DevOps workflow, how does security get prioritized?
8. Are you familiar with the shift left/shift right concepts of DevOps/DevSecOps?
    a. If yes, what shifting approach do you need to integrate in a DevSecOps approach?
9. Can you tell me which common security tools there are being used in DevSecOps?
    a. Do you work with these security tools yourself?
        i. What are the benefits and/or drawbacks of using these tools?
    b. Are you familiar with DAST, SAST, SCA, IAST etc.?
        i. Do you know if there are weaknesses/drawbacks within these security testing methods?
10. Do you think that logging is important in a DevSecOps approach?
11. How can you make sure that a DevSecOps environment, meets compliance requirements?
12. Are you also familiar with the concept of a CI/CD pipeline?
    a. If yes, can you tell me what does this mean for you?
    b. Can you also tell, how security must be implemented within a CI/CD pipeline?
13. Do you have experience with DevSecOps in cloud environments?
    a. If yes, does a cloud environment have other DevSecOps requirement, challenges etc., than a locally/on-premises environment?
    b. Are you familiar with terms as Infrastructure as Code (IaC)?
        i. If yes, can you tell me why this is important within DevSecOps?
14. Is automation everything within DevSecOps really the answer, or can you still perform manual tasks?
15. Can you tell me, from your experiences if there are other, important aspects, components etc. that do improve/increases the security within DevSecOps/ CI/CD pipelines?
16. To wrap it up, do you see from your own experience other challenges when adopting DevSecOps, than those challenges mentioned before in this interview?

Appendix 2 Grounded theory

Below are the elaborations of the grounded theory that has been applied to the interview transcripts.

Interview 1 Coding Elements:

1. **DevOps Experience:** Highlighted 6-7 years of experience in DevOps.
2. **Integration of DevOps:** DevOps as a culture integrating development and operations.
3. **Shared Responsibility:** Emphasized shared responsibility for security.
4. **Security by Design:** Mentioned the need for security from the design phase.
5. **Mindset and Culture:** Discussed the importance of mindset and culture change.
6. **Communication:** Stressed the significance of communication and collaboration.
7. **Automation:** Discussed automation within the CI/CD pipeline.
8. **Tools:** Mentioned Ivanti automation, SCCM, PowerShell, PowerBI, and more.
9. **Logging:** Highlighted the importance of logging for troubleshooting.
10. **Continuous Improvement:** Discussed the need for continuous improvement.
11. **Ownership & Accountability:** Emphasized team ownership and accountability.

Interview 2 Coding Elements:

1. **DevOps Transition:** Described transitioning to DevOps from a traditional environment.
2. **Cultural Change:** Emphasized the cultural change required for DevOps.
3. **Security Integration:** Discussed integrating security practices.
4. **Security Mindset:** Mentioned the security by design mindset.
5. **Communication & Collaboration:** Highlighted the need for effective communication.
6. **Tools and Automation:** Mentioned tools like Jenkins and JIRA.
7. **Compliance:** Discussed meeting compliance and regulatory requirements.
8. **Ownership & Accountability:** Talked about individual accountability.
9. **Continuous Learning:** Emphasized the need for continuous learning.

Interview 3 Coding Elements:

1. **DevOps and DevSecOps Experience:** Described experience in DevOps and DevSecOps.
2. **Culture Change:** Discussed the cultural change needed for DevSecOps.
3. **Security Integration:** Emphasized integrating security from the design phase.
4. **Communication and Collaboration:** Highlighted effective communication.

5. **Continuous Testing:** Mentioned automated testing and monitoring.
6. **Tool Selection:** Discussed tools for DevSecOps processes.
7. **Security Awareness:** Mentioned raising security awareness.
8. **Infrastructure as Code (IaC):** Discussed the shift towards IaC.
9. **Automation:** Talked about automating manual tasks.
10. **Compliance:** Discussed meeting compliance and regulatory requirements.

Interview 4 Coding Elements:

1. **DevOps Implementation:** Discussed implementing DevOps.
2. **Security Awareness:** Emphasized the need for security awareness.
3. **Cultural Shift:** Mentioned the cultural shift towards DevOps.
4. **Security Integration:** Highlighted integrating security into DevOps.
5. **Communication:** Discussed the importance of communication.
6. **Collaboration:** Mentioned collaboration between teams.
7. **Automation:** Talked about automation and CI/CD pipelines.
8. **Infrastructure as Code (IaC):** Discussed IaC and its importance.
9. **Tools:** Mentioned tools like Ansible, Terraform, and Jenkins.
10. **Continuous Improvement:** Emphasized continuous learning and improvement.

Interview 6 Coding Elements:

1. **DevOps and DevSecOps Experience:** Described experience in DevOps and DevSecOps.
2. **Culture Change:** Discussed the cultural change needed for DevSecOps.
3. **Security Integration:** Emphasized integrating security from the design phase.
4. **Communication and Collaboration:** Highlighted effective communication.
5. **Continuous Testing:** Mentioned automated testing and monitoring.
6. **Security Tools:** Discussed Ivanti automation and SCCM.
7. **Infrastructure as Code (IaC):** Talked about transitioning to IaC.
8. **Automation:** Emphasized automation of manual tasks.
9. **Compliance:** Mentioned meeting compliance and regulatory requirements.
10. **Ownership & Accountability:** Discussed accountability for security.

Interview 7 Coding Elements:

1. **DevOps and DevSecOps Experience:** Described experience in DevOps and DevSecOps.
2. **Cultural Change:** Emphasized the challenge of changing mindset and culture.
3. **Security Integration:** Discussed incorporating security from the design phase.
4. **Communication and Collaboration:** Highlighted effective communication.
5. **Continuous Testing:** Mentioned automated testing and monitoring.
6. **Security Tools:** Discussed Ivanti automation, SCCM, and PowerBI.
7. **Infrastructure as Code (IaC):** Talked about transitioning to IaC.
8. **Automation:** Emphasized automation of manual tasks.
9. **Compliance:** Mentioned meeting compliance and regulatory requirements.
10. **Ownership & Accountability:** Discussed accountability for security.

Interview 8 Coding Elements:

1. **DevOps Transition:** Described transitioning to DevOps from a traditional environment.
2. **Security Integration:** Emphasized integrating security practices seamlessly.
3. **Cultural Change:** Mentioned the cultural shift required for DevOps.
4. **Communication & Collaboration:** Highlighted effective communication and collaboration.
5. **Tools and Automation:** Discussed the role of tools and automation.
6. **Ownership & Accountability:** Emphasized team ownership and accountability.
7. **Continuous Learning:** Talked about the need for continuous learning.
8. **Compliance:** Mentioned meeting compliance and regulatory requirements.

Note: In this version, the verbatim quotes of all interviews are redacted, to ensure the anonymity of all interview participants.