

# Lean security



A framework of security activities and  
design factors for DevSecOps

Foreword by Jez Humble

Student: Dennis Verslegers  
Promotor: Prof. dr. Yuri Bobbert



Antwerp  
Management  
School

# Lean security

THESIS

Submitted for the degree of

MASTER OF SCIENCE

IT Governance and Assurance

by

Dennis Verslegers

with promotor

Dr. Yuri Bobbert

at the

Antwerp Management School

May 2020

Noteworthy accomplishments are rarely achieved alone. A tremendous group of people have, directly or indirectly, contributed to allow me to start and finish this research project. So many that I risk not mentioning some of them, I apologise in advance. I would like to express my gratitude and say thank you ...

To my father who did not live to see this moment and would probably never have thought it to be possible,

To my mother who was always convinced that anything was possible, contrary to the odds, and thereby gave me the confidence to take chances in life,

To my wife and daughter for putting up with me during stressful moments and supporting me through this 2 year journey, it takes sacrifice to allow someone to chase their dreams and aspirations. I hope to be able to do the same for the both of you.

To my promotor, for his support throughout this research project. His enthusiasm and dedication have made this project that much more ambitious by challenging me and asking critical questions at the right time. I'm grateful for your support and dedication.

To each of the 20 experts who contributed to this research, they dedicated a significant part of their valuable time and were the ones who made this research possible. Without their answers to the surveys, interviews and online sessions or review of this paper there would be nothing to write about. I could have never expected such voluntarism and I am grateful for that. I will follow your example and seek to contribute to other peoples projects in the same enthusiastic way you supported mine.

---

## Foreword

A key goal of DevOps is to increase software delivery velocity while also improving service reliability. However many of the practices that enable these outcomes require significant shifts in the ways teams deliver software. In particular, software has traditionally been released infrequently in large batches, with QA and security reviews happening after development is complete.

A significant change has been the adoption of continuous delivery, in which teams ensure the software is always in a releasable state. That means teams performing testing and other activities traditionally performed downstream of development as part of the development process.

This new way of working requires changes in process, training, skills, and organizational culture for all roles, including information security. However the marriage of DevOps and information security, while ever more critical in today's world of botnets and advanced persistent threats, has received relatively little attention in the academic literature.

Dennis Verslegers's work addresses this gap. Using the literature and input from experts, he has produced a library of practices that can be used to achieve important control objectives which are designed to complement modern ways of working, and assessed their effectiveness, impact on delay, and financial consequences, as well as considering relevant design factors.

This work represents an important and valuable contribution which will help organisations as they redesign information security and governance, risk and compliance in a continuous world. When applied well, this should help organisations improve their security posture even as they deliver software more rapidly and reliably - a win-win for everybody.

*Jez Humble,*

*Author of the Jolt Award-winning Continuous Delivery,  
Lean Enterprise and co-author of The DevOps Handbook.*

---

# Introduction

The digital world is an important playing field in this era of innovation and business transformation. Meeting market demands requires continuous change to digital platforms. The speed with which this change can be delivered is detrimental for the competitive advantage of an organisation. To satisfy the demand for speed and agility new ways of organising work are explored and gaining in popularity rapidly. As software becomes an integral element of business growth the focus on fast delivery of features with a tangible business value has increased. Organisations want the ability to seize opportunities without being stopped in their tracks.

Increasing pressure from regulators and a decreasing tolerance for security breaches by customers is reducing the risk appetite of key stakeholders and investors. Business value is only achieved if it is done in a reliable and secure way. This leads to increasing attention for secure business value creation.

The challenge which presents itself is increasing the security characteristics of our digital platforms without sacrificing speed and agility. The security industry often states that rapid change increases security risk. This does not necessarily hold true as security itself benefits from the ability to implement changes quickly. It allows a system to react to newly discovered vulnerabilities and the ever changing threat landscape. What is considered secure today may not necessarily hold true tomorrow. The only way to safeguard business value is through rapid change.

The outcome of this research aims to provide a framework of validated security activities for DevSecOps environments ranked by their characteristics to improve security without sacrificing speed and agility. This framework should allow organisations to build a “lean” approach to security.

---

# Table of contents

Part one: drivers & approach	1
Background	1
Problem statement	4
Main research question	7
Selection of research methods	7
Design problems and knowledge questions	9
Research approach	11
Research deliverables	12
Part two: Research	15
Academic literature review	15
Searching the knowledge base	15
Analysis of the dataset	19
Validation and prioritisation of the research findings	25
Preparations	25
Validation and elaboration of findings by expert panel	27
Prioritisation by expert panel	34
Part three: Analysis	39
Collaboration	40
Performing continuous feedback from production to development	40
Provide security training	42
Establish security satellites	44
Practice incident response	46
Establish a security mindset across the organisation	47
Use of non-automated activities	48
Performing security requirements analysis	48
Performing threat modeling	50
Performing risk analysis	53
Establishing security SLA's for cloud providers	55
Performing continuous assurance	57

---

Performing manual security testing	59
Secure the CI/CD pipeline	62
Use of automated activities	64
Performing automated security testing	64
Performing continuous monitoring	70
Implement automated remediation	78
Performing security configuration automation	80
Implement secrets management	81
Manage digital supply chain	82
Part four: Results	87
Answers to research questions	87
Conclusion	93
Limitations	96
Future research	97
Appendices	1
Appendix A: Literature review notes	2
Appendix B: Thematic analysis mapping table	9
Appendix C: thematic analysis details	34
Thematic analysis on security activities	34
Thematic analysis on design factors	35
Appendix D: Expert interviews	39
Interview 1: Cyber Security Designer	39
Interview 2: Cyber Security Designer	41
Interview 3: DevSecOps advisor	43
Appendix E: knowledge nomination worksheet	45
Appendix F: analysis of expert elaborations	47
Appendix G: Results of prioritisation by experts	51
List of figures	56
List of tables	57
References	58
Credits	62

---

# Part one: drivers & approach

---

This part of the paper provides an overview of the drivers and approach for this research project. It consists of the following sections:

- **Background:** specifies the context in relevant to this research project
- **Analysis of the dataset:** presents the results of the thematic analysis applied on the identified gold-set
- **Validation of the research findings:** presents the results of the expert validation & elaboration and the expert prioritisation sessions

## Background

Software application development and IT infrastructure operations have undergone rapid change over the past decade [1]. One of the most discussed changes is the gradual shift from traditional, longer running, procedural approaches to dynamic and iterative processes [2].

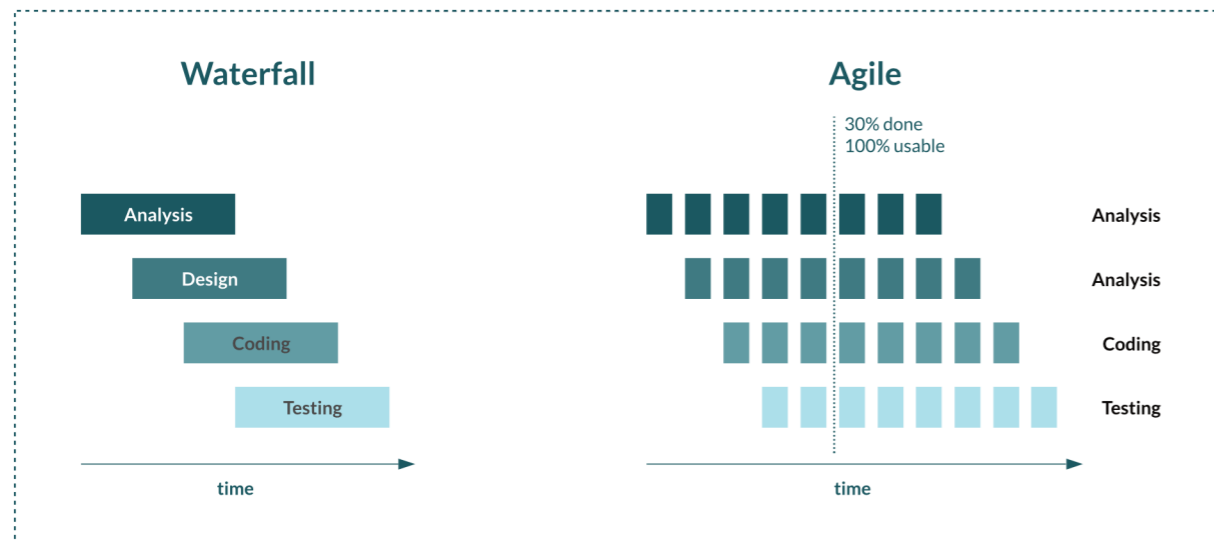
An important driver for this change is the agile development approach as proposed by the manifesto for agile software development. The Agile Manifesto [3] defines the following core principles: (a) people and interactions over processes and tools, (b) working software instead of detailed documentation, (c) active customer participation and involvement rather than time and effort expended on negotiating contracts, and (d) willingness and ability to take on changes over steadfast commitment to a static plan.

As stated by Abrahamsson et al. [4] the main focus of light and agile methods are simplicity and speed. A simplified comparison between procedural, also referred to as waterfall, and agile software development practices is depicted below.

*"Conventional assurance methods, applied naïvely, would create deterring delays between critically short iterations as well as prohibitively inflate the development budgets."*

*Beznosov & Kruchten, 2004 [14]*

Figure 1: Comparison between the typical process steps in waterfall and agile development as explained by Abrahamsson et al. [4]



The impact of these simple tenants has not been limited to development teams. New demands are placed onto operations teams as a result of the changes in development approach [5]. The paradigm shift to focus on small, iterative software deliveries at a high pace adds dynamic complexity to IT operations. Speed in terms of software development translates into a need for fast deployments with a short delta time between release and availability in the production environment [2].

The introduction of agile methodologies in an organisation brings, besides the numerous benefits, also some challenges. From a process perspective some of these challenges include areas such as functional and non-functional requirements identification, tracking of projects, quality management and risk management [6]. Traditional decision-making strategies such as hierarchical approval or review by a technical board no longer fit the increased speed with which development and operations are moving resulting in delegation of authority and end-to-end responsibility for the various teams [7].

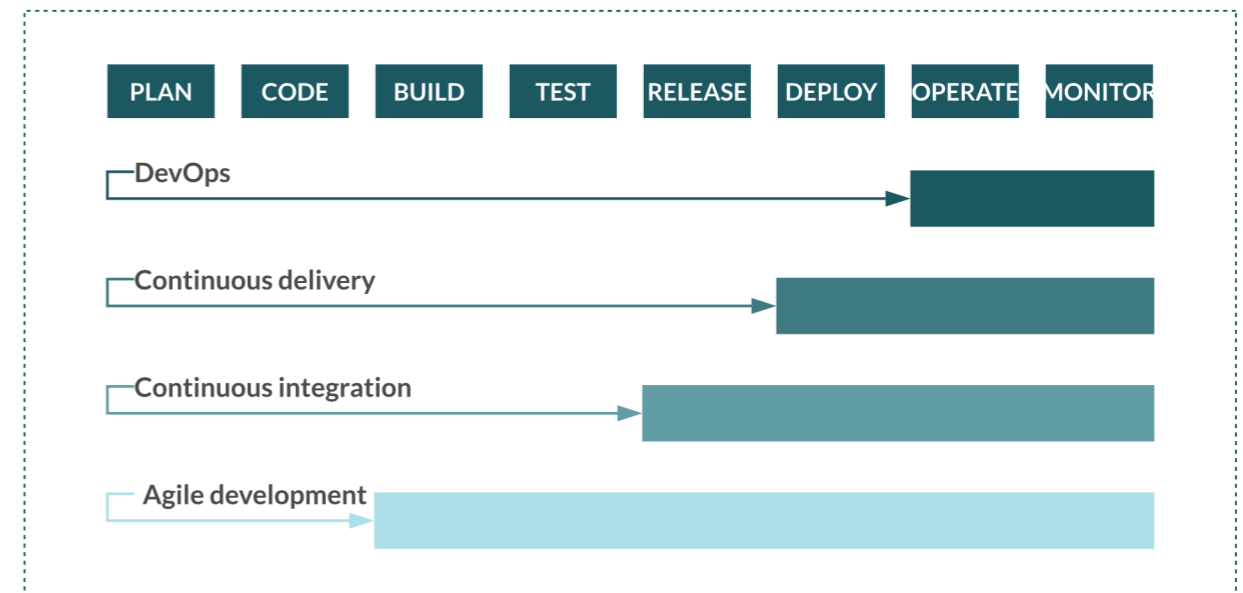
At the same time technological advances such as as infrastructure-as-code [8], containerisation and micro-services [9] require specific skills and knowledge to realise their full potential [9].

A close integration of the operations team with the development team, allowing them to collaborate early in the development cycle, is placed forward as an approach to meet these challenges and reduce friction [11][15]. This approach is commonly referred to as DevOps. Finding a complete and accurate definition of the term DevOps is difficult, for the purpose of this paper we will adopt the definition as provided by Jabbari et al. [10]:

*“DevOps is a development methodology aimed at bridging the gap between Development and Operations, emphasising communication and collaboration, continuous integration, quality assurance and delivery with automated deployment utilising a set of development practices.”*

During a survey of Computer Associates 88% of executives across 1425 organisations indicated to have adopted DevOps or to plan adoption in the next 5 years [12].

Figure 2: Relationship between continuous integration, delivery and deployment according to M. Shahin et al. [5]



DevOps, being a combination of people, process and technology, [3] employs a set of technical practices referred to as continuous deployment (CD). CD practices include a few key processes: version control, continuous integration, test automation and deployment automation [2].

Adoption of these Continuous Deployment practices plays an important role in the software delivery process and may result in increased organizational performance through improved IT performance [2]. An important objective of Continuous Delivery is to reduce the risk of failure in production and gather fast feedback by integrating the various software components early in the cycle to produce deployable results with a high level of certainty [13].

With the evolution described above, where agile iterative approaches are implemented in an increasingly rapidly evolving technological landscape aimed at increasing speed of deployment, the question rises how information security can be integrated.

How can aspects such as regulatory compliance and security assurance on key characteristics such as confidentiality, integrity, availability be considered in this fast paced approach to enable reliable value creation with a managed level of risk? Simply put, one might argue that it does not matter how fast an organisation is capable of developing and deploying software if security and compliance cannot keep up. Eventually the organisation is expected to face security and compliance related challenges or may have to accept an increased level of risk.

*“... It may not not matter how fast we can deploy an application in production if security and compliance cannot keep up ...”*

# Problem statement

Software security assurance aims to provide confidence in the security-related properties and functionalities, as well as the operation and administration procedures, of a given piece of software. Assurance methods are key to achieve confidence that a solution meets its security requirements. Conventional, sequential, security assurances approaches rely on a third party's objectivity and expertise resulting in a side-effect, documentation-focused development, which is in conflict with agile development methods [14]. In this context third parties are defined as security specialists coming "after the show" who are expected to analyse, validate, test and then certify if a more or less finished product meets its security requirements [14]. Research points out that approximately half of the conventional assurance methods and techniques directly clash with the principles and practices of agile development. Most of these techniques create a mismatch due to their reliance on extensive documentation served as a subject of analysis, verification, and validation activities [14].

*"The (conventional) assurance methods, applied naïvely, would create deterring delays between critically short iterations as well as prohibitively inflate the development budgets."*

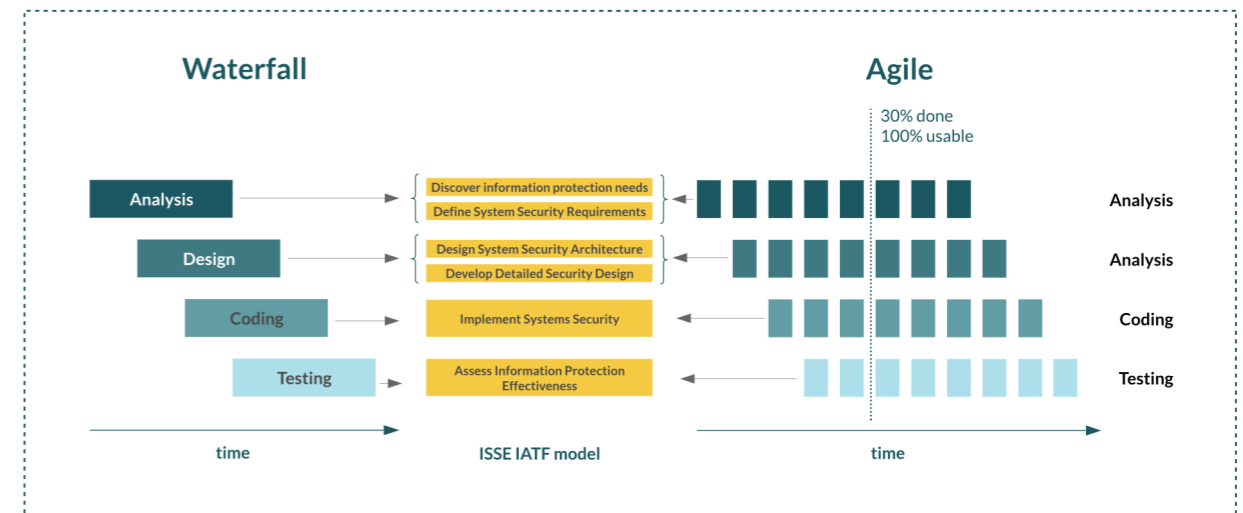
*Beznosov & Kruchten, 2004 [14]*

Many organisations want to apply DevOps, but they are concerned by the security aspects of the produced software [20]. The importance of embedding security assurance in the DevOps process is reflected by the results of a survey on DevOps adoptions and expectations [16] where 28% of the respondents indicate that security and compliance are seen as obstacles to adoption. Increasing regulatory compliance requirements [28] and the potential impact of breaches on the market value [29] and reputation [30] of organisations could be expected to increase these concerns.

In organisations with a strong emphasis on security the security teams can be expected to be important stakeholders in the adoption of DevOps methodologies which requires the integration of security assurance aspects to gain their buy-in [17]. This is not without reason because without adequate involvement of the security team, rapidly deployed software changes are more likely to contain vulnerabilities due to lack of adequate reviews [12].

Where the previous aspects highlight the interest in the integration of security assurance in DevOps from the perspective of preventing negative side effects from occurring there are also some aspirational aspects to the proposed integration. Studies linking DevOps (which is a core principle of DevOps) to organisational performance [18] also point out that the automation of security policy configurations is "mission-critical to reach the highest levels of DevOps evolution" and mentions that the "involvement of security teams in technology design and deployment" should be viewed as an important practice. A recent "State of DevOps report" [61] explicitly mentions that integrating security

Figure 3: Mapping of IATF Information System Security Engineering Process (ISSE) to waterfall and agile process steps



throughout the software delivery lifecycle leads to positive outcomes such as their ability to deploy to production on demand at a significantly higher rate.

There is also an inverse effect where doing DevOps well enables you to do security well [61], the same principles that drive good outcomes for software development such as culture, automation, measurement and sharing are similar to those that drive good security outcomes. Automation is also linked to increased speed of response to security issues [61]. Therefore it can be argued that a strong DevOps culture also supports stronger security.

These aspects are drivers for the tight integration of security into the DevOps, a practice which is commonly referred to as "DevSecOps" or "SecDevOps" [20]. DevSecOps as a concept proposes solutions to improve security on the three dimensions of DevOps namely people, process and technology [21].

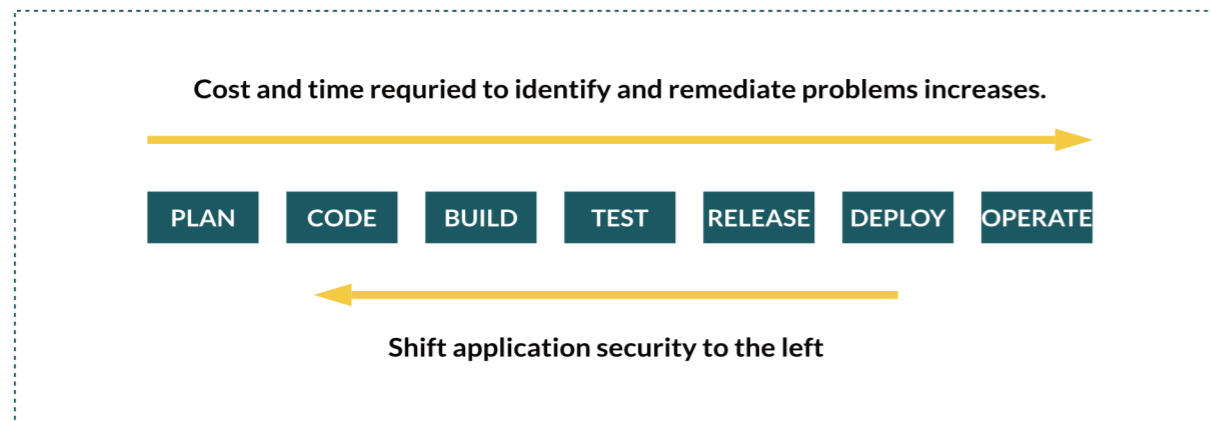
An example of the mindset found in DevSecOps is expressed in the Rugged Software manifesto [22]:

*"Rugged" describes software development organisations that have a culture of rapidly evolving their ability to create available, survivable, defensible, secure, and resilient software.*

A concept frequently cited in the context of DevSecOps is "shift left on security" [23][24][25][26]. It refers to the principle of integrating security related activities as early as possible in the process because the cost and time required to identify and remediate issues as an application advances through the various development stages increases significantly [27]. At its core "shift left on security" advocates delegating authority and coaching of all contributors in the process to integrate security in their work allowing the security team to focus on support in specific complex cases.



Figure 4: Visualisation of the “shift left on security” concept based on the premises of [23] and [27]



The question which rises is how to integrate security assurance activities in DevOps, thereby leveraging its strengths to increase the security posture of the produced software and providing increased speed of response [61], without introducing delays between critically short iterations [14].

To achieve this organisations are in need of a framework of relevant security activities enabling an organisation to integrate security assurance in DevOps with a clear view on the effectiveness as well as the delay caused in the process of continuous delivery.

*Companies could benefit from a framework to effectively and efficiently embed security activities in DevOps.*

## Main research question

The focus of this research is to establish a framework to effectively integrate security assurance activities in DevOps without sacrificing speed and agility. To meet this objective the framework needs to take in to account the factor of delay caused to the development and operations processes. Therefore the main research question is defined as follows:

*Which set off security activities are applicable to DevOps and how do they relate to the perspectives of effectiveness and delay caused in the process of continuous deployment?*

The outcome of this question allows the design of a framework for lean security which integrates security in DevOps without sacrificing speed and agility by providing an overview of the relevant security activities and their characteristics in terms of effectiveness and delay caused to the process of continuous deployment.

## Selection of research methods

The topics of DevOps and DevSecOps are on the vanguard of information technology and therefore are subject to constant and fast evolution. Recent academic papers refer to DevOps as being a novel concept [10] and point out that the number of relevant publications on DevSecOps is low [20]. Therefore the choice was made to perform an exploratory research project with the intention to gain familiarity with the concept of lean security, aim to generate new insights and return these insights to the body of knowledge.

Design science combines behavioural science and design science based on the premise that this combination is most suited to address fundamental problems faced in the productive application of information technology. It also stated that technology and behaviour are inseparable in IS research [31]. This premise aligns with a general consensus in the proposed research domain of this paper, it is believed that both DevOps and Information Security become effective only through the combination of process, technology and people [3][21].

The objective of this master thesis is to develop the specifications of an artefact detailing a set of security activities applicable to DevOps and their characteristics in terms of effectiveness and delay caused in the process of continuous deployment.

This artefact proposes an initial solution to the problem of integrating security in DevOps and as such contributes to the body of knowledge. Therefore, design science methods are selected as the basis for this research. Design science methods are described by Hevner et al. as follows [31]:

*“A research paradigm in which a designer answers questions relevant to human problems via the creation of innovative artefacts, thereby contributing new knowledge to the body of scientific evidence. The designed artefacts are both useful and fundamental in understanding that problem.”*

As described by Recker [32] design science starts from the existing knowledge base to provide the material through which design science research is accomplished thereby achieving rigour. This knowledge may consist of existing theories from science and engineering, specifications of currently known design, useful facts about currently available products, lessons learned from the experience of researchers in earlier design science projects, and plain common sense [33].

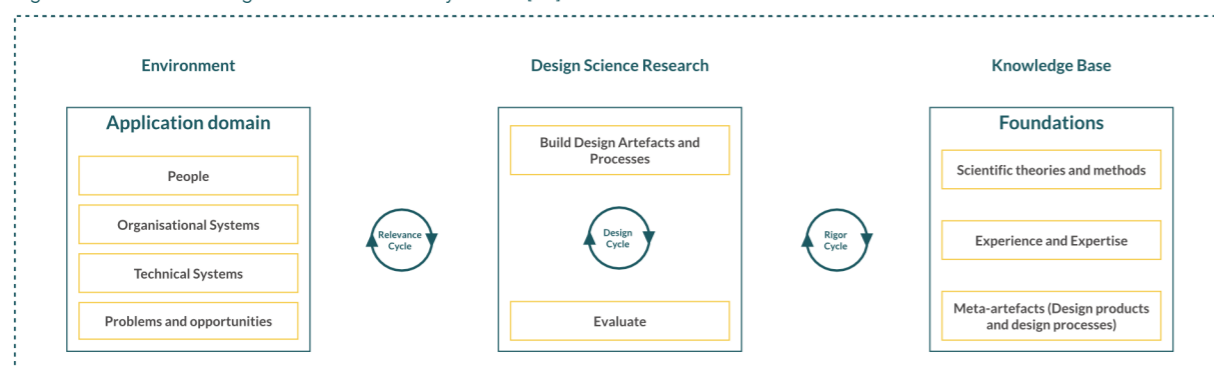
Subsequently the researcher engages in a relevance cycle to bridge the environment of the research project with the design science activities thereby providing relevance in the application domain. At the heart of design science is the design cycle which iterates between the core activity of building and evaluating the design artefacts.

Wieringa indicates that additional research may be required in cases where current scientific research does not provide an answer [33]. The research domain selected for this project is on the vanguard of technological and cultural practices in the field of secure information systems development and operations in agile environments. As a result, knowledge on this subject is constantly evolving and therefore encourages us to employ techniques that facilitate knowledge gathering based on expert experience to generate new insights.

An approach to generating new insights is the use of Group Support Sessions. They facilitate the effective collection, organisation, evaluation, cross-impact analysis and reporting of data [34]. Group Support Sessions have been proposed as a qualitative research method in the decision-making process within the domain of Business Information Security (BIS) due to the stimulation of free-flowing discussion and sharing of experiences eliciting the views of all participants. Previous studies state that GSS provides a solution to the problem of capturing and sharing knowledge and as such can be used to feed decision making [34].

A similar combination of design science research methods and Group Support Sessions for exploratory research have been applied in previous research to derive prioritised lists of items in the field of information security [35].

Figure 5: Overview of design science as defined by Hevner [37]



Based on the need for this research project to elicit the views of practitioners and a similar need to provide a prioritised list of items the choice was made to apply GSS. As a result this research will employ a combination of techniques to interact with experts: (a) surveys, (b) interviews and (c) group support sessions.

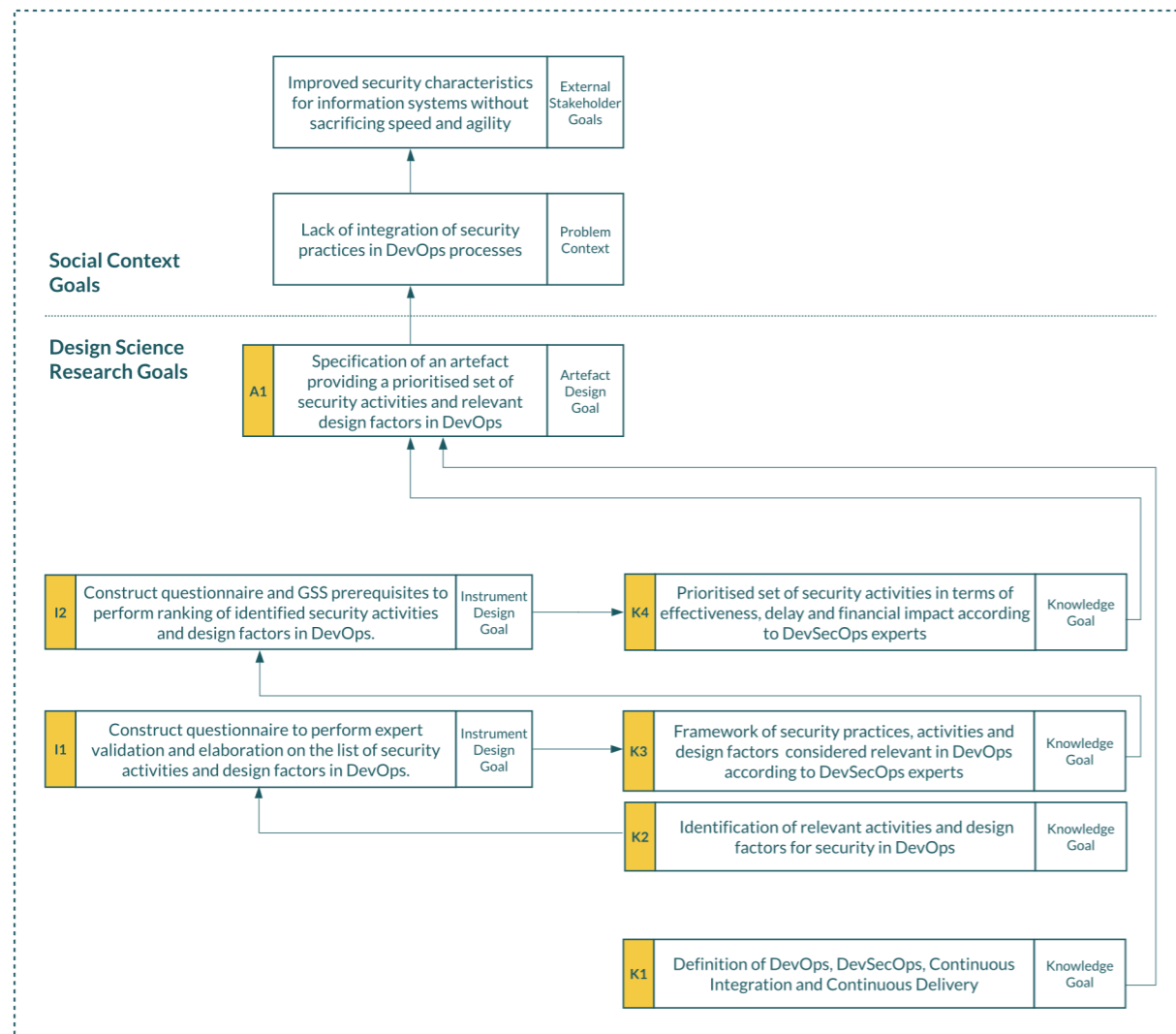
## Design problems and knowledge questions

Wieringa emphasises the need to distinguish design problems from knowledge questions [33]. Design problems call for a change in the real world and are solved through a design which is evaluated by its utility with respect to the stakeholder goals. There is not one single best solution. Knowledge questions on the other hand ask for knowledge about the world as it is. The answers to knowledge questions are evaluated by truth, independent of stakeholder goals. Therefore, Wieringa emphasises the importance to make a clear distinction between knowledge questions and design problems as they need to be treated, and more importantly evaluated, differently. To this end a goal structure for design science research projects is proposed [33].

This proposed structure is applied to the research design for this paper. As stated earlier this research aims to address the need to integrate security activities in DevOps. The objective is therefore to specify an artefact enabling the integration of security activities in DevOps with limited friction (a1).

To this end the following knowledge goals are identified: (k1) definition of the terms DevOps and DevSecOps, (k2) identification of relevant activities and design factors for DevSecOps, (k3) framework of security activities and design factors considered relevant in DevSecOps according to DevSecOps experts and (k4) prioritised framework of security practices, activities and design factors in DevSecOps according to experts. Two instrument design goals are required to facilitate the gathering of information for the knowledge goals: (i1) construct survey to perform expert validation and elaboration on the list identified in K2 and (i2) construct survey and GSS prerequisites to perform ranking of framework established in K3.

Figure 6: Schematic overview of the structure for this research



**Legend**

- Kx** Knowledge goal
- Ix** Instrument design goal
- Ax** Artefact design goal

## Research approach

A clear definition of the terms DevOps and DevSecOps contributes to the reusability and comparability of the results of this study. Therefore the starting point for this research consists of establishing a clear definition for the core concepts referred to in this paper (k1). To achieve this the definitions of DevOps and DevSecOps gathered from literature will be presented to a group of security experts through a survey for validation. The related research questions for this goal are:

**RQ1a: What is the definition of DevOps?**

**RQ1b: What is the definition of DevSecOps?**

A two-step process is applied to gather a list of activities and design factors which are considered relevant for DevSecOps (k2). The first step consists of the identification of security activities applicable in the context of DevOps through literature review of academic literature. During the literature review publications from a period of multiple years will be included to ensure a broad coverage (2015-2020). In a subsequent step the findings of this literature review are listed in a structured way by grouping the security activities that exhibit similar purposes and/or advocate similar notions through the use of thematic analysis as described by Nowell, Norris, White et al. [38]. This activity enables us to answer research question 2:

**RQ2: Which set of security activities and design factors relevant to DevOps processes can be distinguished from academic literature?**

Subsequently the identified list of security activities and design factors will be validated through a survey by a group of DevSecOps experts. This step reduces the risk of inaccuracy introduced by researcher bias and ensures the results of this study incorporate a recent state in the domain of DevSecOps. By relying on the experience of the expert group we achieve a framework of security activities which is relevant to the field (fit for purpose).

The subsequent research question aims to place the identified security activities into context by ranking them based on effectiveness and the friction they introduce. To achieve this a Group Support System (GSS) workshop is organised with security experts to determine which security activities are effective for security in DevOps.

To perform these GSS sessions necessary prerequisites (I2) such as agenda and group composition have been developed. The Group Support Sessions are performed using a sufficiently sized group of information security professionals to ensure reliability and replicability of results. From literature it was observed that there is no 'ideal size' for a focus group however selection the right (number of) experts is key to obtaining collective intelligence. Also the number of items to be discussed is an important variable in the setup of the meeting [34]. The GSS session elicits the view of experts in the field of security and DevOps thereby answering the third research question:

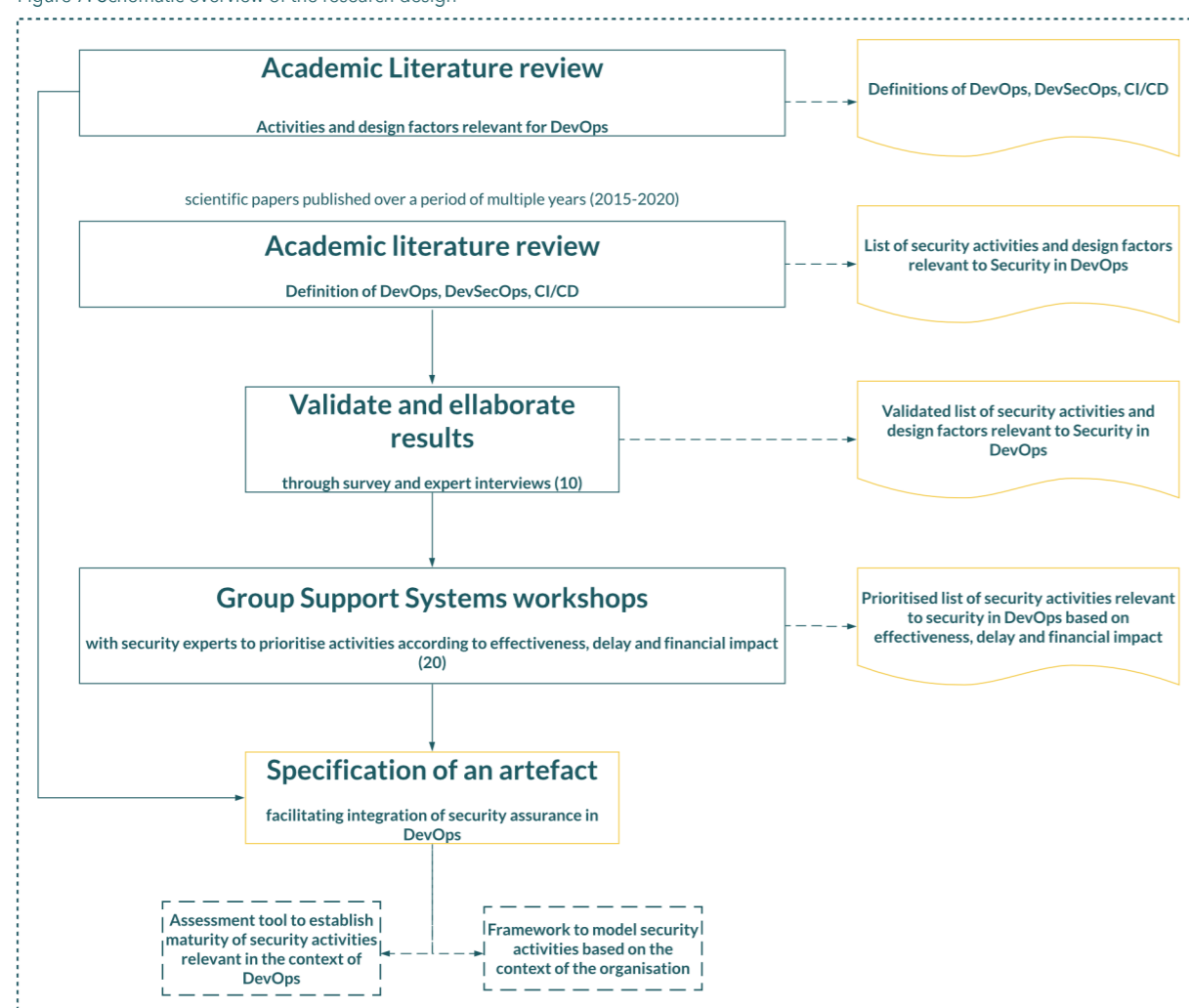
RQ3: How do the identified security activities rank in terms of effectiveness and delay from a practitioner point of view?

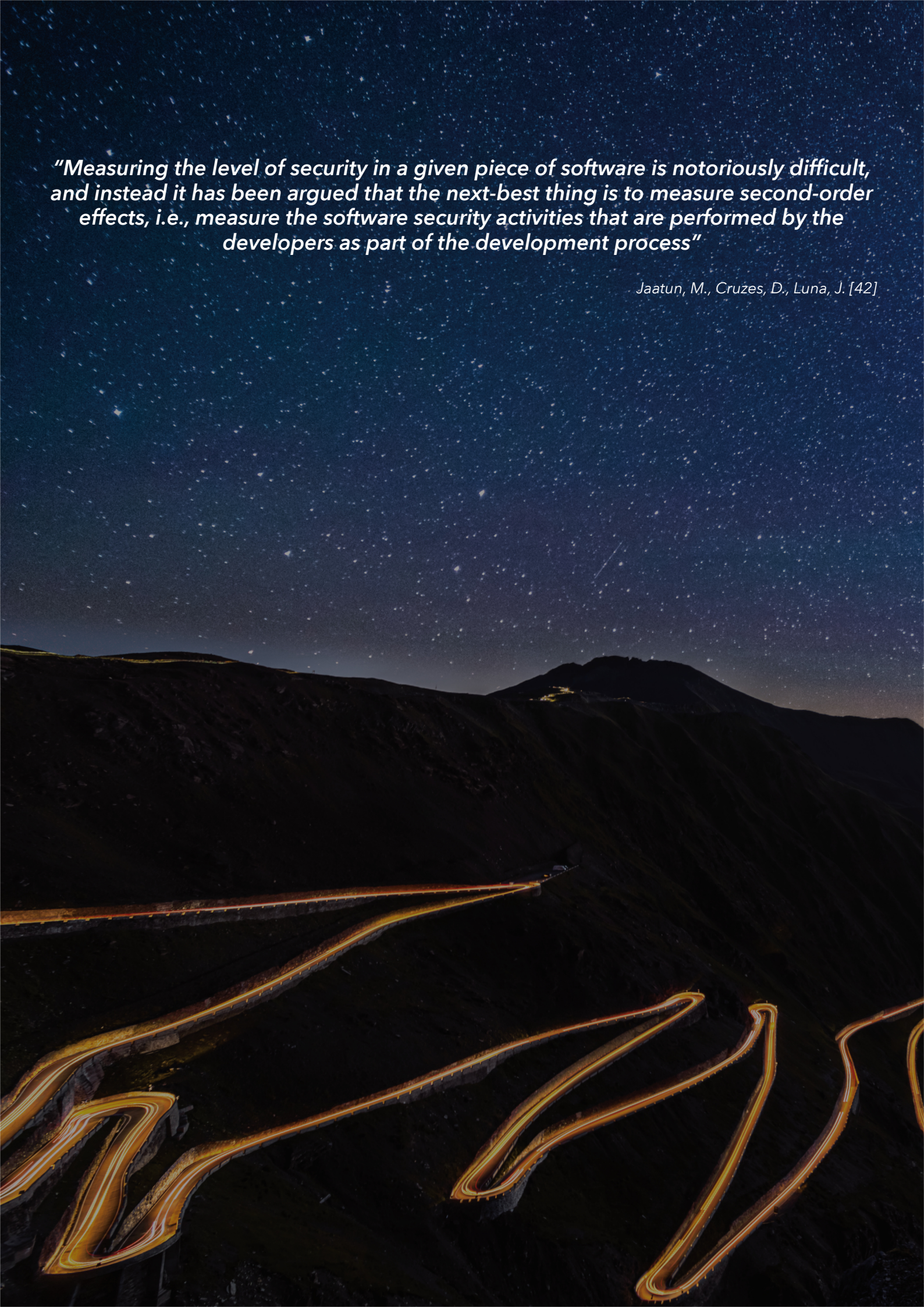
## Research deliverables

The knowledge obtained through this research allows us to design an artefact to enable the integration of security activities in DevOps while minimising friction (a1). This artefact consists of a framework of security activities and design factors relevant for DevOps according to experts which can be used to assess the maturity of, or select the approach approach, to a DevSecOps process.

The final step of the design cycle includes the construction and (iterative) evaluation of the artefact to provide evidence of utility in the stated problem context. This step is considered out of scope of this research due to time constraints and is proposed as a topic for future research.

Figure 7: Schematic overview of the research design





*“Measuring the level of security in a given piece of software is notoriously difficult, and instead it has been argued that the next-best thing is to measure second-order effects, i.e., measure the software security activities that are performed by the developers as part of the development process”*

*Jaatun, M., Cruzes, D., Luna, J. [42]*

---

# Part two: Research

---

This part of the paper contains the outcome of the research as described in part one. It consists of the following sections:

- **Academic literature review:** presents the outcome of the literature review
- **Analysis of the dataset:** presents the results of the thematic analysis applied on the identified gold-set
- **Validation of the research findings:** presents the results of the expert validation & elaboration and the expert prioritisation sessions

## Academic literature review

---

### Searching the knowledge base

---

During this research emphasis is placed on the generation of progressive insights into security activities relevant in the context of DevOps. Therefore this research is exploratory in nature which is reflected in the choice to apply the forward search method for the academic literature review. A forward search attempts to identify relevant papers to the research domain and to follow their citations to discover newer insights.

The preliminary investigation in the scope of producing the research proposal pointed out that finding a clear definition for the terms DevOps and DevSecOps through literature review is challenging (RQ1) [10]. Therefore it was decided to retain the definitions identified during the preparatory phase of this project and to request validation by the DevSecOps experts during the validation and elaboration step. The definitions are:

- **DevOps:** “DevOps is a development methodology aimed at bridging the gap between Development and Operations, emphasising communication and collaboration, continuous integration, quality assurance and delivery with automated deployment utilising a set of development practices.” [10]

- **RuggedOps (DevSecOps):** “Rugged” describes software development organisations that have a culture of rapidly evolving their ability to create available, survivable, defensible, secure, and resilient software [22]

To answer the second research question (RQ2), and achieve knowledge goals K1 and K2, an initial search against the “Web of Science” database is performed based on the keywords ‘DevSecOps’, ‘SecDevOps’ and ‘Security in DevOps’. A query against these keywords in the title, written in English and published between 2015 and 2020 was constructed to identify relevant papers.

The initial search, the results of which can be consulted in table 1, yielded a total of 10 results out of which 9 consisted of academic papers and 1 editorial note. The identified papers are all related to the domain of “Computer Science” and were published in 2016, 2017 and 2019.

While reviewing the artefacts resulting from the initial search, a paper published by Rahman & Williams [12] was identified providing a list of security practices within DevOps. This paper is considered relevant to this research as it relates directly to research question RQ2a. During their study the researchers analysed a set of 60 Internet artefacts and interviewed representatives of nine organisations to explore experiences in utilising security practices.

The results of their research identified 4 security practices and 15 related activities which are relevant in the scope of security in DevOps. The keywords for the initial search were extended to include the keywords used during the research of Rahman et. al. [12] to allow alignment with previous studies. The new extended list of keywords became: ‘DevSecOps’, ‘SecDevOps’, ‘Security in DevOps’, ‘RuggedOps’, ‘SecOps’, ‘Security in Continuous Delivery’, ‘Security in Continuous Deployment’.

A new search using this extended list of keywords using the same criteria against the “Web Of Science” database did not yield any new relevant papers.

The fast-moving nature of this research domain requires us to pay specific attention to include results from recent findings. Therefore a third search iteration is performed using the forward search technique against the previously identified papers to increase coverage and identify more recently published papers. The forward search yielded 8 additional results results, 3 publications from 2019, 4 of 2018 and 1 of 2017. The complete results of the literature review can be consulted in table 1.

Along the duration of this study peers and professors provided additional relevant publications. These papers were also included in the literature review and have been marked as “A” in the literature review search results table.

Figure 8: Schematic overview of the literature research

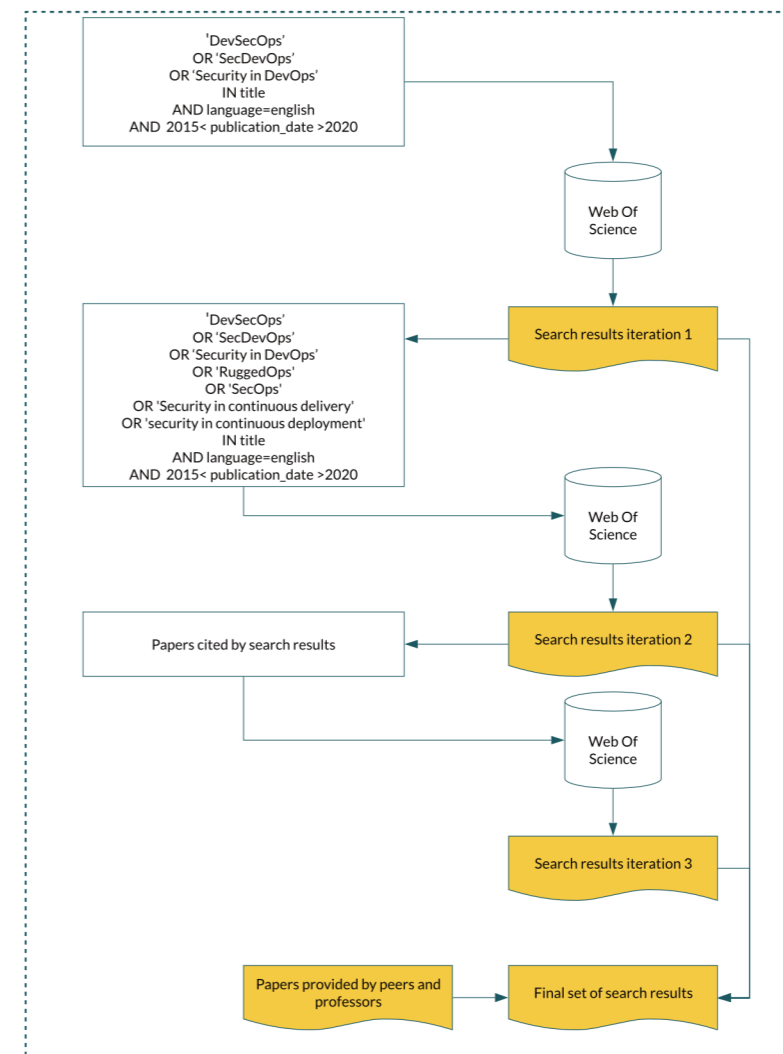


Table 1: Literature review search results

ID	Title	Author(s)	Year	Iteration
P1	Self-Service Cybersecurity Monitoring as Enabler for DevSecOps	Jessica Diaz, Jorge E. Perez, Miguel A. Lopez-Pena, Gabriel A. Mena, Agustin Yague	2019	1
P2	Software Security Activities that Support Incident Management in Secure DevOps	Unknown, Martin Gilje Jaatun	2019	1
P3	Security Assurance in DevOps methodologies and related environments	Grzegorz Siewruk, Wojciech Mazurczyk & Andrzej Karpiński	2019	1
P4	Software Security in DevOps: Synthesizing Practitioners' Perceptions and Practices	Akond Ashfaque Ur Rahman, Laurie Williams	2016	1
P5	DevSecOps: A Multivocal Literature Review	Myrbakken, H; Colomo-Palacios, R	2017	1
P6	SecDevOps: Is It a Marketing Buzzword? Mapping Research on Security in DevOps	Mohan, V; ben Othmane, L	2016	1

Table 1: (Continued.) Literature review search results

ID	Title	Author(s)	Year	Iteration
P7	Security Practices in DevOps	Rahman, AAU; Williams, L	2016	1
P8	Dynamic Security Assurance in Multi-Cloud DevOps	Rios, E; Iturbe, E; Mallouli, W; Rak, M	2017	1
P9	Francois Raynaud on DevSecOps	Carter, K	2017	1
P10	DevOps for Better Software Security in the Cloud	Jaatun, MG; Cruzes, DS; Luna, J	2017	1
P11	A systematic mapping study of infrastructure as code research	Rahman, A; Mandavi-Hezaveh, R; Williams, L	2019	3
P12	Threat analysis of software systems: A systematic literature review	Tuma, K; Calikli, G; Scandariato, R	2018	3
P13	DevOps in practice: A multiple case study of five companies	Lucy Ellen Lwakatare, Terhi Kilamo, Teemu Karvonen, Tanja Sauvola, Ville Heikkilä, Juha Itkonen, Pasi Kuvaja, Tommi Mikkonen, Markku Oivo, Casper Lassenius	2019	3
P14	Service level agreement-based GDPR compliance and security assurance in (multi)Cloud-based systems	Rios, E; Iturbe, E; Larrucea, X; Rak, M; Mallouli, W; Dominiak, J; Munteş, V; Matthews, P; Gonzalez, L	2019	3
P15	A Scaffolding Design Framework for Developing Secure Interoperability Components in Digital Manufacturing Platforms	Fraile, F; Flores, JL; Anaya, V; Saiz, E; Poler, R	2018	3
P16	Scaling Agile Software Development to Large and Globally Distributed Large-scale Organizations	Fraile, F; Flores, JL; Anaya, V; Saiz, E; Poler, R	2018	3
P17	A Multivocal Literature Review on the use of DevOps for e-learning systems	Sanchez-Gordon, M; Colomo-Palacios, R	2018	3
P18	Leveraging Cloud Native Design Patterns for Security-as-a-Service Applications	Torkura, KA; Sukmana, MIH; Cheng, F; Meinel, C	2017	3
P19	A systematic mapping study on security in agile requirements engineering	Curcio, K.; Navarro, T.; Malucelli, A.; Reinehr, S.	2018	A
P20	Security requirements Engineering in the Agile Era: How does it work in Practice?	Daneva, M.; Wang, C.	2018	A
P21	Effectiveness of using card games to teach threat modeling for secure web application development	Thompson, M.; Takabi, H.	2016	A
P22	Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, challenges and Practices	Mojtaba, S.; Muhammad A.B.; Liming Z.	2017	A

## Analysis of the dataset

Data analysis is characterised as the most complex phase of qualitative research and therefore requires a systematic approach which can be transparently communicated to others. Thematic analysis was selected to analyse the dataset because it ensures a transparent approach while still being feasible given the time available to perform this research. Thematic analysis is used to summarise key features of a large data set while forcing researchers to take a well-structured approach to handling data. This helps to produce a clear and organised final report. A risk when applying thematic analysis is that its flexibility can lead to inconsistency and a lack of coherence when developing themes derived from the research data. To improve the trustworthiness of this study the decision was taken to apply the structured approach for thematic analysis as proposed by Lorelli et al. [38].

The procedure is both practical and effective and consists of six steps which may be performed in an iterative fashion: (1) familiarising yourself with your data, (2) generating initial codes, (3) searching for themes, (4) reviewing themes, (5) defining and naming themes and (6) producing the report. In this section a detail is provided on how the different steps of this process are applied to this research.

As a first step the researcher is encouraged to read through the complete data set at least once prior to starting the data analysis. This step aims to ensure the researcher is sufficiently familiar with the information available in the dataset. During this step all identified papers are loaded and categorised in a research tool of choice and read through actively. A short summary of the contents and the rationale for inclusion or exclusion of the paper is written down in appendix A allowing traceability. The papers identified through the fourth iteration were not included in the initial data analysis exercise due to their large number. Instead they were used to verify the completeness of the mapping exercise by reading through them and comparing to the activities and design factors produced through the thematic analysis.

The next step in the approach involved the definition of an initial set of codes based on the previously defined research questions. These codes would then be used to label various sections of the papers allowing a relationship to be identified between the research questions and the information available in the selected set of papers.

The initial set of codes in table 2 was confirmed by reading through the complete data set to ensure it allowed to cover the breadth and depth of the information required for this research. In a second iteration five codes were added to allow capturing and structuring of additional information.

With the codes for thematic analysis defined the next step, where these codes are applied to the contents of the papers in the dataset, was started. Each relationship between a code and a statement was added to a mapping table allowing further analysis. The resulting mapping table can be found in appendix b.

Based on this mapping table the work of reviewing and naming themes starts. As a starting point all excerpts were loaded into a mind mapping software and grouped by code so that all activities,

design factors, etc would be grouped together. Subsequently common themes were identified by grouping the various excerpts into a first level of themes. For some activities the range of activities belonging to a theme was still too broad necessitating a second level grouping. This led to the definition of "level 1" and "level 2" activities where the latter is a more detailed subset of the former.

For instance a theme which quickly became apparent within the activity codes was "manual security testing". All excerpts belonging to this theme were then further grouped based on their content leading to (a) performing manual penetration testing and (b) performing manual security reviews. This approach was applied to all the excerpts mapped to the "activity" and "design factor" codes resulting in the set of activities and design factors displayed in tables 3 and 4. A detailed view on the thematic analysis can be found in appendix c.

## Interim conclusion

The results obtained through this activity, as represented in tables 3 and 4, allow us to complete the second knowledge goal (K2) and as such serves as a starting point to answer the second research question (RQ2): Which set of security activities and design factors relevant to DevOps processes can be distinguished from academic literature?

Table 2: Overview of codes used for thematic analysis

RQ	Code	Description	Iteration
RQ2	Principle	A fundamental truth or proposition that serves as the foundation for DevSecOps	1
RQ2	Practice	A security practice is a collection of activities can be grouped based on existing similarities within activities	1
RQ2	Activity	A security activity focuses on achieving a small, well-defined goal that has a tangible output	1
RQ2	Design factor	A specific way a security activity can be performed to increase its effectiveness or efficiency	1
-	Challenge	An obstacle which may impede or complicate the implementation of a security activity or practice in DevSecOps	2
-	Opportunity	An advantage or improvement related to the implementation of a security activity or practice in DevSecOps	2
-	Methodology	A system of methods applicable to DevSecOps	2
-	Frameworks	A defined and agreed approach that intends to improve security relevant to DevSecOps	2
-	Best practices	A repeated method used by people to improve sec in DevSecOps	2
-	Tool category	A category of security tools which can be used to implement or facilitate a security activity	2

Table 3: Results of thematic analysis of security activities

Activity	Description
Performing continuous feedback from production to development	This activity refers to continuously feeding security metrics and information on security incidents from production back to development.
Provide security training	This activity refers to training a wide range of stake-holders such as developers, architects and product owners on security aspects.
Establish security satellites	This activity refers to creating a network of security savvy people throughout the various teams involved in software development. These people are regularly referred to as security champions.
Practice incident response	This activity refers to practicing incident response through red-team exercises and security drills.
Performing automated security testing	This activity refers to aspects of security testing which can be automated thereby providing actionable information.
Performing automated run-time testing	This activity refers to the dynamic, interactive testing of a deployed application using automated tools (DAST)
Performing automated static testing	This activity refers to code review using automat-ed tools to detect common vulnerability patterns (SAST).
Integrate security tests in unit testing	This activity refers to leveraging unit testing to perform security-oriented tests such as boundary testing.
Performing automated software composition analysis	This activity refers to the verification of all dependencies (e.g. third-party libraries) for known vulnerabilities (SCA)
Implement automated remediation	This activity refers to the dynamic adaptation and reaction to security incidents.
Implement automation of software licensing	This activity refers to ensuring that users are purchasing, installing, and using software as per the conditions set by the software vendor of interest, using automated tools.
Performing security configuration automation	This activity refers to automating security configurations (hardening) throughout the lifecycle of an environment.
Performing security requirements analysis	This activity refers to the definition of security requirements.
Performing threat modeling	This activity refers to performing threat modeling to establish a common model of an application and subsequently identifying potential threats.
Performing risk analysis	This activity refers to analysing the threats to an application in the context of the business impact and likelihood to establish a risk score.
Establishing security SLA's for cloud providers	This activity refers to establishing security service level agreements for cloud providers based on the security requirements of a given application.



Table 3: (Continued.) Results of thematic analysis of security activities

Activity	Description
Performing continuous monitoring	This activity refers to actions enabling a continuous view on various security aspects of development and operations activities.
Performing continuous monitoring of security SLA's	This activity refers to performing continuous monitoring to confirm compliance with security service level agreements for cloud providers.
Performing continuous monitoring of security metrics throughout the SDLC using CI/CD tooling	This activity refers to leveraging the Continuous Integration and Continuous Deployment tools to gather security relevant metrics which can be monitored to identify risks such as coding mistakes or vulnerable dependencies.
Performing continuous monitoring of system metrics using automated tools	This activity refers to continuously monitoring metrics such as resource usage and reaction times. Based on patterns in these metrics malicious activity could potentially be detected.
Performing continuous monitoring of security controls	This activity refers to implementing specific monitoring solutions to determine the effectiveness of security controls for a given application such as SSL settings, access control mechanisms and vulnerable dependencies.
Performing continuous monitoring of application behaviour	This activity refers to continuous monitoring of application behaviour such as input and output to determine changes in patterns which may indicate malicious activity. This activity is commonly implemented through tools such as Web Application Firewalls (WAF).
Provide self-service monitoring capabilities to dev and ops	This activity refers to building monitoring capabilities so that they allow dev and ops to define the collection of metrics, definition of thresholds and alerts themselves making it a shared responsibility.
Performing continuous assurance	This activity refers to continuously validating if the software of interest is compliant with relevant regulatory requirements.
Performing manual security testing	This activity refers to aspects of security testing activities which cannot be automated and need to be performed manually.
Performing manual penetration testing	This activity refers to performing manual penetration tests.
Performing manual security review	This activity refers to performing manual security reviews which is usually a combination of manual code analysis combined with documentation review and stakeholder interviews.
Performing automated security testing of the CI/CD pipeline	This activity refers to performing automated security testing of the CI/CD pipeline to identify weaknesses.

**Legend**

<p><b>Activity</b></p> <p>A security activity focuses on achieving a small, well-defined goal that has a tangible output.</p>	<div style="background-color: #FFD700; width: 20px; height: 20px; margin: 0 auto;"></div> <p><b>L2 Activity</b></p> <p>Detailed security activity which is derived from L1 activities.</p>
---	--

Table 4: Results of thematic analysis of design factors

Activity	Design factor
Performing automated Security Testing	Leverage SecaaS by using cloud provided self-managed, automated and scalable security services
Performing automated Security Testing	Integrate the security tools in an automated deployment pipeline
Performing automated Security Testing	Automate as many security controls and verifications as possible
Perform automated run-time testing	Perform automated run-time testing at four levels: (1) pre-authentication scanning, (2) post-authentication scanning, (3) independent backend scanning and (4) complete workflows
Perform automated run-time testing	Ensure automated run-time testing is implemented for a broad scope of test scenarios
Perform automated run-time testing	Ensure proper unit tests are in place to optimise run-time testing efficiency
Performing automated static testing	Minimise the number of false positives resulting from static testing
Performing security requirements analysis	Treat security requirements as nonfunctional requirements
Performing security requirements analysis	Leverage metrics gathered during the security requirements analysis phase to evaluate the security level of alternative designs
Performing security requirements analysis	Enable the evaluation of alternative designs through suitable metrics during security requirements analysis to determine variations in security levels of a given design and make appropriate choices
Performing security requirements analysis	Leverage goal-oriented requirements analysis (GORE) to perform security requirements analysis
Performing threat modeling	Perform threat modelling from a risk-centric perspective
Performing threat modeling	Perform threat modelling from an attack-centric perspective
Performing threat modeling	Perform threat modelling from a software-centric perspective
Performing threat modeling	Ensure compatibility of threat modelling outcomes from a scope and result perspective
Performing threat modeling	Introduce abuse cases and problem frames to perform threat modeling
Performing threat modeling	Make use of attack or threat trees to perform threat modelling
Performing threat modeling	Implement traceability of threat modelling (results) in the code base
Performing threat modeling	Automated threat impact analysis
Performing risk analysis	Performing risk analysis continuously before each iteration
Performing risk analysis	Performing risk analysis during the design phase
Performing risk analysis	Include a broad range of stakeholders including the business owner when setting security goals

Table 4: (Continued.) Results of thematic analysis of design factors

Activity	Design factor
Performing risk analysis	Establish clear rules regarding information exchange across teams and maintain a log for every access to sensitive data
Performing risk analysis	Provide security knowledge and tools and encourage the Development and Operations teams to integrate themselves
Performing risk analysis	Consider gamification for finding vulnerabilities or bugs
Performing continuous monitoring	Ensure continuous monitoring covers a wide range of resources and metrics including logical security, availability and intrusions
Performing continuous monitoring	Leverage monitoring as code to establish a versioned and repeatable deployment of monitoring infrastructure
Performing manual security testing	Limit manual penetration testing to critical components or perform in parallel to reduce impact on deployment lead times
Performing continuous feedback from production to development	Set a time limit for all lower-priority security defects
Performing continuous feedback from production to development	Give attack patterns to your developers
Performing continuous feedback from production to development	Build an internal forum to discuss attacks
Performing continuous feedback from production to development	Establish emergency code base response
Performing continuous feedback from production to development	Incorporate security tests as part of QA for detected incidents
Providing security training	Teach every developer enough to enable them to identify areas where they would benefit from the advice of an expert

# Validation and prioritisation of the research findings

## Preparations

This research requires capturing the current state-of-the-art approaches in the fast evolving domain of DevSecOps. As such this research cannot remain limited to literature review which mainly provides retrospective insights and often fails to capture new perspectives. To close this gap the choice was made to include a panel of DevSecOps experts. The insights of DevSecOps experts are leveraged in two different phases.

In a first phase a group of experts is consulted to validate and elaborate the findings of the literature research. This allows capturing state-of-the-art knowledge and insights while also reducing researcher bias thereby contributing in answering research question 2 (RQ2). This is achieved through a survey and optional structured interview of these experts. The protocol used for this survey and the results are detailed here.

In a second phase a group of experts has been requested to perform a ranking of the validated activities in terms of effectiveness, financial impact and delay thereby contributing to answering research question 3 (RQ3). The ranking-type Delphi method is applied to develop group consensus about the relative importance in terms of these three aspects for the security practices which were refined in the previous step.

The Delphi method is described by Linstone and Turoff [39] as follows:

*Delphi may be characterised as a method for structuring a group communication process so that the process is effective in allowing a group of individuals, as a whole, to deal with a complex problem. To accomplish this “structured communication” is provided: some feedback of individual contributions of information and knowledge; some assessment of the group judgment or view; some opportunity for individuals to revise views; and some degree of anonymity for the individual responses.*

Due care has been applied to safeguard the validity of the results increase the confidence in this study and improve reusability of the results. Validity as defined by Saunders, Lewis and Thornhill [40] refers to (1) the appropriateness fo the measure used, (2) the accuracy of the analysis of the results and (3) generalisability of the findings.

When looking specifically at the aspect of measurement validity it is important to ensure the use of appropriate measures to gather information so that the data obtained gives an accurate reflection of the actual situation in practice. The exploratory nature of this research combined with the broad subject domain makes it a complex issue requiring the knowledge from people who understand

different aspects of security [40]. Therefore it is important to leverage group knowledge, a group of experts will be able to provide more appropriate answers compared to individual expert's opinions.

Okoli and Pawlowski point out the importance of selecting appropriate experts as a very important aspect of the measurement validity of the Delphi method [41]. Therefore their guidelines were closely followed while preparing the solicitation of qualified experts for both phases of this research.

Table 5 provides an overview of the criteria which were established to identify suitable experts. The skills requirements are defined broadly to ensure inclusion of experts in various roles and across a wide range of organisations. This is important for research in such a broad domain due to the wide range of stakeholders and disciplines involved. As such the background knowledge and the role of the experts in their respective organisations is expected to play an important role, for instance people who come from a development background may have a different view compared to someone from networking or operations teams. Likewise, someone being part of a security architecture team will have a different view compared to application security engineers.

Table 5: Selection criteria for DevSecOps experts

Category	Criteria
	<p>Academic:</p> <ul style="list-style-type: none"> <li>Published article on the subject matter of agile security, DevSecOps or CI/CD in peer reviewed academic magazines; OR</li> <li>Master or PhD thesis on the subject matter of agile security, DevSecOps or CI/CD</li> </ul>
Disciplines and skills	<p>Practitioner:</p> <ul style="list-style-type: none"> <li>min 5 years of professional experience; AND</li> <li>Holding a relevant certification in information security from certifying bodies such as SANS, ISC2 or ISACA; AND</li> <li>Technical experience in a DevSecOps environment and have an affiliation with security; OR</li> <li>Organizational experience structuring people, process or tool aspects of a DevSecOps environment.</li> </ul>
Target Organisations	<ul style="list-style-type: none"> <li>Large sized organisations (+1000 employees) in a highly or moderately regulated sector;</li> <li>Medium sized organisations (+250 employees) in a highly or moderately regulated sector;</li> <li>Digital native organisations;</li> <li>Government institutions;</li> <li>Organisations providing information security services</li> </ul>

When looking at the target organisations some differentiators may be hypothesised to result in different views on Dev(Sec)Ops. These differentiators could be the size, the variation in the level of regulatory requirements and the amount of legacy technology in the organisation. Therefore a wide range of potential organisations was included in the selection criteria.

The knowledge nomination worksheet (KRNW), as found in appendix e, was iteratively populated with the names of potential experts for participation in this study. In a first iteration potential experts from personal networks of the researchers and promoters were leveraged to establish an initial set. Gradually this list was extended by experts proposed by people who were contact for participation in this survey. Potential academic experts were selected from the list of authors whose papers were identified during the literature review phase of this research.

All experts on the KRNW were contacted through a formal invitation to request their collaboration for the first and second phase. The invitations and responses were also included in the KRNW (appendix e).

## Validation and elaboration of findings by expert panel

A total of ten DevSecOps experts responded positively to the invitation for the validation and elaboration survey. The survey consisted of the following sections:

- Contextual questions: the respondent was asked to answer questions related to his/her role and the type of organisation in which he/she is active
- Definitions: the respondent was asked to indicate whether or not he/she agreed to the definition of "DevOps" and "RuggedOps" as found in academic literature
- Activities and design factors: the respondent was asked to indicate for each security activity and design factor, identified previously (see tables 3 and 4), whether they perceived it as relevant in the context of DevSecOps. The choices were binary, either an activity or design factor was considered relevant or not.

The survey was constructed dynamically to limit the strain on the respondents. First the respondents were asked if they believed a security activity (level 1 activity) was perceived as relevant or not. The related design factors and more detailed activities (level 2 activity) were only displayed if the level 1 activity was deemed relevant. Otherwise the questions were filtered out. For each activity and design factor the expert was requested to provide comments and asked to indicate any missing activities or design factors at the end of each section.

The results of the contextual questions (see table 6) confirmed that the selection of experts was performed adequately and that the members could be expected to have the required knowledge to perform the validation and elaboration of the findings. All experts were active in an organisation which performs in-house development of business strategic applications and nearly all of them have a DevSecOps team established and are performing continuous deployments.

When asked about their agreement on the terms DevOps and DevSecOps identified earlier the experts displayed agreement on the term DevOps and DevSecOps with 70% of them agreeing to the provided definition. The respondents who did not agree with the definition provided a comment to explain their perspective (table 7). With this our research question RQ1 and related knowledge goal K1 regarding the definition of DevOps and DevSecOps can be considered answered.

Table 6: Results of contextual questions

Contextual question	4	7	8	9	10	11	12	13	14	16	%
In which type of organisation are you active?	P	G	P	P	P	P	P	P	P	P	
Is your organisation performing continuous delivery?	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	Yes	80%
Is your organisation performing continuous deployments?	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	80%
Does your organisation have a cloud adoption strategy?	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	90%
Is a Dev(Sec)Ops team established within your organisation?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	90%
Is your organisation performing in-house development of business strategic applications?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	100%
Is your organisation performing outsourced development of business strategic applications?	No	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes	70%
Does your organisation have clearly established architectural guidelines in the area of DevSecOps or CI/CD?	Yes	No	Yes	Yes	No	No	No	Yes	No	Yes	50%
Does your organisation have clearly established security guidelines in the area of DevSecOps or CI/CD?	Yes	No	Yes	Yes	No	No	Yes	Yes	Yes	Yes	70%

P = Private, G = Governmental

The survey contained a total of 28 security activities and 35 design factors for which the respondents were requested to indicate if they perceived the activity or design factor as relevant in the context of DevSecOps. A technical issue with the survey resulted in 6 design factors (related to risk analysis) not being displayed to the respondents which were therefore not validated. The results of the survey can be found in tables 7 and 8.

Table 7: Results on the validation of the proposed definitions for DevOps and DevSecOps

Definition	4	7	8	9	10	11	12	13	14	16	%
Do you agree with the provided definition for DevOps?	Yes	No <sup>1</sup>	Yes	Yes	No <sup>2</sup>	No <sup>3</sup>	Yes	Yes	Yes	Yes	70%
Do you agree with the provided definition for DevSecOps?	Yes	No <sup>4</sup>	Yes	No <sup>5</sup>	Yes	Yes	Yes	Yes	No <sup>6</sup>	Yes	70%

<sup>1</sup> Saying that DevOps is a "development methodology" is selling it short. I would call it more a system/software development lifecycle methodology

<sup>2</sup> Should be aimed at business goals and their needs for agility. It should not be aimed at development or operations itself

<sup>3</sup> The last part about automation is what developer brought to it. DevOps is about braking the wall between dev and ops, initially not about tools and automation

<sup>4</sup> I do not see why it has to rapidly evolve

<sup>5</sup> Depends on the teams and the department in which they operate

<sup>6</sup> Only if the deliverable is software. Security is a systemic property that is much wider than software alone

Overall we see a strong consensus regarding the relevance of the security activities which were identified through literature review with the exception of "Implement automation of software licensing", which receiving only a score of 30% can be considered not relevant to DevSecOps. All

design factors were rated as relevant by at least 50% of the respondents and most by over 70% of the respondents. The experts also provided additional insights through comments on the activities and design factors. These comments were analysed and grouped together leading to the creation of 3 new security activities (level 1), 4 new detailed security activities (level 2) and 53 new design factors. Details on the grouping and definition of these supplemental activities and design factors can be consulted in appendix f.

The results from the validation and elaboration step were provided to the experts for their information and feedback however a formal validation step by reissuing the updated survey was not performed due to time limitations and risk for potential "survey fatigue" among the expert panel.

During the survey experts were requested to participate in an optional interview to further discuss their views on the topic of DevSecOps. A total of 5 experts indicated their willingness to participate out of which 3 responded to the invitation. The insights gathered during these interviews were used to further refine the results obtained from the survey. The transcripts of these interviews can be found in appendix d.

The final list of security activities and related design factors was updated and restructured based on the outcome of the survey and the interviews. This validated and elaborated list of security activities and design factors provides the answer to the second research question (RQ2) and related knowledge goal (K3) and forms the basis for the prioritisation exercise performed in the next step of this research. The knowledge and insights gained from the experts is represented in the in-depth descriptions of the various security activities included in the first part of this paper.

## Interim conclusion

The results obtained through the expert survey on the validation of the terms DevOps and DevSecOps (see table 8) allows us to answer our first research question (RQ1) and achieve our first knowledge goal (K1) as follows:

### RQ1a: What is the definition of DevOps?

"DevOps is a development methodology aimed at bridging the gap between Development and Operations, emphasising communication and collaboration, continuous integration, quality assurance and delivery with automated deployment utilising a set of development practices." [10]

### RQ1b: What is the definition of DevSecOps?

RuggedOps (DevSecOps): "Rugged" describes software development organisations that have a culture of rapidly evolving their ability to create available, survivable, defensible, secure, and resilient software [22]

The results obtained through the expert validation and elaboration as represented in tables 9 and 10 allow us to complete the second knowledge goal (K2) and allows us to answer the second research question (RQ2):

**RQ2: Which set of security activities and design factors relevant to DevOps processes can be distinguished from academic literature?**

A total of 31 security activities (table 8) and 34 design factors (table 9) identified through literature research are considered relevant in the context of security for DevOps. The elaborated set based on the expert panels insights totals 33 security activities and 87 design factors.

Table 8: Results of validation of security activities by expert panel

Activity	4	7	8	9	10	11	12	13	14	16	Relevant
Performing automated security testing	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	100%
Performing automated run-time testing	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	80%
Performing automated static testing	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	100%
Performing automated software composition analysis	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	90%
Integrate security tests in unit testing	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	90%
Performing security requirements analysis	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	100%
Performing threat modeling	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	100%
Performing risk analysis	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	80%
Establishing security SLA's for cloud providers	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	80%
Performing continuous monitoring	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	90%
Performing continuous monitoring of security SLA's	Yes	No	Yes	No		No	Yes	No	Yes	Yes	56%
Performing continuous monitoring of security metrics throughout the SDLC using CI/CD tooling	Yes	Yes	Yes	Yes		No	Yes	Yes	Yes	No	78%
Performing continuous monitoring of system metrics using automated tools	No	Yes	Yes	Yes		No	Yes	Yes	No	No	56%
Performing continuous monitoring of security controls	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes	No	89%
Performing continuous monitoring of application behaviour	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes	No	89%
Provide self-service monitoring capabilities to dev and ops	Yes	No	Yes	Yes		Yes	Yes	Yes	No	No	67%
Implement automated remediation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	100%
Implement automation of software licensing	No	No	No	Yes	Yes	Yes	No	No	No	No	30%
Performing continuous assurance	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	80%
Performing manual security testing	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	90%
Performing manual penetration testing	Yes		Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	89%
Performing manual security review	Yes		Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	89%

Table 8: (Continued.) Results of validation of security activities by expert panel

Performing manual security testing	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	90%
Performing manual penetration testing	Yes		Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	89%
Performing manual security review	Yes		Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	89%
Performing automated security testing of the CI/CD pipeline	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	100%
Performing security configuration automation	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	No	70%
Performing continuous feedback from production to development	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	80%
Provide security training	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	80%
Establish security satellites	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	100%
Practice incident response	Yes	No	Yes	No	Yes	Yes	Yes	Yes	No	No	60%

Table 9: Results of validation of design factors by expert panel

Design factor	4	7	8	9	10	11	12	13	14	16	Relevant
Leverage SecaaS by using cloud provided self-managed, automated and scalable security services	Yes	Yes	Yes	Yes	Yes	No		Yes	Yes		88%
Integrate the security tools in an automated deployment pipeline	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	100%
Automate as many security controls and verifications as possible	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	100%
Perform automated run-time testing at four levels: (1) pre-authentication scanning, (2) post-authentication scanning, (3) independent backend scanning and (4) complete workflows	Yes	Yes	Yes	Yes		Yes	Yes	Yes		Yes	100%
Ensure automated run-time testing is implemented for a broad scope of test scenarios	Yes	Yes	Yes	Yes		Yes	Yes	Yes		Yes	100%
Ensure proper unit tests are in place to optimise run-time testing efficiency	Yes	Yes	Yes	Yes		Yes	Yes	No		Yes	88%
Minimise the number of false positives resulting from static testing	Yes	Yes	Yes	Yes		Yes	Yes	Yes		Yes	100%
Treat security requirements as nonfunctional requirements	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes		100%
Leverage metrics gathered during the security requirements analysis phase to evaluate the security level of alternative designs	Yes			No	Yes	No	Yes	Yes	Yes		71%
Enable the evaluation of alternative designs through suitable metrics during security requirements analysis to determine variations in security levels of a given design and make appropriate choices	Yes			Yes	Yes	Yes	Yes				100%
Leverage goal-oriented requirements analysis (GORE) to perform security requirements analysis	Yes			Yes	Yes	Yes	Yes				100%
Perform threat modelling from a risk-centric perspective	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	100%
Perform threat modelling from a attack-centric perspective	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	100%
Perform threat modelling from a software-centric perspective	Yes		Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	89%
Ensure compatibility of threat modelling outcomes from a scope and result perspective	Yes			Yes	Yes	Yes		Yes		Yes	100%
Introduce abuse cases and problem frames to perform threat modeling	Yes		Yes		Yes	Yes	Yes	Yes	Yes	Yes	100%
Make use of attack or threat trees to perform threat modelling	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	100%

Table 9: (Continued.) Results of validation of design factors by expert panel

Design factor	4	7	8	9	10	11	12	13	14	16	Relevance
Implement traceability of threat modelling (results) in the code base	Yes	Yes		Yes	Yes		Yes			Yes	100%
Automated threat impact analysis	Yes	Yes	Yes	Yes	Yes			Yes		Yes	100%
Performing risk analysis continuously before each iteration											
Performing risk analysis during the design phase											
Include a broad range of stakeholders including the business owner when setting security goals											
Establish clear rules regarding information exchange across teams and maintain a log for every access to sensitive data											
Provide security knowledge and tools and encourage the Development and Operations teams to integrate themselves											
Consider gamification for finding vulnerabilities or bugs											
Ensure continuous monitoring covers a wide range of resources and metrics including logical security, availability and intrusions	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes	Yes	100%
Leverage monitoring as code to establish a versioned and repeatable deployment of monitoring infrastructure	Yes	Yes	Yes	Yes		Yes	Yes	Yes		Yes	100%
Limit manual penetration testing to critical components or perform in parallel to reduce impact on deployment lead times	No		Yes	Yes	Yes	No	Yes	Yes	No		63%
Set a time limit for all lower-priority security defects	Yes			Yes	Yes	Yes	No	Yes	No		71%
Give attack patterns to your developers	Yes			Yes	Yes	Yes	Yes	Yes	Yes		100%
Build an internal forum to discuss attacks				Yes	Yes	Yes	No	Yes	Yes		83%
Establish emergency code base response	Yes			Yes	Yes	Yes	No	Yes			83%
Incorporate security tests as part of QA for detected incidents	Yes			Yes	Yes	Yes	No	Yes			83%
Teach every developer enough to enable them to identify areas where they would benefit from the advice of an expert	Yes			Yes	Yes	Yes	Yes	Yes	Yes	Yes	100%

## Prioritisation by expert panel

With the validated and elaborated list of security activities established the remaining objective of this research is to prioritise them on aspects which are relevant to DevSecOps. To achieve this a second expert panel was established to leverage the knowledge of security experts to rank them in terms of effectiveness, delay and financial impact. To facilitate this a Group Support Session (GSS) was organised bringing together the experts while stimulating free-flowing discussions and sharing of experiences. The session was performed on 15<sup>th</sup> of April 2020 from 16:00 to 18:00 GMT+1 through an online video conferencing platform and leveraging group support systems software (MeetingWizard). The target group size for the GSS session was set at 10 participants to ensure an appropriate group size allowing sufficient qualified data without introducing a high level of noise.

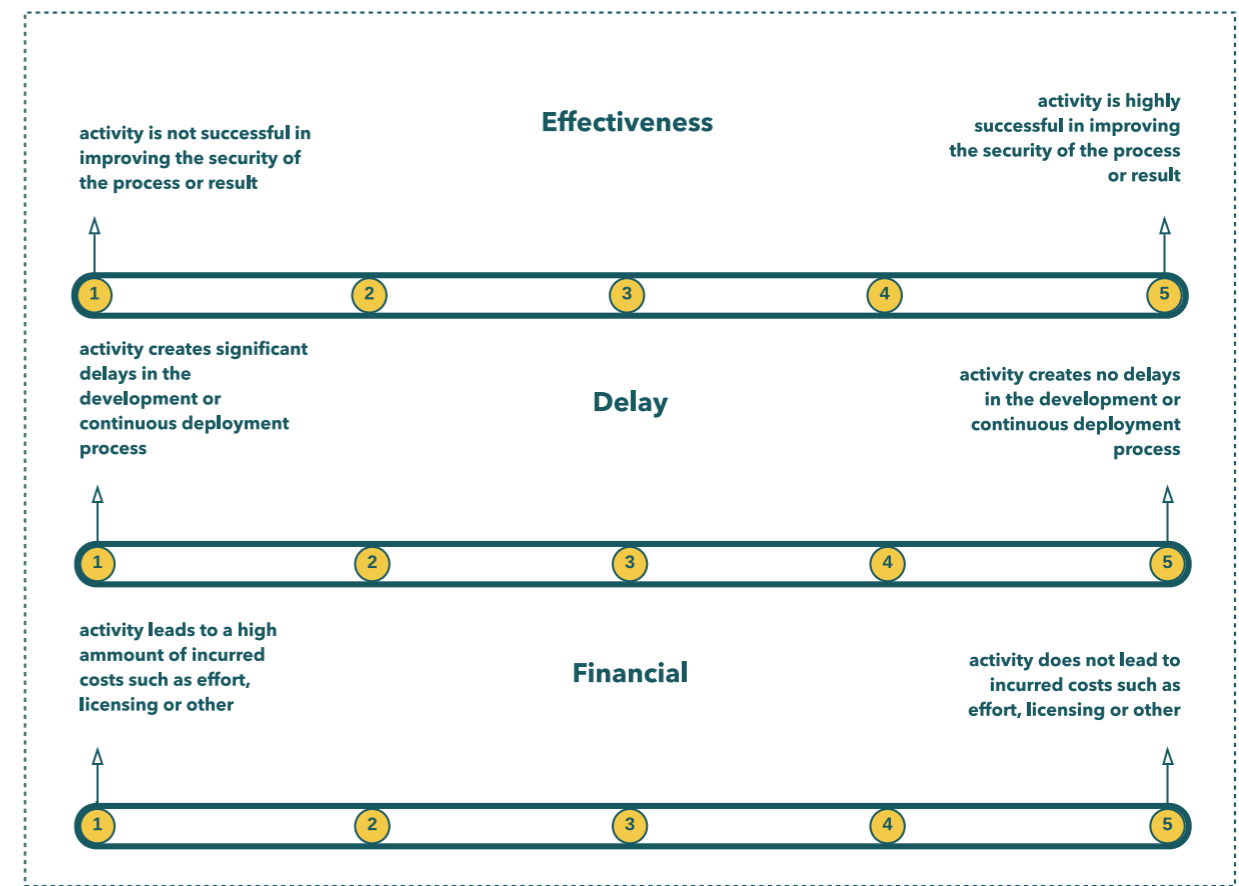
Figure 9: Screenshot of GSS session

The number of items to be discussed was limited to the 29 security activities to make the best out of the limited time and “processing power” of the group [34].

A total of 8 experts participated in the ranking session (see appendix e: knowledge nomination worksheet). These experts received an invitation beforehand together with an overview of the outcome of this research so far allowing time to prepare should they wish to do so.

The security activities were grouped by category (collaboration, non-automated and automated security activities) and treated in these groups to avoid long uninterrupted ranking steps. The experts were explicitly asked not to perform the ranking by category but to perform the ranking holistically. The experts were also invited to provide comments to express their thoughts and reflections, which they did extensively.

Figure 10: Ranking scales used for prioritisation of security activities



Each security activity was ranked on a likert scale from 1 to 5 for each of the following three aspects:

- **effectiveness:** the degree to which an activity contributes to the security of the resulting software
- **delay:** the degree to which an activity is expected to cause delays in the development and operations processes
- **financial:** the degree to which an activity is expected to have financial impact such as effort or licensing costs

This scale was explained to the experts by providing two examples on how the ranking could be applied to “Performing penetration testing” and “Performing continuous assurance”. It was clearly expressed that these were only meant as illustration of the ranking scales. From this point on the session was strictly moderated by prof. Dr. Yuri Bobbert to avoid any influence from the researcher. The participants had the opportunity to ask questions when the context of a certain activity was unclear. The raw result set was downloadable for the experts upon completion of the session.

Descriptive statistics were applied to the dataset to determine the qualitative scores for each of the three aspects for which a consensus exists between most of the experts. This was achieved by calculating the median and Inter-Quartile Range (IQR) for each item. The median is a measure of the central tendency and the IQR provides a measure of the spread. A full SPSS analysis was not

performed on the dataset due to the limited time available. An overview of the results can be consulted in table 10 and the scoring of the participants and comments can be consulted in appendix g.

The comments provided by the participants were read and analysed to place the scoring in context. In most cases where the scoring displayed a large spread (IQR), indicating dissonance of opinion, the comments provided sufficient insight to explain the differences. A write-up of the conclusions for each activity was drafted and provided to the experts for feedback. This conclusion is included in part one of this paper under the detailed security activities section.

## Interim conclusion

The results of this analysis (see table 10) were used to construct the prioritised list of security activities relevant in DevSecOps thereby providing an answer to the third and last research question RQ3 and the related knowledge goal K3.

Table 10: Results of the prioritisation of security activities by the expert panel

Activity	Effectiveness		Delay		Financial	
	Mean	IQR	Mean	IQR	Mean	IQR
<b>Collaboration</b>						
Performing continuous feedback from production to development	4.5	1.25	4.0	1.50	4.0	1.00
Provide security training	4.0	1.00	4.0	1.50	3.0	0.50
Establish security satellites	5.0	1.00	4.0	0.50	3.0	1.50
Practice incident response	4.0	1.25	4.0	1.50	3.0	1.50
Establish a security mindset across the organisation	4.0	1.25	4.0	1.50	3.0	2.00
<b>Use of automated activities</b>						
Performing automated security testing						
Performing automated run-time testing	3.0	1.25	4.0	0.50	3.0	1.50
Performing automated static testing	3.5	2.00	3.0	1.50	2.5	1.00
Integrate security tests in unit testing	4.5	1.25	4.0	0.50	3.0	1.50

Table 10: (Continued.) Results of the prioritisation of security activities by the expert panel

Activity	Effectiveness		Delay		Financial	
	Mean	IQR	Mean	IQR	Mean	IQR
<b>Use of automated activities (continued)</b>						
Performing continuous monitoring						
Performing continuous monitoring of security SLA's	3.0	2.00	4.0	2.00	3.5	1.75
Performing continuous monitoring of security metrics throughout the SDLC using CI/CD tooling	4.0	1.00	4.0	1.50	3.5	1.00
Performing continuous monitoring of system metrics using automated tools	4.0	1.50	4.0	1.50	3.0	0.75
Performing continuous monitoring of security controls	4.0	0.50	4.0	1.00	3.0	1.50
Performing continuous monitoring of application behaviour	4.0	0.25	4.0	1.50	2.0	1.50
Provide self-service monitoring capabilities to dev and ops	3.0	1.25	4.0	1.50	4.0	1.50
Implement centralised dashboard	3.5	1.50	4.0	1.50	4.0	0.00
Implement automated remediation	3.0	1.50	5.0	0.50	3.0	1.50
Performing security configuration automation	4.0	0.25	4.0	1.50	3.0	2.00
Implement secrets management	4.5	2.00	4.0	1.50	3.5	1.75
Manage digital supply chain						
Establish artefact and source code registries which are automatically scanned for vulnerabilities	4.0	1.00	3.0	1.50	4.0	1.50
Implement automated container security scanning	4.0	1.25	4.0	0.00	3.5	1.00
Performing automated software composition analysis	3.0	1.25	3.0	1.00	3.0	1.50
<b>Use of non-automated activities</b>						
Performing security requirements analysis	3.0	0.50	3.0	1.50	3.5	2.00
Performing threat modeling	3.0	0.50	3.0	1.00	3.0	0.25
Performing risk analysis	4.0	0.50	3.5	1.00	4.0	0.25
Establishing security SLA's for cloud providers	3.0	2.00	4.5	2.25	3.5	1.00
Performing continuous assurance	4.0	2.00	3.0	2.00	3.0	0.25
Performing manual security testing						
Performing manual penetration testing	4.0	1.50	2.0	1.50	2.0	1.25
Performing manual security review	3.0	1.50	2.0	1.00	3.0	1.25
Securing the CI/CD pipeline	4.0	1.50	4.0	1.25	3.5	1.25




---

# Part three: Analysis

---

This part of the paper contains in-depth analysis performed for each of the security activities and design factors identified during the previous research steps.



*"Integrating security deeply into the software delivery lifecycle makes teams more than twice as confident of their security posture."*

*State of DevOps report, 2019 [61]*

# Collaboration

## Performing continuous feedback from production to development

Performing continuous feedback between production and development teams is mentioned by various researchers as an important factor for effective DevSecOps. A wide range of activities are present under this umbrella however the main concept revolves around the ability to make quick changes in the code if a vulnerability is discovered (in operations), this is also referred to as emergency code base response [42]. Establishing emergency code base response allows organisations to react to security incidents through swift modification and deployment of applications. Continuous feedback has an added bonus of facilitating continuous improvement by allowing security incidents to be leveraged as a learning opportunity.

### Design factors

A corner stone to enable emergency code base response is a process to feed flaws or bugs that are discovered in operations to the development team. The number of flaws or bugs discovered in production can be leveraged as a metric to measure the level of security in a given piece of software. To facilitate this exchange of information an interface between developers and incident response operators needs to be established. A suggested approach is to create an internal forum to discuss attacks and exchange attack patterns [43]. Another recommended practice would be to assign a contact point for the software security group outside of office hours.

Feeding back information regarding security defects and even incidents is only one part of the approach. As mentioned by researchers [42] and confirmed by various members of the DevSecOps expert panel it remains challenging to ensure that security defects are prioritised against functional requirements. Potential approaches to overcome this commonly found issue is to set time limits for security defects [42] or dedicating a fixed percentage of effort or sprints to implementing security related features and fixing defects. It is advised to ensure that weaknesses detected in production are included in the QA processes to reduce the risk of regressions [44].






### Expert ranking

Within DevSecOps feedback loops are considered crucial to share both successes and issues as a way to foster collaboration and improve the quality of the results. Continuous feedback from production to development is expected to contribute to this objective while also ensuring that issues do not go unnoticed for weeks. There is agreement within the expert panel that performing continuous feedback from production to development is highly effective (mdn= 4.5, IQR= 1.25), does not cause significant delays (mdn= 4.0, IQR= 1.50) and does not have a significant financial impact (mdn= 4.0, IQR= 1.00). This is under the assumption that appropriate stakeholder

management is applied to ensure that only relevant roles are included and that the feedback is automated to the extend possible.

Frameworks	
<b>OWASP SAMM v2.0</b>	Operations: Incident management
<b>BSIMM v10</b>	Deployment: Configuration Management & Vulnerability Management

Design factors	Relevance
Set a time limit for all lower-priority security defects	 5 of 7 experts
Give attack patterns to your developers	 7 of 7 experts
Build an internal forum to discuss attacks	 5 of 6 experts
Establish emergency code base response	 5 of 6 experts
Incorporate security tests as part of QA for detected incidents	 5 of 6 experts

Quick facts	
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>Foster collaboration</li> <li>Ensure that issues are detected and resolved faster</li> </ul>
<b>What to avoid?</b>	<ul style="list-style-type: none"> <li>Perform all activities manually</li> <li>Involve too many or incorrect roles in the organisation</li> </ul>

Expert ranking	Median	IQR	
<b>Effectiveness</b>	4.5	1.25	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	4.0	1.50	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	4.0	1.0	Higher scores indicate lower financial consequences

## Provide security training

Software security education is an important aspect of security regardless of the chosen development approach. Some researchers argue that security training gains in importance in agile development teams due to the decentralisation of responsibilities and the increased speed of deployment.

How much education is required remains a matter of debate as it is not possible to teach every developer to be a software security expert [42][45] which was also mentioned as a limitation by experts in this research.

### Design factors

A potential alternative approach to security training is to involve developers in building security tools which provides them learning opportunities and gives the added benefit of improving DevSecOps processes [20]. The DevSecOps expert panel indicated that security training should be hands-on.

The experts suggest to approach training with a broader perspective and to target all the relevant roles including product owners. Some topics to consider are educating teams how to refactor software and explaining to product owners that this is very important, teaching approaches to think from a risk perspective, perform risk prioritisation and explaining that cloud native services are not secure by default were given as examples on principles to teach stakeholders.

### Expert ranking

Sharing and increasing knowledge within the DevSecOps teams is perceived as an important enabler for DevSecOps. Tailoring the level of security training based on the criticality of the role and the current level of security mindset is recommended. Training on specific topics only needs to be performed once however it is important to ensure that people put their new skills and knowledge into practice. The expert panel agrees that providing security training is an effective activity (mdn=4.0, IQR=1.00) and that it does not cause significant delay (mdn=4.0, IQR=1.50). They do however strongly agree that moderate financial impact is to be expected (mdn=3.0, IQR=0.50). High quality external training material can be expensive therefore the financial consequences are expected to be moderate.

### Frameworks

**OWASP SAMM v2.0** Governance: Education and guidance

**BSIMM v10** Governance: Training

### Design factors

Teach every developer enough to enable them to identify areas where they would benefit from the advice of an expert



Security training should be hands-on

Added by expert n/a

Educate teams how to refactor software

Added by expert n/a

Educate product owners on the importance of security

Added by expert n/a

Ensure that team members can put their new skills and knowledge into practice

Added by expert n/a

Organise centralised training sessions and workshops

Added by expert n/a

### Quick facts

**Why should you do this?**

- Increase the security knowledge and mindset in the organisation

**What to avoid?**

- Not tailoring the contents of the trainings to the specific roles

### Expert ranking

	Median	IQR	
<b>Effectiveness</b>	4.0	1.00	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	4.0	1.50	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	3.0	0.50	Higher scores indicate lower financial consequences

## Establish security satellites

Various studies point out the importance of establishing security satellites as part of the development teams. The principle is that the security team teaches one of the developers about security, and then [that person] disseminates the information to the rest of the team. It's really about knowledge sharing [44].

The importance of embedding security knowledge and responsibility within each development team originates from the idea that short-term helicopter-style incursions of a centralised security team will inevitably be perceived as an outside agent hindering progress [42]. The presence of security-minded developers across the organisation will aid in resolving quick fixes in case of incidents. The term "satellite" is the one used in the BSIMM; some organisations have formalised this as a security champion for some or all development teams [43].

Within the expert panel there was some debate on the definition of the term where experts indicated a difference between security satellites and security champions where the first was perceived as a member of an external security team who could be contacted by the DevOps teams whereas the latter would be a person embedded in the DevOps teams themselves. For the purpose of this research the terms are used interchangeably and adhere to the definition of a security champion.

### Design factors

Experts from the DevSecOps panel point out that establishing security satellites is a starting point to implement the principle of 'Security to the left'. However they feel that the most effective way to implement security champions is to allow the teams to discover the usefulness of this role by themselves and not having it forced upon them.

### Expert ranking

Establishing security satellites allows injecting the DevSecOps teams with security knowledge and best practices. It also provides a feedback loop from the teams towards the central security team on their needs and helps fostering a security mindset in the organisation. Giving appropriate levels of authority to security satellites increases their efficiency. The expert panel agrees that this activity is highly effective (mdn=5.0, IQR=1.00), this is expected to be the case even more in large organisations where there may be boundaries between the development and centralised information security teams. Incorporating this activity in the approach is not believed to cause significant delays in the DevSecOps process (mdn=4.0, IQR=0.50). Different views exist on the expert panel regarding the financial consequences as they are expected to be in direct relation to the time assigned to security duties (mdn=3.0, IQR=3.0).

Frameworks			
<b>OWASP SAMM v2.0</b>	Governance: Education and guidance		
<b>BSIMM v10</b>	Governance: Training		
Design factors			Relevance
Should not be forced but discovered			Added by expert n/a
Should be provided sufficient authority			Added by expert n/a
Quick facts			
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>Improve the security mindset</li> <li>Share knowledge and best practices</li> <li>Gather feedback from teams on security aspects</li> </ul>		
<b>What to avoid?</b>	<ul style="list-style-type: none"> <li>Not providing a sufficient level of authority to security satellites</li> </ul>		
Expert ranking	Median	IQR	
<b>Effectiveness</b>	5.0	1.00	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	4.0	0.50	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	3.0	1.50	Higher scores indicate lower financial consequences

## Practice incident response

DevSecOps teams should organise red-team exercises performing security drills on the deployed software [46]. Such preparedness exercises should involve both incident responders and developers [43]. The objective of the exercise is to find and exploit vulnerabilities in the system thereby not only assisting in the identification of security flaws but also providing metrics on incident response capabilities. These metrics can be leveraged to find solutions and improvements. An additional benefit to these exercises is that the break silos and facilitate collaboration [44].

## Expert ranking

Practicing incident response from DevSecOps perspective aims at involving the development and operations team in the exercise. Doing so allows not only a more effective response to incidents in the future but also to improve the security mindset across the various roles involved. The expert panel agrees that this activity can be effective but to varying degrees (mdn=4.0, IQR=1.25). There is more dissonance on the delay (mdn=4.0, IQR=1.25) and financial consequences (mdn=3.0, IQR=1.50) of this activity which depends on when and by whom this activity is performed. The expert panel points out that incident response exercises could be performed during low time.

Frameworks			
<b>OWASP SAMM v2.0</b>	Incident Management: Mature incident management		
<b>BSIMM v10</b>	Deployment: Configuration management & vulnerability management		
Quick facts			
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>Increase effectiveness of incident response</li> <li>Increase security mindset across various roles</li> </ul>		
<b>What to avoid?</b>			
Expert ranking		Median	IQR
<b>Effectiveness</b>	4.0	1.25	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	4.0	1.50	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	3.0	1.50	Higher scores indicate lower financial consequences

## Establish a security mindset across the organisation

The DevSecOps expert panel indicated the importance of training management on security concepts as an enabler for security. Product owners and project managers are directly involved in balancing security with functionality while taking into account business value. This makes them important stakeholders.

## Expert ranking

The experts mostly agree that this activity can be highly effective (mdn=4.0, IQR=1.25) and is not expected to cause significant delays (mdn=4.0, IQR=1.50). There is debate in terms of financial consequences. Some experts indicated that implementing this activity requires a lot of effort (N=1, 28%) while others indicated that this does not necessarily have to be the case (N=4, 42%). Regardless of effort changing a mindset is expected to be a slow and gradual process .

Frameworks			
<b>OWASP SAMM v2.0</b>	Governance: Education and guidance		
<b>BSIMM v10</b>	Governance: Training		
Design factors			Relevance
Train management on security concepts			<b>Added by expert</b> n/a
Reward the teams who fix the most vulnerabilities			<b>Added by expert</b> n/a
Share best practices all across the organisation			<b>Added by expert</b> n/a
Establish a code quality standard for each programming language			<b>Added by expert</b> n/a
Quick facts			
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>Obtain a good balance between security and functionality from a business value perspective</li> </ul>		
<b>What to avoid?</b>	<ul style="list-style-type: none"> <li>Limiting the "security mindset" to technical roles</li> </ul>		
Expert ranking		Median	IQR
<b>Effectiveness</b>	4.0	1.25	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	4.0	1.50	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	3.0	2.00	Higher scores indicate lower financial consequences

# Use of non-automated activities

## Performing security requirements analysis

Software development is usually performed based on requirements defined by stakeholders. In many cases these requirements focus on functional requirements and omit security focused requirements. Studies call for the definition of security requirements so that they are included from the start of a project [44].

### Design factors

Security requirements should be testable, clear, consistent, complete, unambiguous, measurable and accompanied by acceptance criteria. [47]

A commonly suggested approach to identify these security requirements is through the use of ‘abuse cases’ and ‘misuse cases’ which provides an inverse perspective leading to the identification of security requirements. Another approach for security requirements analysis is to leverage goal-oriented requirements analysis (GORE) [48]. A detailed explanation of goal-oriented requirements analysis is not feasible in the content of this research however various papers and case studied can be found in literature.

The threat modeling and risk analysis steps described elsewhere can also be leveraged to provide input for the security requirements analysis phase. And the security requirements identified during this activity may also be leveraged to compare design options to select the most appropriate approach from a security perspective [42].

The expert panel involved in this research agreed that security requirements form an important aspect in DevSecOps. However a recurring remark during the survey and interviews is that very often security requirements are set back in favour of functional requirements by the business stakeholders. Various experts wonder how to strike the right balance between functional and non-functional requirements. An approach which was applied at one of the respondent organisations was to dedicate a fixed percentage of effort during each sprint to security related requirements.

### Expert ranking

Effective security requirements analysis requires a tailored approach, the effort spent defining security requirements needs to be in line with the level of sensitivity of the application. The definition of high level security requirements itself is not so complex however detailing them towards technical requirements is complicated and requires significant security knowledge. Therefore it makes sense to stop at the definition of high level security requirements for less sensitive applications while going into greater technical detail for sensitive applications. Overall security requirements analysis is perceived as moderately effective (Mdn=3.0, IQR=0.50) which is counterintuitive seeing that experts





throughout this research indicated that security requirements form a corner stone in the process from design over implementation to testing. Delays caused by performing security requirements analysis and the financial consequences are directly related to the depth of the analysis and whether or not the analysis is incorporated in the sprints or not. These variables may explain the different views of experts. A suggested approach would be the creation of proper baselines acting as boiler plates to speed up requirements analysis for common applications.

### Frameworks

<b>OWASP SAMM v2.0</b>	Intelligence: Security features and design
<b>BSIMM v10</b>	Design: Security requirements

---

### Design factors

	Relevance
Treat security requirements as nonfunctional requirements	 9 of 9 experts
Leverage metrics gathered during the security requirements analysis phase to evaluate the security level of alternative designs	 5 of 7 experts
Enable the evaluation of alternative designs through suitable metrics during security requirements analysis to determine variations in security levels of a given design and make appropriate choices	 5 of 5 experts
Leverage goal-oriented requirements analysis (GORE) to perform security requirements analysis	 5 of 5 experts
Leverage process metrics	<b>Added by expert</b> n/a
Establish rules to balance functional and non-functional requirements	<b>Added by expert</b> n/a
Leverage Model-based systems engineering	<b>Added by expert</b> n/a

---

### Quick facts

<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>Ensure security aspects are taking into account during development and testing</li> <li>Identify business logic errors</li> </ul>
<b>What to avoid?</b>	<ul style="list-style-type: none"> <li>Applying the same rigour in terms of security requirements analysis to every application regardless of their level of sensitivity</li> </ul>

---

	Median	IQR	
<b>Effectiveness</b>	3.0	0.50	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	3.0	1.50	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	3.5	2.00	Higher scores indicate lower financial consequences

---

## Performing threat modeling

---

Threat modeling aims at identifying potential weaknesses in a given design and the related threats which could be posed by them. As such threat modeling can be considered as a potential input for risk analysis during the design phase of an information system.

### Design factors

Performing threat modeling from the initial planning stage while placing the results in context through a risk analysis is paramount to establish security by design. By applying this process iteratively throughout the development cycle an approach to identify and prioritise weaknesses and examine controls already in place allows an effective risk treatment strategy to be defined [46].

Threat modeling starts by creating an abstraction of the application under design or development usually employing a Data Flow Diagram. A threat modeling approach, such as STRIDE [49] is applied by iterating over the model elements to identify all potential security threats. The approaches to identifying potential threats are numerous ranging from the use of mnemonics, which is more suitable for experienced security professionals, to the use of attack trees or serious games such as “Elevation of Privilege” [50] or “OWASP Cornucopia” [51] empowering development and operations teams to perform threat modeling autonomously. Results obtained from threat modeling can also be used to develop tests to be incorporated as part of the Quality Assurance (QA) process [44].

Researchers [48] distinguish several categories of threat modeling techniques: risk-centric, attack-centric or software-centric. Not all threat analysis however can be categorised in the aforementioned groups. In the context of DevSecOps analysis velocity is preferred over analysis systematicity. Therefore a strong focus on the most important assets is key to allow timely results.

Shostack argues that software-centric threat modeling techniques are most suitable to identifying threats in information systems. He also mentions that it makes sense to perform threat modeling without (security) experts due to their short supply and that by including the people involved in building the system they obtain a sense of ownership and an understanding of the security model [52].

The DevSecOps panel who were consulted during this research confirmed that the various threat modeling approaches (risk-, attack- and software-centric) can prove valuable depending on the objective and scope of the threat modeling activity. The use of different techniques such as attack trees, abuse cases and problem frames is recommended to assist in the identification of potential weaknesses. However the expert panel places emphasis on selecting a threat modeling approach which is iterative and where possible assisted by automation to reduce the time required to perform a threat modeling exercise. Annotating the outcome of the threat modeling directly in the codebase can be a good method to allow traceability. However, according to experts, these techniques have not yet matured to the level where they can be leveraged broadly.

Teams are more likely to consistently perform threat modeling at the start of each iteration if the time required to perform the activity is reduced. This leads to a wider coverage of the threat models. A challenge related to such an iterative approach is to ensure that the full breadth of the system is covered. There is a need to maintain oversight on the compatibility of the threat modeling outcomes from both a scope and a result perspective.

### Expert ranking









Overall experts on the panel perceive threat modeling as moderately effective (Mdn=3.0, IQR=0.50). There is however a wide range of approaches to threat modeling with different outcomes depending on the technique, the goal and even the stage in the DevSecOps cycle in which the activity is undertaken. This makes it difficult to achieve a common view on the topic. According to the experts threat modeling allows the detection of weaknesses before the actual implementation is started and can be leveraged as the basis for risk analysis and security testing. It also allows the identification of the potential blast radius of an attack by analysing the potential chains of weaknesses. Some experts point out that manual threat modeling is only moderately effective for these purposes, instead they recognise a significant advantage in using threat modeling to increase the security mindset in developers by raising awareness. Most experts agree however that automation of threat modeling and focusing on technical security components are key characteristics of effective threat modeling. Threat modeling is considered to have a moderate impact in terms of delay and financial consequences. In principle it can be done reasonably fast by experienced people with appropriate tooling. Licenses for these tools and the required training to do it will have however a cost associated.

## Frameworks

**OWASP SAMM v2.0** Design: Threat assessment

**BSIMM v10** Intelligence SSDL touchpoints: Attack Models Architecture Analysis

## Design factors

	Relevance	
Perform threat modelling from a risk-centric perspective		10 of 10 experts
Perform threat modelling from an attack-centric perspective		10 of 10 experts
Perform threat modelling from a software-centric perspective		8 of 9 experts
Ensure compatibility of threat modelling outcomes from a scope and result perspective		6 of 6 experts
Introduce abuse cases and problem frames to perform threat modeling		8 of 8 experts
Make use of attack or threat trees to perform threat modelling		10 of 10 experts
Implement traceability of threat modelling (results) in the code base		6 of 6 experts
Automated threat impact analysis		7 of 7 experts
Threat modeling should be performed through an iterative approach	Added by expert	n/a
Leverage both quantitative and qualitative approaches	Added by expert	n/a

## Quick facts

- Why should you do this?**
- As a starting point for risk analysis
  - To identify areas for security testing
  - To identify the blast radius of potential attacks
  - To increase the security mindset (when performed manually)
- What to avoid?**
- To apply threat modeling at the business process level

## Expert ranking

	Median	IQR	
<b>Effectiveness</b>	3.0	0.50	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	3.0	1.00	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	3.0	0.25	Higher scores indicate lower financial consequences

## Performing risk analysis

Risk analysis aims to identify risks and quantify them as a function of likelihood and impact. The results of a threat modeling exercise can be placed in context through a risk assessment. A calculation is applied to the identified weaknesses based on attributes such as likelihood and impact to determine a risk factor while taking into account existing countermeasures.

The DREAD framework, which defines the attributes Damage, Reproducibility, Exploitability, Affected Users and Discoverability, is sometimes used as a methodology in (technical) risk assessments. Based on experiences at Microsoft Shostack states that DREAD is not well suited for software-centric risk analysis as it seems to add numbers without defining their scales, generating a risk of making a risk assessment appear algorithmic when it's not [52].

## Design factors

The outcome of a risk assessment enables stakeholders to decide on risk treatment options taking into account effectiveness, cost and impact. It is emphasised that all stakeholders, including business, should be part of the risk assessment to ensure that security is taken into consideration during product development and to provide visibility on a continuous basis [47].

Previous studies on multi-cloud DevOps implementations [53] indicate that continuous risk assessment is also key in selecting the security controls and metrics to be included in Security SLA's for cloud services.

## Expert ranking

Performing risk analysis is perceived by the expert panel as a very effective approach (mdn=4.0, IQR=0.50) to prioritise security efforts such as the rigour applied in defining security requirements and performing security testing. When performing risk analysis it is important to ensure the approach remains practical and yields tangible results. This prevents it becoming a theoretical exercise which provides little benefits. Direct impact on delay in the DevSecOps process is expected to be relatively low (mdn=3.5, IQR=1.00) as it is assumed this process is performed in a parallel track and in all cases is completed before the actual technical activities start. However selecting a practical approach with the appropriate level of intensity is key to ensure that team members with the required security knowledge to complete this activity can keep up with the pace of development. The financial impact of performing risk analysis is presumed to be limited (mdn=4.0, IQR=0.25) assuming that, as recommended, a practical and pragmatic approach and intensity is selected.



## Frameworks

**OWASP SAMM v2.0** Design: Threat assessment

**BSIMM v10** Intelligence SSDL touchpoints: Attack Models Architecture Analysis

## Design factors

	Relevance	
Performing risk analysis continuously before each iteration	Not validated	n/a
Performing risk analysis during the design phase	Not validated	n/a
Include a broad range of stakeholders including the business owner when setting security goals	Not validated	n/a
Establish clear rules regarding information exchange across teams and maintain a log for every access to sensitive data	Not validated	n/a
Provide security knowledge and tools and encourage the Development and Operations teams to integrate themselves	Not validated	n/a
Consider gamification for finding vulnerabilities or bugs	Not validated	n/a

## Quick facts

- Why should you do this?**
- To prioritise security related efforts
- What to avoid?**
- Performing risk analysis to accurately measure risk

## Expert ranking

	Median	IQR	
<b>Effectiveness</b>	4.0	0.50	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	3.5	1.00	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	4.0	0.25	Higher scores indicate lower financial consequences

## Establishing security SLA's for cloud providers

Various cloud technology oriented research papers propose approaches to the definition of security service level agreements for cloud providers. The challenge resides in generating the required Security SLA's for across various cloud providers and maintaining oversight of the applications consuming the cloud services related to these security SLAs [54]. An area of research related to this is the automation of Cloud Security SLA generation.

## Design factors

The DevSecOps expert panel mentioned that it is important for a DevSecOps team to be involved in determining the required and desired cloud provider from a security control capability perspective. They also mention that education of the decision makers with regards to the retention of responsibility for security in cloud environments is key.

## Expert ranking

Expert panel members point out the importance of availability and integrity aspects of information systems residing in the cloud. Establishing security SLAs can be seen as a measure to achieve certainty regarding these two crucial aspects. Other experts however point out that these security SLAs only have minimal effect on the actual security levels and that the statements in these SLAs are defined on such a high level that they do not necessarily provide tangible benefits. This explains part of the variability in the effectiveness scoring, the experts agree on the importance but see varying levels of actual benefits to investing effort in establishing these SLAs. About one third of the experts perceives this activity as very effective (N=4, 37%) whereas another third of the experts indicate low effectiveness (N=2, 37%) The experts do also not seem to have a common view on how much delay establishing these security SLAs introduces in the DevSecOps process (mdn=4.5, IQR=2.25), some point out that in most cases this activity is not organised in-line with the pipeline and therefore there should not be any delay. When looking at the financial impact of this activity the panel agrees that in general establishing the SLAs does not necessarily introduce significant costs (mdn=3.5, IQR=1.75), in some cases premium services may be required to obtain reasonable SLAs which in turn would represent a considerable financial consequences.

Frameworks			
<b>OWASP SAMM v2.0</b>	Design: Security requirements		
<b>BSIMM v10</b>	Intelligence: Security features and design		
Design factors		Relevance	
Ensure that the DevSecOps team is involved in determining the required cloud provider security controls		<b>Added by expert</b> n/a	
Educate decision makers on the retention of responsibility for security in cloud environments.		<b>Added by expert</b> n/a	
Quick facts			
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>To obtain additional securities regarding availability and integrity of cloud services</li> </ul>		
<b>What to avoid?</b>			
Expert ranking			
	Median	IQR	
<b>Effectiveness</b>	3.0	2.00	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	4.5	2.25	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	3.5	1.00	Higher scores indicate lower financial consequences

## Performing continuous assurance

Performing continuous assurance as an activity refers to continuously checking if a system satisfies the federal regulations set by the government as per the domain of interest [12]. This activity can also be defined more broadly to include compliance with internal policies.

### Design factors

The majority of experts on the DevSecOps panel agreed with the relevance of this activity and pointed in the direction of “policy-as-code” whereby requirements would be expressed in code and automatically enforced through the infrastructure and the CI/CD pipeline.


### Expert ranking

Expert panel members believe that compliance-as-code is key to highly regulated industries however there is some disagreement on its bottom line effectiveness. Over half of the experts rated this activity as very effective (N=4/N=5, 60%) while some rated it as very ineffective. These experts warn that compliance should not be confused with security, an insecure application may still be compliant. Performing continuous assurance may increase security awareness thereby helping the establishment of a security mindset within the organisation. This activity is expected to create a significant delay in the DevSecOps process due to the coordination efforts between the large number of stakeholders involved (CISO, Compliance, Business, Dev and Ops). Judging the financial impact of continuous assurance is complex when offsetting the cost against the potential gains in risk avoidance. The experts point out that for this reason the business should always be in the driver seat to decide on the balance between the cost of non-compliance (mainly determined by fines) and the cost of compliance (including effort but also the cost of delaying a potentially lucrative business feature). Overall the inherent financial impact of performing continuous assurance is perceived as significant.

Frameworks			
<b>OWASP SAMM v2.0</b>	Governance: Policy & Compliance		
<b>BSIMM v10</b>	Governance: Compliance & Policy		
Design factors			Relevance
Leverage compliance-as-code			Added by expert n/a
Quick facts			
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>Potentially part of the barrier to entry for your industry</li> <li>To avoid impact from sanctions and fines</li> <li>To establish a security mindset</li> </ul>		
<b>What to avoid?</b>	<ul style="list-style-type: none"> <li>Not putting the business stakeholders in the drivers seat when determining compliance objectives</li> </ul>		
Expert ranking		Median	IQR
<b>Effectiveness</b>	4.0	2.00	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	3.0	2.00	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	3.0	0.25	Higher scores indicate lower financial consequences

## Performing manual security testing

DevOps in general and DevSecOps specifically have place a strong emphasis on the automation of activities. Never the less some manual security testing activities are referenced both in literature and during interviews with members of the expert panel. The two main types of activities which were distinguished consist of “manual penetration testing” and “manual security reviews”. Whether they are part of DevSecOps is under debate however their effectiveness is generally agreed upon.

Frameworks		
<b>OWASP SAMM v2.0</b>	Verification: Security testing	
<b>BSIMM v10</b>	SSDL touchpoints: Security testing	
Design factors		Relevance
Limit manual penetration testing to critical components or perform in parallel to reduce impact on deployment lead times		 5 of 8 experts
Automated security testing is an important enabler to increase value of manual testing		Added by expert n/a
Performing manual penetration testing should depend on the criticality of the application and the level customisation		Added by expert n/a

## Performing manual penetration testing

Manual penetration testing provides a view of the potential attack surfaces and methods to which an application is vulnerable through real world testing by a security professional. The results of manual penetration testing are proportional to the knowledge of the person performing the test and the time attributed to the engagement. In general manual penetration testing allows for more accurate results because a human can try to examine context which is otherwise difficult to automate [55]. This activity is mentioned in several research studies [42][43].

An issue mentioned by researchers [55] and members of the DevSecOps expert panel is that performing manual penetration testing in environments where changes are deployed frequently is difficult to implement due to the time constraints.

A proposed approach to overcome this limitation is to limit manual penetration testing to critical components or perform them in parallel to the continuous deployment process to reduce the impact on deployment lead times [55].

The DevSecOps expert panel points out that performing automated security testing should be seen as an enabler for valuable manual security testing. Recurring low level activities can be performed through automated security testing allowing the penetration tester to focus on aspects which cannot be automated. They also confirm that the necessity to perform manual security testing could be determined based on the criticality of the application and the level of customisation the application has gone through.

## Expert ranking

Manual penetration testing is rated very effective by the expert panel (mdn=4.0, IQR=1.50), however there are some differing views. The skillset of the person performing the penetration test, the process to handle the results of the penetration test and the frequency of testing are expected to have an impact on the effectiveness. Experts who scored the activity lower in terms of effectiveness point out that for these reasons it is more beneficial to focus on automated penetration testing (N=2, 25%). Experts who rated the activity as highly effective state that a skilled penetration tester with proper preparation (e.g. access to security requirements and threat modeling outcomes) is very effective in identifying vulnerabilities in applications. The experts agree that penetration testing is very time consuming (mdn=2.0, IQR=1.50) and therefore must be organised outside of the DevSecOps pipeline to avoid introducing delays. The time required combined with the expert skills required to perform this activity results in a significant financial impact (mdn=2.0, IQR=1.25).

Quick facts			
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>Obtain direct results in terms of vulnerabilities and exploitability in an application</li> <li>Perform a penetration test as a one off prior to going to production</li> </ul>		
<b>What to avoid?</b>	<ul style="list-style-type: none"> <li>Perform a penetration test without providing output of requirements analysis and threat modeling activities</li> <li>Perform penetration testing in-line with the deployment pipeline</li> </ul>		
Expert ranking		Median	IQR
<b>Effectiveness</b>	4.0	1.50	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	2.0	1.50	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	2.0	1.25	Higher scores indicate lower financial consequences

## Performing manual security review

The activity of performing a manual security review is viewed as an approach to increase the security of software [20][43]. The activity consists of a manual review taking into account the security requirements and outcomes of the threat modeling phases to ensure that the required security measures are implemented correctly. Stakeholders could be invited to also perform security tests during the product review, also referred to as a demo, providing the chance to break the system's security and try things that intruders or deceptive users would do to see how the system reacts [47].

## Expert ranking

The expert panel generally perceive manual security reviews to only be moderately effective (mdn=3.0, IQR=1.50). Some experts indicated that it can act as an effective way to get a quick insight in the actual security level of an application given that this is not just performed based on documentation but consists of an actual review of the implementation together with the development and operations teams. The expert panel does agree that this activity is expected to increase significant delays (mdn=2.0, IQR=1.00) in the DevSecOps process because it requires time from the development resources. The financial impact is expected to be moderate (mdn=3.0, IQR=1.25).

Quick facts			
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>Gain insight in the way security requirements are actually implemented in an application</li> </ul>		
<b>What to avoid?</b>	<ul style="list-style-type: none"> <li>Do not base the manual security review purely on documentation</li> </ul>		
Expert ranking		Median	IQR
<b>Effectiveness</b>	3.0	1.50	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	2.0	1.00	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	3.0	1.25	Higher scores indicate lower financial consequences

## Secure the CI/CD pipeline

The CI/CD pipeline, forming the technical backbone of DevSecOps activities, may itself be vulnerable to security attacks or misconfigurations [56]. The security of a deployment pipeline may be threatened by malicious code being deployed through the pipeline or by allowing direct communication between components in the testing and production environments [57]. Therefore it is important to ensure the CI/CD pipeline itself is properly secured and that proper role based access permissions and auditing is performed against the automated activities included in the pipeline.

When looking at the security of the pipeline three distinct scenarios play a role: (a) the pipeline may deploy an artefact which has not been validated, (b) an artefact may be deployed without going through the complete pipeline or (c) the production environment may be accessible from a different environment. The objective of securing the CI/CD pipeline should be to provide assurance that the pipeline is secure against attacks and cannot be made to behave in a way that is not the intended one.

### Expert ranking

There is some debate about the effectiveness, delay and financial cost of securing the CI/CD pipeline within the expert panel. Overall they believe this activity to be very effective (mdn=4.0, IQR=1.50) , some experts pointed out that the CI/CD pipeline is the central axe along which DevSecOps activities are performed and therefore securing it is key. It is also assumed that a base level of security can be fairly easily attained because most CI/CD related tools and products have some protections already built-in to them. Delay to the DevSecOps process is expected to be low (mdn=4.0, IQR=1.25) because securing the pipeline is a parallel process and is not subject to continuous change. The financial cost of this activity is determined by the complexity of the pipeline and the chosen depth of the security controls and therefore moderate financial impact can be expected (mdn=3.5, IQR=1.25). Costs to be account for include the time required to implemented the selected controls and potential associated licensing costs.

### Frameworks

<b>OWASP SAMM v2.0</b>	Implementation: Secure build & deployment
<b>BSIMM v10</b>	Deployment: Configuration & Vulnerability management

### Quick facts

- |                                |   |
|--------------------------------|---|
| <b>Why should you do this?</b> | <ul style="list-style-type: none"><li>• If the CI/CD pipeline plays an important role in developing and subsequently deploying applications</li></ul> |
| <b>What to avoid?</b>          | <ul style="list-style-type: none"><li>• Not leveraging the default security controls build into the tooling</li></ul>                                 |

### Expert ranking

	Median	IQR	
<b>Effectiveness</b>	4.0	1.50	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	4.0	1.25	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	3.5	1.25	Higher scores indicate lower financial consequences

# Use of automated activities

## Performing automated security testing

The objective to automate security controls and verifications where possible is a core concept in DevSecOps [46]. To achieve this a wide variety of tools are integrated with the automated deployment pipeline [44]. An option to speed up implementation and adoption is to leverage self-managed, automated and scalable security services provided by SecaaS platforms [45].

The DevSecOps expert panel point out the importance of the “Fail fast” principle when evaluating efficiency gains through automated security testing. These automated security tests should be integrated early in the development cycle allowing short feedback loops on security. It is important to keep in mind that good APIs for automated security testing tools can significantly facilitate organisational processes. Overall the implementation should be understandable to developers and the expected outcomes should be included in the definition of done.

Due to the breadth and depth of this activity three detailed security activities were derived from this activity.

Frameworks	
<b>OWASP SAMM v2.0</b>	Verification: Security testing
<b>BSIMM v10</b>	SSDL touchpoints: Security testing

---

Design factors	
Leverage SecaaS by using cloud provided self-managed, automated and scalable security services	 7 of 8 experts
Integrate the security tools in an automated deployment pipeline	 10 of 10 experts
Automate as many security controls and verifications as possible	 10 of 10 experts
Ensure the team and management understands and supports the security validations integrated in the automated deployment pipeline	Added by expert n/a
Fail fast when security validations do not pass	Added by expert n/a
Integrate the validations in the Definition of Done	Added by expert n/a
Ensure APIs (of security verifications) align with organisational processes allowing the implementation to be easy to understand	Added by expert n/a
Automated testing is geared towards finding implementation bugs but generally not suited to spot design flaws.	Added by expert n/a

## Performing automated run-time testing

Automated run-time testing, also referred to as Dynamic Application Security Testing (DAST) is a testing methodology where an application is tested from the outside. It requires the application to be running and functional in order to be able to scan for security vulnerabilities.

The limited scope of automated run-time testing is an important drawback of this technique [55]. As pointed out by an expert on the DevSecOps panel automated run-time testing has issues supporting newer (web) technologies leading to a severely reduced coverage.

## Design factors

To compensate for this limitation one could leverage functional tests by running them through automated run-time testing tooling allowing the tool to cover a broader set of the application function. An important note is that in this case the coverage of the scans is dependent on the coverage of the functional tests [55].

Experts on the DevSecOps panel point out that implementing good automated run-time testing is challenging when DevOps is not at a proper level (of automation). The recommended approach is to start small and extend later on by focusing on the most relevant applications.

## Expert ranking

The expert panel mostly agrees that automated run-time testing is moderately effective (mdn=3.0, IQR=1.25). Experts who rated the effectiveness lower have done so indicating that this technique has become less effective in detecting vulnerabilities in modern technologies (Single Page Applications, SSO, MFA) and the potential number of high false positives. Those who rated it as very effective indicated that it provides good direct feedback and is able to cover a large volume of applications. The delays caused in the DevSecOps process are expected to be limited (mdn=4.0, IQR=0.50) assuming that the time required to perform a scan is maintained at a low level and the number of false positives is limited. Implementing these tools to block the pipeline on failures is expected to cause significant delays. There is debate regarding the financial consequences of automated run-time testing. Some experts rated this as very low whereas other rated it very high. This can be explained by the choice of tools and licensing models. The financial impact is considered low if you opt for using some of the open-source tools available, however it may rise quickly if top-of-the-line commercial products are selected.

## Design factors

Perform automated run-time testing at four levels: (1) pre-authentication scanning, (2) post-authentication scanning, (3) independent backend scanning and (4) complete workflows

Ensure automated run-time testing is implemented for a broad scope of test scenarios

Ensure proper unit tests are in place to optimise run-time testing efficiency

Start small, extend later on.

Get DevOps to a proper maturity level before implementing automated run-time testing.



## Quick facts

### Why should you do this?

- If you need to cover a large number of web applications

### What to avoid?

- Implement block on fail

## Expert ranking

	Median	IQR	
<b>Effectiveness</b>	3.0	1.25	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	4.0	0.50	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	3.0	1.50	Higher scores indicate lower financial consequences

## Performing automated static testing

During automated static testing, also referred to as Static Application Security Scanning (SAST), the application code is analysed with the purpose of identifying potential security or other quality related errors. These errors are usually related to coding mistakes or bad coding practices which may introduce vulnerabilities in the application under development.

It is argued that this verification should be performed during the software development to allow shorter feedback loops in the development sprints [47]. By integrating an automated code review into the development process appropriate feedback can be provided to the software developers of interest, using open source and commercial static analysis tools [12]. The extent of static code analyses in the CI chain is recognised in the DevSecOps maturity model as one of the four important axes to achieve security aspects [20].

## Design factors

An important design factor for automated code review is the speed and accuracy of the feedback provided to developers. If the feedback loop created through automated code review is slow developers are less likely to include it in their workflow or will use it less frequently. Additionally, if the results of the automated code review contain too many false positives the developer will need to spend a lot of time sifting through the findings and is more likely to miss important findings or to dismiss the results all together [55].


According to the the DevSecOps expert panel it is important to ensure sufficient code coverage by automated static testing. Static testing performed on modules may not properly follow complete flows when applications are not scanned as a whole. It is also important to require independent justification upon marking findings as false positives and to perform adequate follow-up to ensure fixes genuinely address the root cause of the issue. Building a good process around static testing is key, one approach which has proven useful is to implement so called build breakers which stop the CI/CD pipeline upon discovery of new severe issues.

One should consider to prioritise efforts on static code analysis for the most relevant applications due to the time and effort required to get it right.

## Expert ranking

There is debate among the experts regarding the effectiveness of performing static security testing (mdn=3.5, IQR=2.0). The majority believes however that it is very (N=5, 36%) to highly effective (N=4, 14%) while one expert find it highly ineffective (N=1, 12%). The main limitation to the effectiveness of static testing is the potential high number of false positives. These same false positives are also the main source of delay, in general running the tooling can be fairly fast however managing the false positives can become time consuming, certainly when used in blocking mode where the pipeline is

interrupted upon issues being detected. The impact of false positives can be reduced by performing a baseline scan to suppress false positives and creating global lists of accepted false positives. This explains the slight disagreement between the experts (mdn=3.0, IQR=1.50). A significant financial impact is expected (mdn=2.5, IQR=1.00) as most experts point out that there is a lack of good open-source tooling and the commercial products can become quite expensive.

Design factors		Relevance	
Minimise the number of false positives resulting from static testing		8 of 8 experts	
Independent justification of whether a false positive is really false and whether the fix genuinely addresses the cause	Added by expert	n/a	
Ensure code coverage (applications should be tested as a whole, not scans run on separate modules, otherwise the static testing will not properly follow flows).	Added by expert	n/a	
Should prioritise the most relevant applications first since this takes a lot of time.	Added by expert	n/a	
Good process is key, build breakers are important.	Added by expert	n/a	

---

Quick facts	
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>Provide fast feedback loops for developers on common secure coding violations</li> </ul>
<b>What to avoid?</b>	<ul style="list-style-type: none"> <li>Not tuning the rulesets by performing a baseline scan or establishing lists of globally accepted false positives</li> </ul>

---

Expert ranking	Median	IQR	
<b>Effectiveness</b>	3.5	2.00	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	3.0	1.50	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	2.5	1.00	Higher scores indicate lower financial consequences

## Integrate security tests in unit testing

Security testing should start at the feature or component level, before any system integration is performed. Tests should cover both unauthorised misuse and violations of the assumptions on which the component was build [12].

### Design factors

It is recommended to include tests to verify that security requirements are correctly implemented for a given feature. This applies not only to specific security features but also for non functional security requirements of other features that are used in security contexts. Several experts on the DevSecOps panel indicated that security testing should start as early as possible in the development cycle (shift left) and that good coverage of security aspects in unit testing allows, in combination with other activities, the security experts to focus on the complex aspects of security.

### Expert ranking

Security experts strongly disagree on the effectiveness of this activity. Half of the expert panel indicated that this activity is highly effective however some other experts rated it very low explaining that it only works in theory and is not realistic in practice. A prerequisite to this activity are detailed security requirements from the start of the project and adequate security knowledge on the part of the developers. Experts are also divided on the delays caused by this activity (mdn=3.0, IQR=3.50), however they mostly agree that it has severe financial consequences (mdn=1.0, IQR=1.50).

Design factors		Relevance	
Consider the team needs to have special security related knowledge	Added by expert	n/a	
Start early on to avoid bottlenecks when deploying to production	Added by expert	n/a	

---

Quick facts	
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>Gain deep assurance for very sensitive software features or functions which have detailed security requirements</li> </ul>
<b>What to avoid?</b>	

---

Expert ranking	Median	IQR	
<b>Effectiveness</b>	4.5	2.25	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	3.0	3.50	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	1.0	1.50	Higher scores indicate lower financial consequences



## Performing continuous monitoring

Performing continuous monitoring can be applied to a variety of system and application aspects and therefore covers a wide range of detailed activities. Researchers suggest monitoring system-related data [12] but also software vulnerabilities [58], security metrics [12] and security SLA's [53]. It is also recommended to gather security metrics during the SDLC [46] and to maintain an overview of application behaviour [43].

In addition continuous monitoring is cited as relevant in the context of demonstrating compliance with policies [20]. It is important (when automating security controls) to be able to generate evidence on demand that controls are working and that they are effective [46]. To that end, complete coverage of all assets and resources must be ensured.



Metrics gathered through continuous monitoring are preferably be exposed to development and operations teams through so called self-service monitoring and alerting [59]. According to the DevSecOps expert panel this is key in supporting the concept of shifting security to the left" and plays a role in preventing unneeded escalations while allowing teams to become self-organised when it comes to fixing issues.

Continuous monitoring itself should be implemented as code allowing a versioned and repeatable deployment of the required monitoring infrastructure (monitoring-as-code) [59].

Members of the DevSecOps expert panel pointed out that automated deployment is a key enabler to become successful at automated continuous monitoring. Without it the challenge to ensure the correct monitoring configuration is applied to each resource rises.

Frameworks	
<b>OWASP SAMM v2.0</b>	Governance: Strategy & Metrics
<b>BSIMM v10</b>	Governance: Strategy & Metrics

---

Design factors	Relevance
Ensure continuous monitoring covers a wide range of resources and metrics including logical security, availability and intrusions	 9 of 9 experts
Leverage monitoring as code to establish a versioned and repeatable deployment of monitoring infrastructure	 8 of 8 experts

## Performing continuous monitoring of security SLAs

Continuous monitoring of security service level agreements once components are deployed and running allows reaction measures to be taken in case of potential or actual violations [53]. These results can be fed back to development to provide learning opportunities. There are also tools being developed which allow dynamic adaptation of multi-cloud applications to ensure the security status included in security service level agreements is maintained allowing early reaction to possible security incidents. The DevOps team should verify that the metrics defined for the security controls are reaching the target levels thereby ensuring that security and privacy levels of multi-cloud applications are attained [54].

### Expert ranking

The expert panel has not reached agreement on the ranking of this activity. Most experts agree that monitoring the security SLAs makes sense but only if you can do something with this information from a security perspective. Therefore they rate it as moderately effective (mdn=3.0, IQR=2.00). The activity is only seen as effective in organisation with multi-cloud platforms where the capability exists to move resources from one provider to another in case of a security failure. Delays on the DevSecOps process are considered limited assuming the monitoring is automated and running in parallel. The financial consequences are expected to be moderate (mdn=3.5, IQR=1.75).

Quick facts			
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>To leverage the capabilities offered by multi-cloud platforms to react to security failures by moving resources</li> </ul>		
<b>What to avoid?</b>			

---

Expert ranking	Median	IQR	
<b>Effectiveness</b>	3.0	2.00	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	4.0	2.00	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	3.5	1.75	Higher scores indicate lower financial consequences

## Performing continuous monitoring of security metrics throughout the SDLC using CI/CD tooling

Leveraging Continuous Integration and Continuous Deployment (CI/CD) tools to gather security relevant metrics which can be gathered to identify issues such as coding mistakes or vulnerable dependencies [46]. This enables organisations to track threats and vulnerabilities in real-time and allows continuous evaluation of required versus achieved security levels [42].

### Expert ranking

Gathering security metrics by leveraging CI/CD tooling is considered a highly effective security activity by the vast majority of the security experts (mdn=4.0, IQR=1.00) assuming that the metrics are carefully chosen. The delays caused by this activity are considered limited (mdn=4.0, IQR=1.50) as the heavy lifting is mostly done through other activities in the area of continuous monitoring. This activity leverages the outputs generated by other tools to provide indicators of security and knowledge levels in teams. The experts believe there is a moderate financial impact related to this activity which is mainly driven by effort to implement and maintain it (mdn=3.0, IQR=1.50).

Quick facts			
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>To identify security issues in software</li> <li>To measure security knowledge of teams and identify blind spots</li> </ul>		
<b>What to avoid?</b>	<ul style="list-style-type: none"> <li>Collect metrics without a clear strategy</li> </ul>		
Expert ranking		Median	IQR
<b>Effectiveness</b>	4.0	1.00	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	4.0	1.50	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	3.0	1.50	Higher scores indicate lower financial consequences

## Performing continuous monitoring of system metrics using automated tools

Continuously monitoring metrics such as resource usage and reaction times allows potential detection of malicious activity. System-related information such as CPU usage and memory usage can be gathered and stored for further analysis using automated tools [12].

### Design factors

Experts point out that automated deployments and configuration management are required to make effective use of this activity.

### Expert ranking

There is some debate regarding the effectiveness of this activity however the majority of the experts perceives it as being very effective (mdn=4.0, IQR=1.50) without cause a significant delay in the process (mdn=4.0, IQR=1.50). The experts agree that there are moderate financial consequences to be expected mainly driven by efforts to implement and maintain it (mdn=3.0, IQR=0.75).

Design factors		Relevance	
Ensure that other enablers such as automated deployments and configuration management are implemented	<b>Added by expert</b>	n/a	
Quick facts			
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>To gain security knowledge from system behaviour</li> </ul>		
<b>What to avoid?</b>			
Expert ranking		Median	IQR
<b>Effectiveness</b>	4.0	1.50	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	4.0	1.50	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	3.0	0.75	Higher scores indicate lower financial consequences

## Performing continuous monitoring of security controls

This activity refers to implementing specific monitoring solutions to determine the effectiveness of security controls for a given application such as SSL settings, access control mechanisms and vulnerable dependencies. This allows the generation of evidence on demand that controls are working and that they are effective [46].

### Design factors

Experts on the panel pointed out that this activity is especially relevant in highly regulated environments and where formal controls can and should be tracked.

### Expert ranking

The expert panel strongly agrees that performing continuous monitoring of security controls is highly effective (mdn=4.0, IQR=0.50) and agree that it introduces limited delays (mdn=4.0, IQR=1.00). Some debate exists on the financial consequences however a moderate financial consequence, mainly driven by the effort to implement and maintain the solution, is expected (mdn=3.0, IQR=1.50).

Design factors		Relevance	
Ensure that formal controls are tracked	Added by expert	n/a	
Especially important in highly regulated environments	Added by expert	n/a	
Quick facts			
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>To gain insights on the effectiveness of security controls</li> </ul>		
<b>What to avoid?</b>			
Expert ranking		Median	IQR
<b>Effectiveness</b>	4.0	0.50	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	4.0	1.00	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	3.0	1.50	Higher scores indicate lower financial consequences

## Performing continuous monitoring of application behaviour

Continuous monitoring of application behaviour, such as input and output, may assist in detecting malicious activity by identifying changes in patterns [60]. This activity is commonly implemented through tools such as Web Application Firewalls (WAF) or run-time application security protection (RASP).

### Design factors

Experts on the panel point out that the DevOps team needs to have a good understanding of normal application behaviour before attempting to identify deviations.

### Expert ranking

The expert panel strongly agrees on the effectiveness of performing continuous monitoring of application behaviour (mdn=4.0, IQR=0.25). There is some debate on the delay and financial impact caused by this activity however they experts do believe the delay caused will be limited (mdn=4.0, IQR=1.50) while the financial consequences will be considerable (mdn=2.0, IQR=1.50).

Design factors		Relevance	
A DevOps team should be aware of the normal application behaviour allowing identification of deviations	Added by expert	n/a	
Quick facts			
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>To detect abnormal activity in operations</li> </ul>		
<b>What to avoid?</b>			
Expert ranking		Median	IQR
<b>Effectiveness</b>	4.0	0.25	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	4.0	1.50	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	2.0	1.50	Higher scores indicate lower financial consequences

## Provide self-service monitoring capabilities to dev and ops

Providing a flexible monitoring infrastructure allowing teams to configure their monitoring and alerting services according to their criteria allows 'fast and continuous feedback from Ops to Dev [59]. This concept is referred to as 'you build, you run and now you monitor it'. Such a self-service monitoring and alerting solution allows breaking silos between dev, ops and security teams by opening access to key security metrics, enabling a sharing culture and continuous improvement. When looking at the implementation of such a solution one should strive for a large extend of automation leading to the concept of 'monitoring as code'.

### Design factors

Self-service monitoring capabilities should allow teams to be self-organised when it comes to fixing issues. This prevents unnecessary escalations and forms an important enabler for shifting security to the left.

### Expert ranking

Implementing self-service monitoring capabilities is perceived as moderately effective by the experts (mdn=3.0, IQR=1.25). The activity is however not expected to cause significant delays (mdn=4.0, IQR=1.50) nor significant financial impact (mdn=4.0, IQR=1.50).

Design factors	Relevance	
Enable teams to be self-organised when it comes to fix any issues	Added by expert	n/a

---

Quick facts			
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>To establish a sharing culture and allow teams to become self-organised with respect to fixing issues</li> <li>To reduce the number of escalations</li> </ul>		
<b>What to avoid?</b>			

---

Expert ranking	Median	IQR	
<b>Effectiveness</b>	3.0	1.25	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	4.0	1.50	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	4.0	1.50	Higher scores indicate lower financial consequences

## Implement centralised dashboards

This activity was identified through the DevSecOps expert panel and refers to establishing centralised dashboards which provide a clear view on security metrics related to secure development and operations.

### Expert ranking

Building centralised dashboards is considered a moderately effective security activity (mdn=3.5, IQR=1.50) and is expected to cause very limited delays in the DevOps process (mdn=4.0, IQR=1.50). The experts fully agree that this activity is not expected to cause any significant financial consequences (mdn=4.0, IQR=0.00). This is under the assumption that the process of gathering and reporting on the information is fully automated.

Quick facts			
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>To provide visibility into security related aspects to a wide variety of stakeholders</li> </ul>		
<b>What to avoid?</b>			

---

Expert ranking	Median	IQR	
<b>Effectiveness</b>	3.5	1.50	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	4.0	1.50	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	4.0	0.00	Higher scores indicate lower financial consequences

## Implement automated remediation

Dynamic adaptation and reaction to security incidents has been mentioned in the context of multi-cloud environments [53]. The principle is to ensure security levels are maintained across multi-cloud applications by automatically taking required remediation measures on security degradation. Our DevSecOps expert panel indicated that they have never seen it being effective in practice.

### Design factors

Experts point out that manual verification remains important to validate the completeness and appropriateness of automated remediation.

### Expert ranking

The expert panel considers this activity to be moderately effective (mdn=3.0, IQR=1.50). Some experts point out that getting automated remediation up and running requires attention to potential breaking changes and that effort is potentially better spent in preventing the issue from occurring. Automated remediation can only become effective when many other aspects are fully automated. Due to the emphasis on automation it is not expected that this activity will cause significant delays in the DevSecOps process (mdn=5.0, IQR=0.50) however it can represent a moderate financial impact (mdn=3.0, IQR=1.50) driven by licensing costs and effort to implement and maintain.

Frameworks			
<b>OWASP SAMM v2.0</b>	Operations: Incident management		
<b>BSIMM v10</b>	Deployment: Configuration management & Vulnerability management		
Design factors			
Always manually verify the results from time to time	<p style="text-align: right;"><b>Relevance</b></p> <p style="text-align: right;"><b>Added by expert</b>    n/a</p>		
Quick facts			
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>To leverage existing automated capabilities to automatically react to security events</li> </ul>		
<b>What to avoid?</b>	<ul style="list-style-type: none"> <li>Solely rely on automated remediation without verifying its effectiveness</li> <li>Attempt to implement automated remediation in environments with a low DevSecOps maturity</li> </ul>		
Expert ranking			
	Median	IQR	
<b>Effectiveness</b>	3.0	1.50	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	5.0	0.50	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	3.0	1.50	Higher scores indicate lower financial consequences

## Performing security configuration automation

Security configuration automation is mentioned several times throughout academic literature for example in the context of automating software defined firewalls to allow the deployment of consistent policies across firewalls in an organisation [12]. Security configuration automation is also referred to as security-as-code allowing the definition of security policies which can be embedded and enforced throughout the development and operations processes from the project get-go. Such codified security policies can be activated automatically or manually and stored in a central repository for reuse on new project [46].

### Expert ranking

Performing security configuration automation is seen as a effective approach (mdn=4.0, IQR=0.25) while causing minimal delays (mdn=4.0, IQR=1.50). The concepts behind configuration automation and potential best practices are proven, well known and can easily be integrated in existing solutions such as system configuration automation tooling. There is debate regarding the financial consequences, mainly driven by the effort to implement and maintain the solution. A moderate financial impact should be expected (mdn=3.0, IQR=2.00).

Frameworks			
<b>OWASP SAMM v2.0</b>	Implementation: Secure build		
<b>BSIMM v10</b>	Deployment: Configuration management & Vulnerability management		
Quick facts			
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>To consistently manage security configurations across a wide range and large volume of devices</li> </ul>		
<b>What to avoid?</b>			
Expert ranking		Median	IQR
<b>Effectiveness</b>	4.0	0.25	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	4.0	1.50	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	3.0	2.00	Higher scores indicate lower financial consequences

## Implement secrets management

This activity was identified through the DevSecOps expert panel and refers to automating the management of secrets used by applications and infrastructure components.

### Expert ranking

Implementing secrets management is perceived as an effective security activity, with half of the experts (N=5, 50%) rating is as highly effective. They do point out that it may sometimes become complicated or even theoretically impossible to secure all secrets in an environment. It is important to pay appropriate attention to ensure the availability of this system and to implement a proper authorisation model to increase its effectiveness. This activity It is not expected to cause significant delays (mdn=4.0, IQR=1.50) in the DevSecOps process however, depending on the choice of product, is expected to have some financial consequences (mdn=3.5, 1.75).

Frameworks			
<b>OWASP SAMM v2.0</b>	Implementation: Secure build		
<b>BSIMM v10</b>	Deployment: Configuration management & Vulnerability management		
Quick facts			
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>To enable organisation wide protection of secrets</li> <li>To establish control over secrets such as authorisation and traceability</li> </ul>		
<b>What to avoid?</b>	<ul style="list-style-type: none"> <li>Do not forget to pay special attention to the availability of this system as it is a critical component</li> </ul>		
Expert ranking		Median	IQR
<b>Effectiveness</b>	4.5	2.00	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	4.0	1.50	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	3.5	1.75	Higher scores indicate lower financial consequences

## Manage digital supply chain

During this research activities were identified which relate to managing the risks originating from using external components in the software development process. These components are not under the direct control of the organisation and any weaknesses in them may significantly impact the security of the systems build on top of them. Therefore the choice was made to group these activities under the umbrella of 'managing the digital supply chain'.

### Frameworks

<b>OWASP SAMM v2.0</b>	Implementation: Secure build
<b>BSIMM v10</b>	Deployment: Configuration management & Vulnerability management

## Establish artefact and source code registries which are automatically scanned for vulnerabilities

This activity was added by a member of the DevSecOps expert panel and refers to the actions taken to continuously scan and approve entries in artefact and source code registries thereby preventing the use of vulnerable resources in development or operations.

### Expert ranking

Experts perceive this activity to be very effective (mdn=4.0, IQR=1.00) and is expected to cause moderate delays (mdn=3.0, IQR=1.50) specifically if manual verifications are required. There are no significant financial consequences anticipated (mdn=4.0, IQR=1.50) beyond some licensing costs for the tooling.

### Quick facts

<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>Prevent developers from leveraging vulnerable components in their applications</li> </ul>
<b>What to avoid?</b>	<ul style="list-style-type: none"> <li>Manual verifications</li> </ul>

Expert ranking	Median	IQR	
<b>Effectiveness</b>	4.0	1.00	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	3.0	1.50	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	4.0	1.50	Higher scores indicate lower financial consequences

## Implement automated container security scanning

This activity was added by a member of the DevSecOps expert panel and refers to implementing automated security scanning of container images and configuration for known vulnerabilities and weaknesses preventing vulnerable images or insecure configurations to be used as a basis for development or operations.

### Expert ranking

Automated container security scanning has been rated as a very effective security activity by the expert panel (mdn=4.0, IQR=1.25). However experts do point out that container scanning is only a part of the puzzle. In principle container scans are very fast to execute, not causing significant delays (mdn=4.0, IQR=0.00). Some financial consequences are anticipated (mdn=3.5, IQR=1.00).

### Quick facts

<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>To detect vulnerabilities or vulnerable configurations in container images</li> </ul>
<b>What to avoid?</b>	

Expert ranking	Median	IQR	
<b>Effectiveness</b>	4.0	1.25	Higher scores indicate a higher level of effectiveness
<b>Impact on delay</b>	4.0	0.00	Higher scores indicate a lower friction or delay
<b>Financial consequences</b>	3.5	1.00	Higher scores indicate lower financial consequences

## Performing automated software composition analysis

Automated software composition analysis is a technique where the third party libraries and code on which an application depend are verified for known vulnerabilities [20]. This is achieved by comparing the components and their versions against a database of known vulnerabilities. It is generally accepted that the eco system of dependencies on which applications are build may introduce significant risk to an application over the course of its entire lifespan. Therefore automated software composition analysis should be performed continuously over time and not only during the development cycle. This activity is closely related to the concept of controlling open source risks [43].

### Design factors

Several experts on the DevSecOps panel mentioned that ensure that only approved libraries can be used during development time is an approach to reduce the problem at the source. However this does not provide coverage for vulnerabilities which are detected while an application is active in a production environment and not subject to development activities. A process to handle new vulnerabilities is required to ensure proper prioritisation and sign-off. Experts recommend to start with the most critical applications first as the potential fall-out may be significant.

### Expert ranking

Performing automated software composition analysis is a moderately effective security activity (mdn=3.0, IQR=1.25), mainly due to the potential overload of information. The technique is very capable of findings security weaknesses however verification of the applicability to the context of the application is required and may often indicate that the vulnerable function in the dependency is not actually used in the context of the application. It may introduce a moderate delay in the DevSecOps process (mdn=3.0, IQR=1.00) if applications have many dependencies and a full scan is expected on every build. A moderate financial impact is expected (mdn=3.0, IQR=1.50) mainly due to effort related to verifying false positives and placing results in the context of the application. Treating the identified issues late in the development or operations process may however represent considerable effort.

Design factors		Relevance	
(Make use of) Approved libraries	Added by expert	n/a	
Start with the most critical (end-user facing) applications first.	Added by expert	n/a	
Establishing a process to handle new vulnerabilities is important.	Added by expert	n/a	
Quick facts			
<b>Why should you do this?</b>	<ul style="list-style-type: none"> <li>To detect vulnerabilities in libraries and dependencies used by software</li> </ul>		
<b>What to avoid?</b>	<ul style="list-style-type: none"> <li>Requiring a full scan of all dependencies on every build in continuous deployment environments</li> </ul>		
Expert ranking		Median	IQR
<b>Effectiveness</b>		3.0	1.25
<b>Impact on delay</b>		3.0	1.00
<b>Financial consequences</b>		3.0	1.50
			Higher scores indicate a higher level of effectiveness
			Higher scores indicate a lower friction or delay
			Higher scores indicate lower financial consequences



*“treating security as a “bolt-on” to the end of the process is far costlier and can damage the relationship between security and development teams”*

*Francois Raynaud, 2017 [44]*



---

# Part four: Results

---

This part of the paper provides an overview of the results obtained during this research project. It consists of the following sections:

- **Answers to research questions:** presents the answers to the research questions as initially proposed in the first part of this paper
- **Conclusion:** presents the conclusions drafted based on the outcome of this research
- **Limitations:** presents the limitations to which this research was subject
- **Future research:** proposes topics for future work

## Answers to research questions

At the start of this research project three research questions were formulated to provide an answer to the challenge on how to integrate security assurance activities in DevOps without creating friction. As a foundation for this research the first question aimed to provide an agreed definition on DevOps and DevSecOps. The proposed definitions were derived from academic papers [10][22] and presented to a panel of 10 DevSecOps experts out of which 7 agreed on the proposed definitions. The details can be found in table 7 (Results on the validation of the proposed definitions for DevOps and DevSecOps). These results allow us to answer research questions 1a and 1b as follows:

### RQ1a: What is the definition of DevOps?

DevOps is a development methodology aimed at bridging the gap between Development and Operations, emphasising communication and collaboration, continuous integration, quality assurance and delivery with automated deployment utilising a set of development practices.

### RQ1b: What is the definition of DevSecOps?

RuggedOps (DevSecOps): “Rugged” describes software development organisations that have a culture of rapidly evolving their ability to create available, survivable, defensible, secure, and resilient software.

Subsequently this research set out to identify a set of security activities and design factors which are relevant in the context of DevSecOps. This set of relevant security activities and design factors was

established through a literature review followed by an expert validation and elaboration session involving 10 DevSecOps experts. This resulted in a set of 29 validated security activities on which there is strong agreement between the experts on their relevance to DevSecOps. This allowed us to answer the second research question as follows:

**RQ2: Which set of security activities and design factors relevant to DevOps processes can be distinguished from academic literature?**

A total of 31 security activities (table 8) and 34 design factors (table 9) identified through literature research are considered relevant in the context of security for DevOps. During the validation and elaboration the expert panel indicated one activity to be irrelevant to DevSecOps (-1) and added 3 additional activities (+3) bringing the total on 33 security activities. The expert panel also added an additional 53 design factors bringing the total number of design factors to 87.

The third and final research question for this research project aims to rank the identified security activities based on effectiveness and delay caused in the process of continuous deployment by a group of practitioners. The results of the previous research question were prioritised by a group of 8 DevSecOps experts during a Group Support Session (GSS), the results of which can be consulted in table 10 (Results of the prioritisation of security activities by the expert panel) allowing us to answer research question 3 (RQ3) as follows:

**RQ3: How do the identified security practices and activities rank in terms of effectiveness and delay from a practitioner point of view?**

The security activities, grouped into the categories of collaboration, use of automated activities and use of non-automated activities, have been ranked by the experts as presented in table 11 (prioritised list of security activities) according to the following three aspects:

**Relevance:** an indication of the number of experts who perceived the activity as relevant to DevSecOps. This score was obtained during the second phase of the process

**Effectiveness:** an indication of the degree to which the expert panel believes the activity is contributing to the security of the software under development

**Delay:** an indication of the degree to which the expert panel believes the activity causes delays in the development or operations process

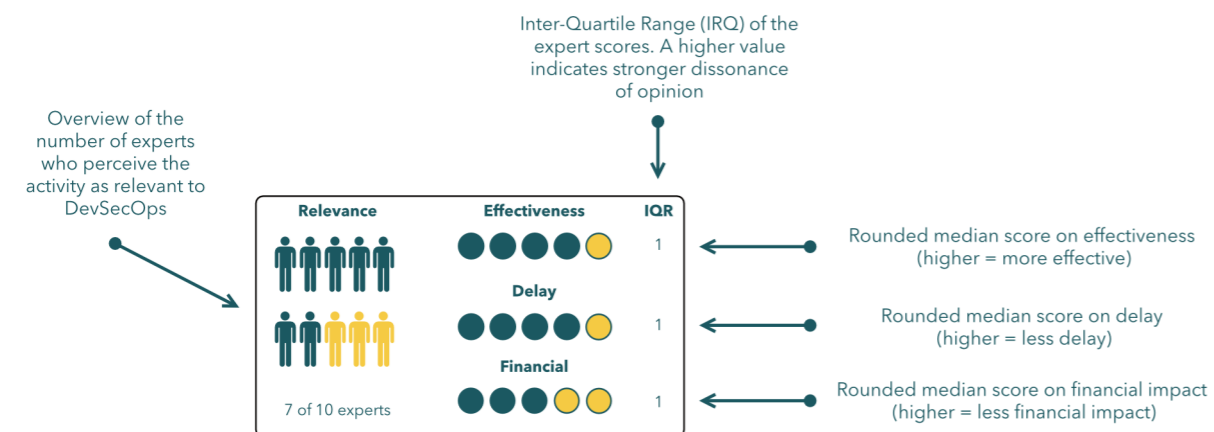


Table 11: Prioritised list of security activities

Collaboration			Relevance	Effectiveness	IQR
Performing continuous feedback from production to development	This activity refers to continuously feeding security metrics and information on security incidents from production back to development.	9 of 10 experts	●●●●●●●●●●	●●●●●●●●●● Delay ●●●●●●●●●● Financial	1.25 1.50 1.00
Provide security training	This activity refers to training a wide range of stakeholders such as developers, architects and product owners on security aspects.	9 of 10 experts	●●●●●●●●●●	●●●●●●●●●● Delay ●●●●●●●●●● Financial	1.00 1.50 0.50
Establish security satellites	This activity refers to creating a network of security savvy people throughout the various teams involved in software development. These people are regularly referred to as security champions.	10 of 10 experts	●●●●●●●●●●	●●●●●●●●●● Delay ●●●●●●●●●● Financial	1.00 0.50 1.50
Practice incident response	This activity refers to practicing incident response through red-team exercises and security drills.	7 of 10 experts	●●●●●●●●●●	●●●●●●●●●● Delay ●●●●●●●●●● Financial	1.25 1.50 1.50
Establish a security mindset across the organisation	This activity refers to actions taken to increase the attention and awareness on security related aspects of secure development and operations across both operational and managerial levels of an organisation.	added by expert	●●●●●●●●●●	●●●●●●●●●● Delay ●●●●●●●●●● Financial	1.25 1.50 2.00
Use of automated activities			Relevance	Effectiveness	IQR
Performing automated security testing	This activity refers to aspects of security testing which can be automated thereby providing actionable information.	10 of 10 experts	●●●●●●●●●●	●●●●●●●●●●	1.25
Performing automated run-time testing	This activity refers to the dynamic, interactive testing of a deployed application using automated tools (DAST)	8 of 10 experts	●●●●●●●●●●	●●●●●●●●●● Delay ●●●●●●●●●● Financial	1.25 0.50 1.50
Performing automated static testing	This activity refers to code review using automat-ed tools to detect common vulnerability patterns (SAST).	8 of 10 experts	●●●●●●●●●●	●●●●●●●●●● Delay ●●●●●●●●●● Financial	2.00 1.50 1.00
Integrate security tests in unit testing	This activity refers to leveraging unit testing to perform security-oriented tests such as boundary testing.	9 of 10 experts	●●●●●●●●●●	●●●●●●●●●● Delay ●●●●●●●●●● Financial	2.25 3.50 1.50

Table 11: (Continued.) Prioritised list of security activities

Use of automated activities (continued)		
Performing continuous monitoring	This activity refers to actions enabling a continuous view on various security aspects of development and operations activities.	<p>Relevance: 9 of 10 experts</p> <p>Effectiveness: 2.00</p> <p>Delay: 2.00</p> <p>Financial: 1.75</p>
Performing continuous monitoring of security SLAs	This activity refers to performing continuous monitoring to confirm compliance with security service level agreements for cloud providers.	<p>Relevance: 8 of 9 experts</p> <p>Effectiveness: 2.00</p> <p>Delay: 2.00</p> <p>Financial: 1.75</p>
Performing continuous monitoring of security metrics throughout the SDLC using CI/CD tooling	This activity refers to leveraging the Continuous Integration and Continuous Deployment tools to gather security relevant metrics which can be monitored to identify risks such as coding mistakes or vulnerable dependencies.	<p>Relevance: 7 of 9 experts</p> <p>Effectiveness: 1.00</p> <p>Delay: 1.50</p> <p>Financial: 1.00</p>
Performing continuous monitoring of system metrics using automated tools	This activity refers to continuously monitoring metrics such as resource usage and reaction times. Based on patterns in these metrics malicious activity could potentially be detected.	<p>Relevance: 5 of 9 experts</p> <p>Effectiveness: 1.50</p> <p>Delay: 1.50</p> <p>Financial: 0.75</p>
Performing continuous monitoring of security controls	This activity refers to implementing specific monitoring solutions to determine the effectiveness of security controls for a given application such as SSL settings, access control mechanisms and vulnerable dependencies.	<p>Relevance: 8 of 9 experts</p> <p>Effectiveness: 0.50</p> <p>Delay: 1.00</p> <p>Financial: 1.50</p>
Performing continuous monitoring of application behaviour	This activity refers to continuous monitoring of application behaviour such as input and output to determine changes in patterns which may indicate malicious activity. This activity is commonly implemented through tools such as Web Application Firewalls (WAF).	<p>Relevance: 8 of 9 experts</p> <p>Effectiveness: 0.25</p> <p>Delay: 1.50</p> <p>Financial: 1.50</p>
Provide self-service monitoring capabilities to dev and ops	This activity refers to building monitoring capabilities so that they allow dev and ops to define the collection of metrics, definition of thresholds and alerts themselves making it a shared responsibility.	<p>Relevance: 8 of 9 experts</p> <p>Effectiveness: 1.25</p> <p>Delay: 1.50</p> <p>Financial: 1.50</p>
Implement centralised dashboards	This activity refers to establishing centralised dashboard which provide a clear view on security metrics related to secure development and operations.	<p>Relevance: added by expert</p> <p>Effectiveness: 1.50</p> <p>Delay: 1.50</p> <p>Financial: 0.00</p>

Practice	Activity	L2 Activity
A security practice is a collection of activities can be grouped based on existing similarities within activities	A security activity focuses on achieving a small, well-defined goal that has a tangible output.	Detailed security activity which is derived from L1 activities.

Table 11: (Continued.) Prioritised list of security activities

Use of automated activities (continued)		
Implement automated remediation	This activity refers to actions enabling a continuous view on various security aspects of development and operations activities.	<p>Relevance: 10 of 10 experts</p> <p>Effectiveness: 1.50</p> <p>Delay: 0.50</p> <p>Financial: 1.50</p>
Performing security configuration automation	This activity refers to automating security configurations (hardening) throughout the lifecycle of an environment.	<p>Relevance: 9 of 10 experts</p> <p>Effectiveness: 0.25</p> <p>Delay: 1.50</p> <p>Financial: 2.00</p>
Implement secrets management	This activity refers to automating the management of secrets used by applications and infrastructure components.	<p>Relevance: added by expert</p> <p>Effectiveness: 2.00</p> <p>Delay: 1.50</p> <p>Financial: 1.75</p>
Manage digital supply chain	This activity refers to actions taken to manage the risks introduced through the use of external components which are not under the direct control of the organisation.	<p>Relevance: added by expert</p>
Establish artefact and source code registries which are automatically scanned for vulnerabilities	This activity refers to the actions taken to continuously scan and approve entries in artefact and source code registries thereby preventing the use of vulnerable resources in development or operations.	<p>Relevance: added by expert</p> <p>Effectiveness: 1.00</p> <p>Delay: 1.50</p> <p>Financial: 1.50</p>
Implement automated container security scanning	This activity refers to implementing automated security scanning of container images and configuration for known vulnerabilities and weaknesses preventing vulnerable images or insecure configurations to be used as a basis for development or operations.	<p>Relevance: added by expert</p> <p>Effectiveness: 1.25</p> <p>Delay: 0.00</p> <p>Financial: 1.00</p>
Performing automated software composition analysis	This activity refers to the verification of all dependencies (e.g. third-party libraries) for known vulnerabilities (SCA)	<p>Relevance: 8 of 10 experts</p> <p>Effectiveness: 1.25</p> <p>Delay: 1.00</p> <p>Financial: 1.50</p>

Table 11: (Continued.) Prioritised list of security activities

Use of non-automated activities		Relevance	Effectiveness	IQR
Performing security requirements analysis	This activity refers to the definition of security requirements.	9 of 10 experts	Effectiveness: 2.00 Delay: 1.50 Financial: 2.00	
Performing threat modeling	This activity refers to performing threat modeling to establish a common model of an application and subsequently identifying potential threats.	10 of 10 experts	Effectiveness: 0.50 Delay: 1.00 Financial: 0.25	
Performing risk analysis	This activity refers to analysing the threats to an application in the context of the business impact and likelihood to establish a risk score.	10 of 10 experts	Effectiveness: 0.50 Delay: 1.00 Financial: 0.25	
Establishing security SLA's for cloud providers	This activity refers to establishing security service level agreements for cloud providers based on the security requirements of a given application.	10 of 10 experts	Effectiveness: 2.00 Delay: 2.25 Financial: 1.00	
Performing continuous assurance	This activity refers to continuously validating if the software of interest is compliant with relevant regulatory requirements.	9 of 10 experts	Effectiveness: 2.00 Delay: 2.00 Financial: 0.25	
Performing manual security testing	This activity refers to aspects of security testing activities which cannot be automated and need to be performed manually.	8 of 10 experts		
Performing manual penetration testing	This activity refers to performing manual penetration tests.	8 of 8 experts	Effectiveness: 1.50 Delay: 1.50 Financial: 1.25	
Performing manual security review	This activity refers to performing manual security reviews which is usually a combination of manual code analysis combined with documentation review and stakeholder interviews.	8 of 8 experts	Effectiveness: 1.50 Delay: 1.00 Financial: 1.25	
Secure the Ci/Cd pipeline	This activity refers to using a set of tools to prevent any changes such as access control or build script definitions which may impact the security of the CI/CD pipeline.	8 of 10 experts	Effectiveness: 1.50 Delay: 1.25 Financial: 1.25	

## Conclusion

The premise of this research project revolves around the concept that DevOps and security assurance can reinforce each other when implemented appropriately. Integrating security assurance activities enables organisations to gain confidence in their security posture [62], improve security characteristics [12] and may remove an obstacle for the adoption of DevOps in an organisation [16][17]. It also allows organisations to increased speed of response to security issues [62] and open the path to leverage concepts such as quality culture and automation for the benefit of security [62]. However this integration must be performed appropriately to avoid the introduction of significant delays [14] which could potentially prohibit the materialisation of increased organisational performance [2].

Therefore this research set out to define a framework of relevant security activities enabling an organisation to integrate security assurance in DevOps with a clear view on the effectiveness but also the delay caused to the process of continuous delivery.

Based on the results of this research project, as represented in table 11, we conclude that for the most part traditional security assurance activities remain relevant in DevOps. There was a strong agreement among the security experts that the identified activities remained relevant in the context of DevSecOps. Furthermore the activities, while having varying rating in terms of effectiveness and financial impact, display a similar level of delay.

This leads us to the conclusion that in order to avoid introducing delays in DevSecOps it is not so much about doing different things as it is about doing things differently. These differences are visible in the design factors gathered throughout this research and represented in part three of this paper. Overall the following tenant expresses the difference in approach quite eloquently:

*“We prefer automated over manual, repeatable over one-off.”*

Some of the trade-offs which are made in DevSecOps are counterintuitive and radically different from traditional approaches to information security. Some examples of these are:

- Reduce the detection rate in favour of reducing the number of false positives;
- Prefer short iterative threat modelling exercises at the beginning of each sprint at the expense of threat model coverage;
- Prefer limiting the available libraries during development to pre-approved versions over pre-production validations;
- Foster engineering spirit and peer review over quality gates with validation steps;
- Continuously feed information from production to development
- Parallelise slower processes and invest in blue green deployments to roll back when needed.

These trade-offs and differences in approaches are translated into design factors for each of the security activities and presented in part three of this paper.

All of the involved security experts strongly emphasised the aspects of collaboration and knowledge for people working in DevSecOps teams. One could say that this is an instantiation of solid engineering practices, in essence DevSecOps is about teams of engineers who gather around a problem in an attempt to understand and solve it through mutually agreed approaches. It is built on the concept of combining disciplines and creating a learning culture. The main theme in DevSecOps is the creation of feedback loops for learning purposes and ensuring you have the right measurements in place to get the most learnings out of each investment thereby gradually finding out what works and what doesn't, while taking into account the relevant security aspects.

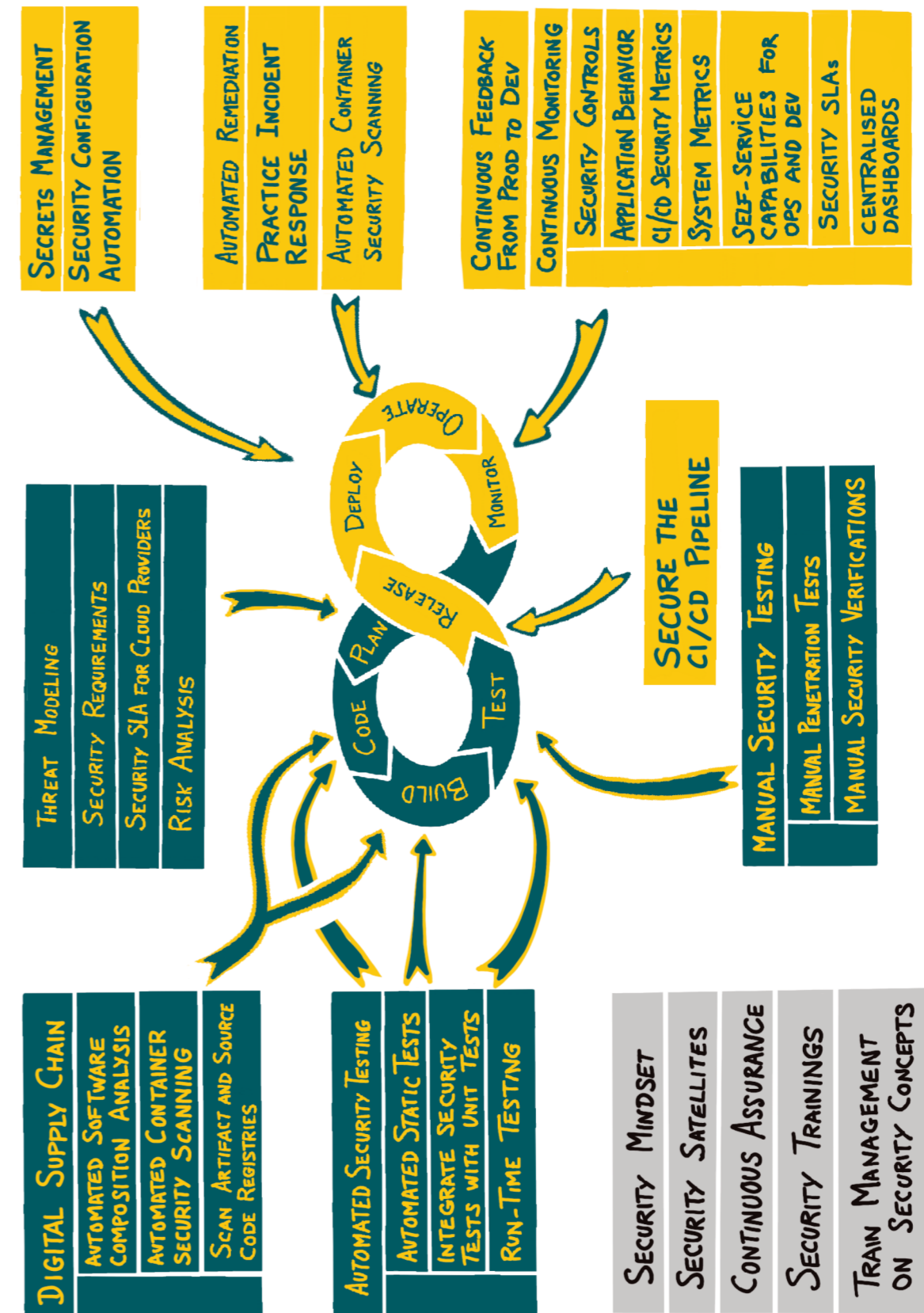
Overall the following tenants seem to hold true:

- Share security learning experiences and create a security engineering mindset;
- Shift security responsibility to the teams and create supporting mechanisms to get the job done;
- Leverage security automation whenever possible;
- Establish security measurements to gain insight and learning opportunities;
- When selecting activities and approaches one should favour reducing delay over reducing financial impact

The identified set of security activities, their ranking in terms of effectiveness, delay and financial impact and the in-depth design factors presented in this paper can be leveraged by organisations to model their DevSecOps approach to include security activities based on their characteristics in terms of effectiveness, delay and financial consequences. The design factors represented in this paper provide insights in suitable implementations of these activities. The activities and design factors combined can be used to establish a roadmap for integrating security activities in a DevSecOps environment and as a benchmarking tool to assess the existing implementation.

As a closing summary a map of DevSecOps activities as identified during this research linked to the various stages of the DevSecOps process is presented in figure 11.

Figure 11: Map of DevSecOps activities identified during this research



---

## Limitations

During this research project some limitations were encountered which may influence the validity of the outcomes. They are listed here for your reference:

- The literature review performed in the scope of this research project focused on relatively narrow search criteria to keep the number of identified papers within a workable limit given the time available for this study. Therefore important contributions in the field of agile security and DevSecOps may be absent in the dataset used for this research;
- The grouping of security activities and design factors was performed by the researcher thereby potentially introducing bias. This threat has been partially reduced by performing a validation step with an expert panel;
- A mistake was made during the creation of the survey leading to the design factors for risk analysis not being validated by the expert panel;
- The results of the elaboration by the expert panel were interpreted by the researcher to extend the list of identified security activities and design factors. The comments were interpreted by the researcher and presented to the experts for verification however this was not done through a formal survey due to time limitations and the risk of “survey fatigue” on the respondents part;
- The prioritisation performed by the expert panel was based on short descriptions of the security activities, differences in interpretation between the members of the expert panel may have remained unnoticed and may have influenced the scoring;
- The mapping with OWASP SAMM and BSIMM and the mapping with the various stages of the DevSecOps process as presented in this paper were not validated by an expert panel;
- The analysis of the prioritisation by the expert panel was performed using fairly simple methods of descriptive statistics, there was no complete SPSS analysis performed;
- The artefact answering research question 3 was not evaluated in real world environments therefore the design cycle was not completed.

---

## Future research

Various avenues for future activities were identified during this research project. The framework of security activities relevant for DevSecOps, developed as an answer to the third research question, should be tested in real world situations to measure and assess actual usability and to perform improvements to the framework. This is left as a potential avenue for future research.

Each of the identified security activities identified during this project represent a potential subject for future research. Such research could focus on establishing complete inventory of design factors and measure their influence on effectiveness, delay and financial consequences.

In addition an overview of the tooling landscape for each of the identified security activities could be created to increase the real-world usability of the framework.

A final approach to extend the framework would be to create an extensive mapping between existing standards such as the Agile Secure Software Framework (SSA), OWASP SAMM and BSIMM but also to standards such as ISO27000, NIST CyberSecurity Framework or CIS controls.

---

---

# Appendices

---

This section contains additional information for your reference.

# Appendix A: Literature review notes

## Security Assurance in DevOps methodologies and related environments

(Siewruk et al., 2019b)

This study focuses on the challenges introduced by high speed software development coupled with cloud capabilities from a security perspective. It states that a good set of practices and tools is lacking to include IT security issues into the whole production and deployment processes. The paper describes a proposed approach for large mobile telecommunication operator environments.

The main focus of this paper is on the system architecture to allow an IT security evaluation of the cloud environment. It tackles this problem from a threat management perspective at a system component level. The paper refers to Continuous Integration and Continuous Deployment tooling which allow task automation and infrastructure as code using tools such as ansible or terraform which perform automated software provisioning and configuration management. The paper continues to propose an approach to calculate security level metrics of the solutions deployed in the cloud environment to allow an overall security level to be calculated at a system level.

This research has a focus on assessing the security level of a full stack deployment (network, infrastructure, application) in cloud environments. Additional attention will be required when incorporating the results in this study when identifying practices, activities or design factors for DevSecOps across the complete dataset. Some of the proposed approaches in this study may refer to mechanisms specific to cloud environments.

## Dynamic Security Assurance in Multi-Cloud DevOps

(Rios, Iturbe, Mallouli, et al., 2017b)

This study focuses on the challenges introduced by development and operations of applications which combine resources across a range of cloud providers. The researchers state that the challenge of addressing security assurance in such heterogenous applications is challenges and currently not addressed by the state of the art. The researchers analyse the use of the MUSA framework developed through EU funded research and propose an approach to Dynamic Security Assurance in multi-cloud DevOps.

This research has a focus on identifying security requirements and assessing security levels when combining resources across clouds from various providers. Additional attention will be required when incorporating the results in this study when identifying practices, activities or design factors for DevSecOps across the complete dataset. Some of the proposed approaches in this study may refer to mechanisms specific to cloud environments.

## Security practices in DevOps & Software Security in DevOps: Synthesising Practitioners' Perceptions and Practices

(A. A. U. Rahman & Williams, 2016a) (A. A. U. Rahman & Williams, 2016c)

We identified two publications related to the same research during the structured search. The paper titled "Security practices in DevOps" provides a short summary of the results of research described in the paper titled "Software Security in DevOps: Synthesising Practitioners' Perceptions and Practices". The latter publication was identified as a landmark paper partially influencing the keywords used during the structured search and providing a starting point for the set of initial codes.

The research revolves around the analysis of a large number of internet artefacts such as blog posts and video presentations. The researchers identified a set of security practices and activities used to integrate security in DevOps and developed a survey which ran across nine organisations for further investigation.

The primary focus of this study is directly aligned with our research and therefore is expected to yield results which are easy to integrate in the identification of practices, activities or design factors for DevSecOps (identification of themes).

## DevSecOps: A Multi-vocal Literature Review

(Myrbakken & Colomo-Palacios, 2017b)

The scope of this study is to provide an overview of how to implement DevSecOps and aims to list the benefits gained from DevSecOps but also to provide insight in the challenges. The researchers believe that implementing security in a way so that it can keep up with DevOps is a challenge but provides great benefits. During the study the researchers performed a review of all accessible literature on the topic of DevSecOps (multi-vocal literature review). The result of this review is a summary of the challenges and the characteristics related to DevSecOps.

The primary focus of this study is directly aligned with our research and therefore is expected to yield results which are easy to integrate in the identification of practices, activities or design factors for DevSecOps (identification of themes).

## SecDevOps: Is It a Marketing Buzzword?

(Mohan & Othmane, 2016b)

This paper surveys the literature from academia and industry to identify the main aspects of DevSecOps. The researchers aim to identify the prevailing definition of SecDevOps as a concept and identify which aspects are commonly included when defining SecDevOps. The main aspects they found are definition, security best practices, compliance, process automation, tools for SecDevOps, software configuration, team collaboration, availability of activity data and information secrecy.



---

The primary focus of this study is directly aligned with our research and therefore is expected to yield results which are easy to integrate in the identification of practices, activities or design factors for DevSecOps (identification of themes).

### **DevOps for Better Software Security in the Cloud**

(Jaatun et al., 2017b)

This paper focuses on demonstrating how DevOps can lead to better software security in the cloud. The researchers describe the changes introduced by DevOps and cloud technologies which ease development, deployment and operations of software. They state that the major problem with software security is that it is impossible to know all attacks that the system will be exposed to. As such they argue that the increased capability to modify the software can be used to quickly respond to security issues.

The primary focus of this study is directly aligned with our research and therefore is expected to yield results which are easy to integrate in the identification of practices, activities or design factors for DevSecOps (identification of themes).

### **Software Security Activities that Support Incident Management in Secure DevOps**

(Jaatun, 2018b)

The focus of this research is to identify how incident management and software development can be beneficial to ensure there is a second level of defence for security vulnerabilities introduced during the DevOps cycle. The paper outlines the characteristics of computer security incident management based on ISO/IEC 27035 and argue that the need for collaboration between incident response teams and development teams have an increased as a result of the shorter development cycles resulting from DevOps. The researchers performed an analysis of the 113 BSIMM activities and identified those which are in their view directly relevant to incident response in a DevOps setting. These activities were subsequently mapped to ISO/IEC 27035.

This research has a focus on DevSecOps with a specific focus on incident management. Attention will be required when incorporating the results in this study when identifying practices, activities or design factors for DevSecOps across the complete dataset.

### **Self-Service Cybersecurity Monitoring as Enabler for DevSecOps**

(Diaz et al., 2019)

This paper investigates the potential for self-service security monitoring to strengthen DevSecOps in the development and operations of IoT systems. The premise of the authors is that the distributed and heterogeneous nature of IoT solutions leads to complex development and deployment

pipelines. This complexity is expected to have a negative effect on cybersecurity as a whole and event monitoring specifically. The researchers believe that self-service cybersecurity monitoring can act as an enabler to security practices in a DevOps environment by allowing 'Fast and Continuous Feedback from Ops to Dev'. The researchers therefore extend a commonly used concept: 'You build, you run, and now you monitor. Based on a case study of the proposed solution the authors demonstrate that feeding back detection of threats in the operational environment facilitates the developer to make more informed decisions to fix problems or support software evolution.

This research has a focus on DevSecOps with a specific focus on self-service security monitoring. Attention will be required when incorporating the results in this study with regards to identifying practices, activities or design factors for DevSecOps as a whole.

### **Francois Raynaud on DevSecOps**

(Carter, 2017b)

In this article a security professional experienced in DevSecOps shares his view on the need for a DevSecOps approach and what such an approach could look like.

The primary focus of this study is directly aligned with our research and therefore is expected to yield results which are easy to integrate in the identification of practices, activities or design factors for DevSecOps (identification of themes).

### **A systematic mapping study of infrastructure as code research**

(Rahman et al., 2019)

This research aims to provide insight in the potential areas for future research related to infrastructure as code through a systematic mapping study. Due to the nature of this study it did not provide any contribution to our research questions and was therefore excluded from the dataset.

### **Threat analysis of software systems: A systematic literature review**

(Tuma, Calikli, Ariato, 2018)

This research consisted of a systematic literature review on the existing techniques for threat analysis. A comparison of 26 methodologies for threat analysis is performed with the objective to providing an overview of these methodologies in terms of required input, procedural characteristics, outcome characteristics and ease of adoption. As such this overview can be used by practitioners to select an appropriate threat analysis methodology for a given context.

An aid in performing an appropriate selection in terms of threat analysis methodology for DevOps is provided by the researchers through their mapping to appropriate abstraction levels for the various methodologies. An assessment was performed for each technique in terms of applicability at the level of requirements, architecture, design and implementation.

This paper goes in depth on one of the activities relevant to DevSecOps and therefore provides useful contributions in the form of design factors for the activity of threat analysis. Attention should be paid as the results of this study are not specific to the context of DevSecOps which may lead to the inclusion of design factors which are not relevant for a high-speed iterative approach. The authors propose themselves a reflection on the topic of using threat analysis in environment driven by change:

To conclude, in light of DevOps and Agile, where software development is driven by change, there are three important aspect where existing analysis techniques have yet to mature: (i) traceability of analysis in the code base, (ii) composability of analysis outcomes and (iii) threat impact analysis automation.

#### **DevOps in practice: A multiple case study of five companies**

(Lwakatare et al., 2019)

This paper focuses on providing descriptions of how DevOps is implemented in practice with a specific focus on web application and service development in small and medium sized organisations. To this end the researchers performed a multiple-case study resulting in an overview of the concept, related practices and impact of DevOps in this context.

The researchers identified that security is a practice which is of importance to both development and operations personnel and therefore is also a factor when looking at a deployment pipeline. They point out that securing the pipeline itself from malicious attackers and scenarios and ensuring security conformance from a security audit perspective is an important factor to take into account.

This research has no clear focus on DevOps and as such does not describe specific practices or activities relevant for DevSecOps. This artefact was therefore excluded from our gold set.

#### **Service level agreement-based GDPR compliance and security assurance in (multi)Cloud-based systems**

(Rios, Iturbe, Larrucea, et al., 2019b)

This research focuses on the challenges related to GDPR compliance in Cloud-based systems. The aim is to define a DevOps framework aimed at supporting Cloud consumers in designing, deploying and operating Cloud systems that include necessary privacy and security controls. The emphasis is placed on the definition of security specifications to be included in the system service level agreement and continuous monitoring and enforcement.

This research has a strong focus on the approach to performing security requirements in the context of a cloud environment from a GDPR compliance perspective. Upon reading the paper it is clear that due to this focus most of the proposed approaches are not specific to Dev(Sec)Ops. Care must be taken when analysing the content of this paper in the context of our research.

#### **A scaffolding design framework for developing secure interoperability components in digital manufacturing platforms**

(Fraile et al., 2018b)

This paper proposes a design tool to support the development of components capable of interacting with manufacturing assets ranging from physical devices to software applications. This research has does not have a focus on DevSecOps and therefore it does not propose any security practices or activities in the specific context of DevSecOps. The proposed approach of building so-called scaffolding frameworks, also referred to as chassis, is however a concept which is also imagined to be particularly useful for DevOps in general and DevSecOps specifically as it allows a lot of the complex work to be performed upfront allowing faster development and implementation of standardised components based on the chassis.

For the above-mentioned reasons this research is excluded from our gold set, the concept of developing chassis or scaffolds is however a useful concept for our conclusions and proposals.

#### **Scaling agile software development to Large and Globally Distributed Large-scale organisations**

(Putta, 2018)

This paper has a focus on the practices and frameworks for scaling the agile methodology to an enterprise level, also referred to as enterprise agile. The paper does not propose any practices or activities relevant in the scope of DevSecOps and therefore is excluded from our gold set.

#### **A multivocal Literature Review on the use of DevOps for e-learning systems**

(Sanchez-Gordon & Colomo-Palacios, 2018)

This paper provides insight in the potential of applying DevOps methodologies in the context of e-learning solutions. The researchers performed a multi-vocal literature review which pointed out that within the academic studies little interest in the topic of DevOps for this specific context exists while the e-learning industry has an increasing but limited interest in the use of DevOps.

The paper does not propose any practices or activities relevant in the scope of DevSecOps and therefore is excluded from our gold set.

#### **Leveraging Cloud Native Design Patterns for Security-as-a-Service Applications**

(Torkura et al., 2017a)

This paper proposes a new approach for designing and deploying Security-as-a-Service applications through the application of cloud native design patterns. Some security practices and activities for DevSecOps can be gleaned from this research because these are the scenarios which SecaaS applications attempt to support through a technical implementation.

This research only has an indirect relationship to DevSecOps. Attention will be required when incorporating the results in this study with regards to identifying practices, activities or design factors for DevSecOps as a whole. The results of this study are expected to mainly provide design factors for the practice of security automation.

### A Systematic Mapping Study on Security in Agile Requirements Engineering

(Villamizar et al., 2018)

This paper provides insight in the challenges related to agile requirements engineering. The researchers point out that while traditionally security requirements engineering is mostly based on formal and extensive documentation which does not align well with the objectives of agile methods. During their research they discovered that the focus for the examined papers lies on both the conceptual specification, document and analytical layers while validation and general quality assurance topics seemed to be out of scope.

We were able to distinguish the following design factors relevant to the practice of “Security requirements analysis”:

- The introduction of new features, for example the addition of a security backlog, and new activities, such as vulnerability analysis, to the agile method.
- The introduction of new artefacts, for example the extension of a user story to also cover abuser stories.
- The introduction of guidelines to handle security issues for various stakeholders in the agile method.

### Security Requirements Engineering in the Agile Era: How Does it Work in Practice?

(Daneva & Wang, 2018)

### Effectiveness of using card games to teach threat modeling for secure web application development

(Thompson & Takabi, 2016)

## Appendix B: Thematic analysis mapping table

ID	DOI	Code	Excerpt	Notes
1	10.24425/ijet.2019.126303	Tool category	SAST (Static Application Security Testing) security scanner	
2	10.24425/ijet.2019.126303	Tool category	DAST (Dynamic Application Security Testing) security scanner	
3	10.24425/ijet.2019.126303	Tool category	A system which aim is to detect violations of IT Security principles, described in [17] can be used as Intrusion Detection System (IDS)	
4	10.24425/ijet.2019.126303	Opportunity	Discussing the details of the DevOps model is outside the scope of this paper, but in a nutshell its essence is related to treating the whole infrastructure as a code written in ansible or terraform ... This approach allows to prepare the piece of a source code that is responsible for configuring the virtual machine, configuring firewall rules, pulling the source code from the repository and then start the web application server. Taking IT security into consideration, such an approach has many advantages like recurrence of operations (a user may forget to implement one of hundreds firewall rules, however once prepared program cannot), homogeneity of the environment and the speed of action.	
5	10.24425/ijet.2019.126303	Challenge	there are also disadvantages to be acknowledged, such as - broad range of permissions given to a tool that is widely available. Critical vulnerability found in one element (CI/CD) which contains provider configuration may put the whole platform at risk as it can be compromised	
6	10.24425/ijet.2019.126303	Activity	In the whole process of software development, many other points are related with the identified threats such as identity management, verification of images used (which are installed on the servers), software testing or checking for available updates.	
7	10.24425/ijet.2019.126303	Activity	Therefore, in this case, it is crucial to perform a risk analysis to identify areas that require special attention. It should be performed to adopt appropriate security mechanisms that would minimise the probability of launching a successful attach on a system.	Risk analysis is proposed in the context of growing complexity of platforms used to run applications (e.g. OpenStack).

ID	DOI	Code	Excerpt	Notes
8	10.24425/ijet.2019.126303	Tool category	Center for Internet Security is sharing security benchmarks for OSes like: Centos, Redhat, or Ubuntu	Infrastructure related
9	10.24425/ijet.2019.126303	Tool category	Test suites are available for solutions such as Kubernetes and Docker	Infrastructure related
10	10.24425/ijet.2019.126303	Activity	It is possible to prepare a set of security tests by automated tools like Tenable Nessus or OpenSource solution like w3af	Infrastructure related
11	10.24425/ijet.2019.126303	Tool category	It is possible to prepare a set of security tests by automated tools like Tenable Nessus or OpenSource solution like w3af	Infrastructure related
12	10.24425/ijet.2019.126303	Activity	If we are considering the security of applications, there are two sets of tests to be mentioned: automatic security tests (done by a specific tools) and manual penetration testing (done by a qualified expert). The latter is always more accurate (a human can try to examine context which machines typically do not understand) but in environments like the one we are mentioning in this paper (where changes are deployed frequently) it is almost impossible to implement. The main reason for this is the time which an expert need to perform the penetration test. During the time required for performing a single test, a team of developers could prepare several changes. Thus, waiting for the test to be completed delays the release of application version which is ready to be put on production infrastructure (one of the key reasons why to switch to DevOps methodology is related to shorter releases time).	Application related
13	10.24425/ijet.2019.126303	Design factor	If we are considering the security of applications, there are two sets of tests to be mentioned: automatic security tests (done by a specific tools) and manual penetration testing (done by a qualified expert). The latter is always more accurate (a human can try to examine context which machines typically do not understand) but in environments like the one we are mentioning in this paper (where changes are deployed frequently) it is almost impossible to implement. The main reason for this is the time which an expert needs to perform the penetration test. During the time required for performing a single test, a team of developers could prepare several changes. Thus, waiting for the test to be completed delays the release of application version which is ready to be put on production infrastructure (one of the key reasons why to switch to DevOps methodology is related to shorter releases time).	Application related
14	10.24425/ijet.2019.126303	Activity	Automatic security tests can be done by several types of scanners.	Application related
15	10.24425/ijet.2019.126303	Tool category	Static application security testing (SAST - mostly source code analysis) which conducts a set of tests on the static source code.	Application related
16	10.24425/ijet.2019.126303	Design factor	Unfortunately, results of the SAST scanning often contain multiple false positives, for example, identified vulnerability could be impossible to exploit in the context of running application.	Application related

ID	DOI	Code	Excerpt	Notes
17	10.24425/ijet.2019.126303	Tool category	... Dynamic Application Security Testing (DAST) which conducts automatic penetration security testing, e.g., by crawling a website and use well-known web application vulnerabilities in order to evaluate its security are more accurate.	Application related
18	10.24425/ijet.2019.126303	Design factor	But they (DAST) also have some issues - from which the biggest one is the limited scope of testing scenarios which depends on type of tool used	Application related
19	10.24425/ijet.2019.126303	Tool category	There is also another class of automated testing tools - Interactive Application Security Testing (IAST) which combines static and dynamic testing. In this paper we do not consider IAST scanners which are available to use (both open-source and commercial ones) as they are strongly attached to the software development strategy.	Application related
20	10.24425/ijet.2019.126303	Design factor	Most of the IAST scanners are being executed during functional unit testing (where it is known which part of the source code is responsible for the certain website functionality). However, it is crucial that the unit tests of application should be created properly for the IAST to function.	Application related
21	10.24425/ijet.2019.126303	Challenge	The scope and quantity of the processed data and the pace at which new environments are being built, strengthen our belief that the current development methodologies lack a generic way to calculate system overall security level.	Full stack related
22	10.1109/cns.2017.8228701	Framework	This paper introduces the MUSA DevOps approach to holistic security assurance in multi-cloud applications and details particularly the proposed approach to dynamic assurance at operation phase	Cloud oriented
23	10.1109/cns.2017.8228701	Activity	The MUSA framework comes in form of a single solution that seamlessly integrates a number of mechanisms supporting different steps in the (multi-)cloud-based application lifecycle: application modelling, risk analysis, cloud service selection based on security controls they offer, automatic generation of composite Service Level Agreement (SLA), multi-cloud deployment, and continuous assurance (monitoring and enforcement of security behaviour) to minimise risks at runtime.	Cloud oriented
24	10.1109/cns.2017.8228701	Activity	Modelling of the application cloud and security requirements: The first step in the multi-cloud application design is the specification of the Cloud Provider Independent Model (CPIM) of the application, a task supported by the MUSA Modeler. The CPIM, captured in a MUSA extended CAMEL language, is the specification of the multi-cloud application in a level of abstraction independent from specific Cloud services and providers the application will use.	Cloud oriented

ID	DOI	Code	Excerpt	Notes
25	10.1109/ cns.2017.8228701	Tool category	Modelling of the application cloud and security requirements: The first step in the multi-cloud application design is the specification of the Cloud Provider Independent Model (CPIM) of the application, a task supported by the MUSA Modeler. The CPIM, captured in a MUSA extended CAMEL language, is the specification of the multi-cloud application in a level of abstraction independent from specific Cloud services and providers the application will use.	Cloud oriented
26	10.1109/ cns.2017.8228701	Design factor	Continuous Risk Assessment that helps in the selection of the security controls and metrics that will be granted in the Security SLA and controlled at runtime. The activity follows a methodology similar to the one described in [25]. It allows for selecting the relevant threats according to the component nature, evaluating the technical and business impact of the threat exploitation, as well as, in order to minimise such impact, defining the desired countermeasures or controls required over the cloud services the components will use or be deployed in. The risk assessment is continuously updated with the feedback from the continuous monitoring of the controls behaviour at runtime.	Cloud oriented
27	10.1109/ cns.2017.8228701	Tool category	The Cloud services selection relies on the use of the MUSA Decision Support Tool (DST). In order to take most out of cloud services combination in terms of security, the DevOps Team is supported in the selection of cloud services that best match the security requirements of the multi-cloud application components. The best match is calculated by comparing the security controls offered by the cloud services under study (those previously categorised in the MUSA CSP Data Repository) with the security requirements of the individual components.	Cloud oriented
28	10.1109/ cns.2017.8228701	Activity	Multi-cloud application components security SLA templates generation. Once the most appropriate cloud service is selected for each of the components, the DevOps Team will use the MUSA SLA Generator to automatically create the Security SLA templates of the components. The Security SLA templates define the required security Service Level Objectives (SLOs) of the components in the basis of the SLOs required over the cloud services that they will use. For this, the MUSA framework supports the verification of the feasibility of the components Security SLA templates by checking whether the cloud service offerings selected in the previous step do offer such security requirements (in form of security controls). In case they do not, the MUSA security enforcement agents may be adopted to offer them.	Cloud oriented

ID	DOI	Code	Excerpt	Notes
29	10.1109/ cns.2017.8228701	Tool category	Multi-cloud application components security SLA templates generation. Once the most appropriate cloud service is selected for each of the components, the DevOps Team will use the MUSA SLA Generator to automatically create the Security SLA templates of the components. The Security SLA templates define the required security Service Level Objectives (SLOs) of the components in the basis of the SLOs required over the cloud services that they will use. For this, the MUSA framework supports the verification of the feasibility of the components Security SLA templates by checking whether the cloud service offerings selected in the previous step do offer such security requirements (in form of security controls). In case they do not, the MUSA security enforcement agents may be adopted to offer them.	Cloud oriented
30	10.1109/ cns.2017.8228701	Activity	Multi-cloud application Composite Security SLA generation. In this step the DevOps Team is supported in the automatic generation of the final offered Security SLA of the multi-cloud application. The Security SLA of the overall application is the result of the composition of the individual components Security SLAs, i.e. it considers the Security SLAs of individual components as well as the component nature (e.g. web server, database, etc.) and the relationships between the components (e.g. uses, is deployed in, protects, etc.). The last step in design process will therefore be the Security SLA composition activity.	Cloud oriented
31	10.1109/ cns.2017.8228701	Activity	Continuous Monitoring of the application Security SLA once the components are deployed and running, and early feedback to development. Finally, at runtime or operation phase, the MUSA Security Assurance platform starts monitoring the multi-cloud application based on the final SLAs and the Implementation plan. In case potential or actual violations of the SLA are found reaction measures such as remodeling the application or re-evaluating risks again are recommended.	Cloud oriented
32	10.1109/ cns.2017.8228701	Tool category	Continuous Monitoring of the application Security SLA once the components are deployed and running, and early feedback to development. Finally, at runtime or operation phase, the MUSA Security Assurance platform starts monitoring the multi-cloud application based on the final SLAs and the Implementation plan. In case potential or actual violations of the SLA are found reaction measures such as remodeling the application or re-evaluating risks again are recommended.	Cloud oriented
33	10.1109/ cns.2017.8228701	Activity	Dynamic adaptation of the multi-cloud application to meet the security status guaranteed in the Security SLA. The MUSA Security Assurance platform also supports the dynamic enforcement of secure behavior of the application by means of activation of MUSA security enforcement agents. The agents are activated as a reaction mechanism to a security problem detected in previous step. The monitoring step is informed on the status of the activation of the enforcement agents as well as on the required enforcement events.	Cloud oriented

ID	DOI	Code	Excerpt	Notes
34	10.1109/ cns.2017.8228701	Activity	Continuous Monitoring of Security SLAs fulfilment: The objective of this activity is to monitor the runtime security behavior of the selected multi-cloud application components in order to early react to possible security incidents.	Cloud oriented
35	10.1109/ cns.2017.8228701	Activity	Dynamic adaptation and reaction to security incidents: The goal of this activity is to decide and execute the needed reaction measures in case security incidents or Security SLA violations occur. The reaction to security incidents in MUSA relies on different mechanisms depending on the cause of the incident and whether it is an alert or a violation. In general, is up to the DevOps Team the decision of whether to react at the level of alert before any violation takes place.	Cloud specific
36	10.1145/2896941.28 96946	Practice	Automation of all activities related to software development is one of the common software practices used in DevOps culture. In this paper we refer to this security practice as 'use of automation activities'.	DevOps generic
37	10.1145/2896941.28 96946	Activity	Automation of code review: Code review is the activity of presenting source code changes for comment, approval, and improvisation. Automation of code review is the activity of performing code review and giving appropriate feedback to the software developers of interest, using open source and commercial static analysis tools.	DevOps generic
38	10.1145/2896941.28 96946	Activity	Automation of monitoring: Automation of monitoring refers to the activity of gathering, reporting, and storing system-related information such as CPU usage and memory usage for further analysis using automated tools.	DevOps generic
39	10.1145/2896941.28 96946	Activity	Automation of software defined firewall: Automation of software defined firewall is the activity to maintain consistent policies to manage the settings of the organization's firewall using automated tools.	DevOps generic
40	10.1145/2896941.28 96946	Activity	Automation of software licensing: Software licensing is the activity of enabling users to purchase, install, and use software in accordance with a set of conditions set by the software vendor. We define automation of software licensing as the activity that ensures users are purchasing, installing, and using the software as per the conditions set by the software vendor of interest, using automated tools.	DevOps generic
41	10.1145/2896941.28 96946	Activity	Automation of testing: Automation of testing refers to the activity of automatically performing testing tasks, such as test case management, test monitoring and control, and test data generation, for different types of tests namely, functional testing, integration testing, and unit testing.	DevOps generic

ID	DOI	Code	Excerpt	Notes
42	10.1145/2896941.28 96946	Practice	The practice of actively collaborating with other teams is another practice that software practitioners have stated in the Internet artifacts. Increased collaboration between development teams, security teams, and operation teams have been mentioned in 16 Internet artifacts. In 13 Internet artifacts, authors reported that instead of operating in silos, the security team could adopt existing DevOps automation activities inside the organization and customize existing security tools in a way that ensures the feedback cycle between the security team, with the other teams is short. In another Internet artifact, an author stated developers could learn from the security team and build or customize the necessary security tools by themselves.	DevOps generic
43	10.1145/2896941.28 96946	Practice	Software practitioners referred security training for development team members to integrate security in DevOps organizations. This practice was mentioned in three Internet artifacts. Completing relevant online coursework, attending developer boot camps, and in-house security awareness meetings are the three activities that the authors mentioned on implementing this practice.	DevOps generic
44	10.1145/2896941.28 96946	Practice	Use of Non-Automated Security Activities We identified 10 security activities from the Internet artifacts of interest.	DevOps generic
45	10.1145/2896941.28 96946	Activity	Design Review: Design review is the activity of reviewing the design of the entire software as well as different modules of the software to identify potential security flaws that might be exposed at latter stages of software development.	DevOps generic
46	10.1145/2896941.28 96946	Activity	Input Validation: Input validation is the activity of performing data validation and rejecting non-conformant data that are both entering and exiting the software of interest.	DevOps generic
47	10.1145/2896941.28 96946	Activity	Isolation of Untrusted Inputs: Isolation of untrusted inputs is the activity of identifying and performing security measures on resources that are not verified as secure by the system vendor for example, third party library used to develop the software.	DevOps generic
48	10.1145/2896941.28 96946	Activity	Performing Compliance Requirements: Performing compliance requirements is the activity that continuously checks if the software of interest satisfies the federal regulations set by the government as per the domain of interest such as healthcare, and trade organizations.	DevOps generic
49	10.1145/2896941.28 96946	Activity	Performing Security Configurations: Performing security configurations is the activity of identifying potential resources that contain configuration information related to the software and securing them using security tests.	DevOps generic
50	10.1145/2896941.28 96946	Design factor	Performing Security Policies: Performing security policies is the activity of ensuring all software related information is only accessible to entities with appropriate level of authorization.	DevOps generic

ID	DOI	Code	Excerpt	Notes
51	10.1145/2896941.2896946	Activity	Security Requirements Analysis: Security requirements analysis is the activity of identifying a set of capabilities that must be possessed by the software to satisfy a set of specifications that ensures prevention of intentional or unintentional unauthorized access to the software.	DevOps generic
52	10.1145/2896941.2896946	Activity	Performing Manual Security Tests: Performing manual security tests is the activity that aims to reduce software risk by applying two tasks: ensuring that the software's functionality is properly implemented and executing risk-based security testing via simulating an attacker. Security tests such as penetration testing can be performed in an automated or non-automated fashion to systematically compromise different parts of the software. We do not include performing manual security tests as part of the 'use of automation activities' practice as this activity is non-automated.	DevOps generic
53	10.1145/2896941.2896946	Activity	Risk Analysis: Risk analysis is the activity of creating design specifications relevant to security and later on testing those design specifications.	DevOps generic
54	10.1145/2896941.2896946	Activity	Threat Modeling: Threat modeling is the activity of identifying, describing, and categorizing threats along with the actors or agents who are associated with those threats.	DevOps generic
55	10.1007/978-3-319-67383-7_2	Practice	It promotes an extension to DevOps' goal of promoting collaboration between developers and operators by involving security experts from the start as well.	DevOps generic
56	10.1007/978-3-319-67383-7_2	Principle	The principles that characterize DevSecOps are based on DevOps and the CAMS principles, culture, automation, measurement, and sharing, but with the addition of adding security from the start.	DevOps generic
57	10.1007/978-3-319-67383-7_2	Practice	DevSecOps means to include collaboration with the security team as well as promote a culture where operations and development also work on integrating security in their work. That means involving the security team from the planning stages, and making sure everyone agrees that security is everyone's responsibilities	DevOps generic
58	10.1007/978-3-319-67383-7_2	Practice	DevSecOps promotes a focus on automating security as well, to be able to keep up with the speed and scale achieved by DevOps	DevOps generic
59	10.1007/978-3-319-67383-7_2	Design factor	The aim should be 100% automation of security controls, where the controls can be deployed and managed without manual interference	Design factor for automation of security controls

ID	DOI	Code	Excerpt	Notes
60	10.1007/978-3-319-67383-7_2	Design factor	Any security functionality not automated in the available tools will create friction in the DevOps cycle.	Design factor for automation of security controls
61	10.1007/978-3-319-67383-7_2	Design factor	It is important to implement automatic security in a way that does not hinder DevOps' agility in any way, which can cause friction	Design factor for automation of security controls
62	10.1007/978-3-319-67383-7_2	Activity	DevSecOps promotes the use and development of metrics that track threats and vulnerabilities throughout the software development process	DevOps generic
63	10.1007/978-3-319-67383-7_2	Design factor	Automatic security controls throughout the software development process means metrics are available to track threats and vulnerabilities in real-time and that allows the organization to verify how good an application is on demand	Design factor for security monitoring
64	10.1007/978-3-319-67383-7_2	Opportunity	This allows security controls to be fast, scalable and effective thus making it possible to keep a high pace for detecting errors, alerting about the errors, fixing the errors, finding countermeasures for future errors and forensics to identify why an error occurred. This not only helps to lower risk and time spent on errors, but also makes it easier to understand risk and create policies and procedures. The automation allows processes to be consistent and repeatable, with predictable outcomes for similar tests, it allows logging and documentation to be automatic and letting security tests be run at the push of a button frees up developers' time to write code instead of running tests. This also reduces the risk for human error. The ability to store security policy templates that is created during a development process in a central repository means that security teams don't need to manually configure every new environment when starting a new project which frees security experts from manual, repetitive and unproductive work.	Benefit for automation of security controls
65	10.1007/978-3-319-67383-7_2	Principle	DevSecOps promotes the inclusion of the security team in the sharing promoted in a DevOps environment	DevOps generic
66	10.1007/978-3-319-67383-7_2	Principle	DevSecOps promotes a shift to the left for security, where it is to be included in every part of the software development process. This means that security teams are involved from the very first planning step and is part of planning every iteration of the development cycle. It also means security is there to help developers and operators on security considerations.	DevOps generic
67	10.1007/978-3-319-67383-7_2	Opportunity	By involving security experts from the start of the development process it is easier to plan and execute integration of security controls throughout the development process without causing delays or creating issues by implementing security controls after systems are running.	Benefit for shift left on security

ID	DOI	Code	Excerpt	Notes
68	10.1007/978-3-319-67383-7_2	Activity	Practicing secure DevOps means that organizations have to develop expertise and processes to best discover, protect against, and find solutions to threats and risks, preferably ahead of time. Performing risk assessments from the first planning stage and continuously before every iteration is important as a way to prioritize risks, examine controls already in place and decide which are needed going forward.	DevOps generic
69	10.1007/978-3-319-67383-7_2	Activity	Threat modeling is another method where you attack your system on paper early in the development cycle to identify how an attack can occur and where it is most likely to happen.	DevOps generic
70	10.1007/978-3-319-67383-7_2	Activity	Continuous testing: Automatic security controls at every part of the software development process is important for security assurance and allows tests to continuously scan code for changes, continuously detect anomalies, and automatic rollback of code when needed.	DevOps generic
71	10.1007/978-3-319-67383-7_2	Activity	Monitoring and logging: When automating security controls throughout the software developing process it is important for those involved to be able to generate evidence on demand that controls are working and that they are effective.	DevOps generic
72	10.1007/978-3-319-67383-7_2	Design factor	To that end, it is important to monitor every part of the inventory and to log every resource.	DevOps generic
73	10.1007/978-3-319-67383-7_2	Activity	Security as code: This means to define security policies, for example integration testing, and network configuration and access, and write scripted templates or configuration files that can be implemented into the development process from the start of the project. These codified security policies can then be activated automatically according to schedules or be activated by user (simple push of a button) and be stored in a central repository for reuse on new projects.	DevOps generic
74	10.1007/978-3-319-67383-7_2	Activity	Red-Team and security drills: To stay ahead of possible attackers, practitioners of DevSecOps create a Red-Team that runs security drill on the deployed soft-ware. They have the task of finding and exploiting vulnerabilities in the system. This not only helps to find security flaws, but improves measurements, and helps the organization find solutions. The point of the Red-Team is to have people that never claim something can't possibly happen.	DevOps generic
75	10.1109/ares.2016.92	Activity	automating tests to detect non-compliance, tracking compliance breaches through automated reporting of violations, continuous monitoring and maintenance of a service catalog with tested and certified services	DevOps generic

ID	DOI	Code	Excerpt	Notes
76	10.1109/ares.2016.92	Activity	Cash et al. call for integrating into SecDevOps security scanning and configuration automation.	DevOps generic
77	10.1109/ares.2016.92	Activity	automated monitoring, automated deployment pipeline, and automated testing contribute positively to the security of the software	DevOps generic
78	10.1109/ares.2016.92	Activity	Schneider [6] describes the different stages of dynamic security scanning that can be used by organizations to integrate security into DevOps.	Application
79	10.1109/ares.2016.92	Design factor	The four levels of scanning are: (1) pre-authentication scanning, that involves scanning the public attack surface; (2) post-authentication scanning, that involves session maintenance, user role management, and logout and auto-relogin detection; (3) backend scanning of the various application layers independently; and (4) scanning workflows specific to the targeted application.	Design factor for dynamic security scanning
80	10.1109/ares.2016.92	Practice	Vries believes that security activities need to adopt concepts used in DevOps. The talk advocates the collaboration between the development team and the business owner of the software to set the security goals.	
81	10.1109/ares.2016.92	Design factor	Vries believes that security activities need to adopt concepts used in DevOps. The talk advocates the collaboration between the development team and the business owner of the software to set the security goals.	Design factor for collaboration
82	10.1109/ares.2016.92	Activity	The talk also calls for automating security tests and security scans in SecDevOps processes.	
83	10.1109/ares.2016.92	Framework	Schneider introduces the SecDevOps Maturity Model (SDOMM). The model is a manual to help projects achieve certain security aspects through automation in a continuous integration (CI) build chain. The model is useful for organizations willing to make the change to SecDevOps.	
84	10.1109/ares.2016.92	Design factor	Farroha et al. suggest a set of requirements for compliance policies which include prohibiting unauthorized access, maintaining a log for accesses to sensitive data, and monitoring data operations.	Design factor for policies
85	10.1109/ares.2016.92	Activity	Use of automation activities like automated code review, automated monitoring, automated testing are popular automation activities for security integration in DevOps environments.	
86	10.1109/ares.2016.92	Activity	Matteti et al. describe the need to secure Linux containers, which are considered a break-through for DevOps because of their contribution to simplifying automated deployments. Linux containers expose file systems, networks and kernels to attacks.	Container technology



ID	DOI	Code	Excerpt	Notes
87	10.1109/ares.2016.92	Framework	The authors propose the LiCSHield Framework that provides protection to hosts, by confining accesses of containers and container management daemons to perform only the operations observed in testing environments and restricting container operations, by tightening the internal noisy environment.	Container technology
88	10.1109/ares.2016.92	Tool category	Security tools; monitoring & alerting tools; logging tools	
89	10.1109/ares.2016.92	Activity	Security testing, static analysis and formal verification are some techniques that may be used to secure a pipeline.	CI/CD technology oriented
90	10.1109/ares.2016.92	Tool category	Scanning tools, Security frameworks, Results consolidation tools; Monitoring tools; Logging tools	
91	10.1109/ares.2016.92	Activity	Bass et al. propose studying the fetching of code from third party libraries for their vulnerabilities and the security of the cloud where the image is deployed.	
92	10.1109/ares.2016.92	Design factor	Farroha et al. advocate the importance of involving software stakeholders to build a secure system. The rights to protect sensitive data and ensure compliance need to be granted to stakeholders to enable security.	Design factor for collaboration
93	10.1109/ares.2016.92	Practice	Rahman et al. reviewed the literature about SecDevOps and found that most artifacts propose enforcing the collaboration among the security team, the development team, and the operations team for a better integration of security principles into DevOps.	
94	10.1109/ares.2016.92	Activity	In addition, they found that the literature suggests training the developers to build security tools, which could then be integrated into SecDevOps processes	
95	10.1109/ares.2016.92	Design factor	Well-defined policies regarding information exchange across teams should be in place to prevent security threats due to collaboration.	Design factor for policies
96	10.1145/3098954.3103172	Opportunity	For security, this implies that information on detected attacks can be fed back to the development, enabling faster eradication of vulnerabilities in software	
97	10.1145/3098954.3103172	Design factor	Suitable metrics, for example, would let designers evaluate alternative designs and determine which is more secure for a given deployment (cf., Section 4). Designers would also be able to reason about the minimum capabilities and effort an attacker needs to violate the security properties.	Design factor for security design

ID	DOI	Code	Excerpt	Notes
98	10.1145/3098954.3103172	Principle	Despite the conspicuous advantages of this approach, measuring the level of security in a given piece of software is notoriously difficult, and instead it has been argued that the next-best thing is to measure second-order effects, i.e., measure the software security activities that are performed by the developers as part of the development process.	
99	10.1145/3098954.3103172	Activity	The ability to make quick changes in the code if a vulnerability is discovered in operations	
100	10.1145/3098954.3103172	Activity	A closely related activity is ensuring that flaws or bugs that are discovered in operations are fed back to development.	
101	10.1145/3098954.3103172	Activity	In the case of DevOps, a promising approach relies on the elicitation of security metrics during the Software Development Life Cycle (SDLC) in order to allow the continuous/agile evaluation of required vs. achieved security levels.	
102	10.1145/3098954.3103172	Activity	The risk management process ensures that issues are identified and mitigated early in the development process and followed by periodic reviews aligned to the agile DevOps vision.	
103	10.1145/3098954.3103172	Activity	A risk-based approach to DevOps for cloud services is a holistic activity that should be integrated into every aspect of the developer organization, from planning and system development life cycle processes to security controls/metrics allocation.	Cloud oriented
104	10.1145/3098954.3103172	Activity	Prioritization of defects is challenging.	
105	10.1145/3098954.3103172	Design factor	Highly critical security defects can be defined as blocking defects, and as such are easier to deal with - they have to be fixed as quickly as possible. Level 2 or lower security defects are more tricky; they have to compete with all other feature requests and defects, and risk getting pushed back at every junction. One approach to deal with this might be to set a time limit for all lower-priority security defects that are discovered. This will acknowledge that there may be more important issues for the developers to fix right at the moment but will take into account that the likelihood of a security defect being exploited will increase as time goes by, and eventually it will become a number 1 priority.	Design factor for prioritization of defects
106	10.1145/3098954.3103172	Activity	It could be argued that software security education of developers is more important in agile development than in traditional waterfall.	

ID	DOI	Code	Excerpt	Notes
107	10.1145/3098954.3103172	Design factor	However, exactly how much is needed is a matter for debate. We certainly don't think that it will be possible to teach every developer to be a software security expert, but we could aspire to teach every developer enough to enable them to identify areas where they would benefit from the advice of an expert. It has been claimed that if we can only make developers think about security, the number of security defects are reduced by 50%.	Design factor for security training
108	10.1145/3098954.3103172	Practice	Tools are important, and anything that can be automated should be automated. This is particularly true in DevOps, where rapid deployments are only possible due to tightly configured deployment scripts.	
109	10.1145/3098954.3103172	Practice	Communications between Dev and Ops is also vital in any situation where they are not actually the same persons. It therefore becomes important to establish who should know what.	
110	10.1145/3098954.3103172	Framework	The Building Security in Maturity Model (BSIMM) states ...	
111	10.1145/3098954.3103172	Activity	... the necessity of having a Software Security Group in any software development organization but based on industry reports it seems even more important to have security champions who are part of the development teams. An external SSG who performs short-term helicopter-style incursions will inevitably be perceived as an outside agent hindering progress. Another issue is the size of the development organization; many of the BSIMM organizations have hundreds and thousands of developers, and it is not immediately clear if their experiences are applicable to small European firms with tens of developers.	
112	10.1145/3230833.3233275	Framework	The Building Security In Maturity Model (BSIMM) is a software security framework of four domains each broken down into three practices, where each practice consists of a collection of concrete and measurable software security activities.	
113	10.1145/3230833.3233275	Activity	It is effectively impossible to say which is "more secure". Instead, McGraw and colleagues decided to measure second-order effects, i.e., identify which software development activities contribute positively to software security, and measure to what extent these are performed in a given organization	
114	10.1145/3230833.3233275	Activity	Yasar and Kontostathis propose focusing on security requirements, threat modeling, environment configuration, static analysis, code review, penetration testing, environment testing, and finally a manual security review.	
115	10.1145/3230833.3233275	Activity	SM2.3 Create or grow a satellite. The presence of security-minded developers across the organization will aid in resolving quick fixes in case of incidents. The term "satellite" is the one used in the BSIMM; some organizations have formalized this as a security champion for some or all development teams.	DevOps Incident response oriented

ID	DOI	Code	Excerpt	Notes
116	10.1145/3230833.3233275	Activity	CP2.1 Identify PII data inventory. Due to GDPR, it will be paramount to know what kind of Personal Identifiable Information (PII) is handled by the system under attack, and where in the system it is stored or handled.	DevOps Incident response oriented
117	10.1145/3230833.3233275	Activity	T3.5 Establish SSG office hours. In case of an incident, the handlers will benefit from having an available point of contact for software security issues.	DevOps Incident response oriented
118	10.1145/3230833.3233275	Activity	AM2.7 Build an internal forum to discuss attacks. In order to learn from attacks, they need to be discussed -both to improve handling and to update software so that the same attack cannot succeed twice.	DevOps Incident response oriented
119	10.1145/3230833.3233275	Activity	SR3.1 Control open source risk. If an attack is due to an open source vulnerability, you need to know which components use the library in question.	DevOps Incident response oriented
120	10.1145/3230833.3233275	Activity	SE1.1 Use application input monitoring. Monitoring the input to your application can help detecting an attack as it happens.	DevOps Incident response oriented
121	10.1145/3230833.3233275	Activity	SE3.3 Use application behavior monitoring and diagnostics. Monitoring the behavior of your application can help detecting an attack as it happens.	DevOps Incident response oriented
122	10.1145/3230833.3233275	Activity	CMVM1.1 Create or interface with incident response. To have any hope of being able to make software changes quickly enough when an attack is manifest, there must be an interface between developers and incident response.	DevOps Incident response oriented
123	10.1145/3230833.3233275	Activity	CMVM1.2 Identify software defects found in operations and feed them back to development. This has the dual effect of learning from incidents and improving the development lifecycle.	DevOps Incident response oriented
124	10.1145/3230833.3233275	Activity	CMVM2.1 Have emergency codebase response. When a software related attack occurs, it is important to be able to make quick changes in order to stop the attack and prevent the same type of attack from occurring again.	DevOps Incident response oriented
125	10.1145/3230833.3233275	Activity	CMVM2.3 Develop an operations inventory of applications. This extends SR3.1 by creating a complete overview of which libraries and/or components are used in which applications.	DevOps Incident response oriented

ID	DOI	Code	Excerpt	Notes
126	10.1145/3230833.3233275	Activity	CMVM3.3 Simulate software crises. This implies running preparedness exercises involving both incident responders and developers.	DevOps Incident response oriented
127	10.1109/ access.2019.2930000	Activity	To that end, we have defined and formalized an activity that supports 'Fast and Continuous Feedback from Ops to Dev' by providing a flexible monitoring infrastructure so that teams can configure their monitoring and alerting services according to their criteria (you build, you run, and now you monitor) to obtain fast and continuous feedback from the operation and thus, better anticipate problems when a production deployment is performed. This activity has been formalized using the Software & Systems Process Engineering Metamodel by OMG and its instantiation is described through a case study that shows the versioned and repeatable configuration of a cybersecurity monitoring infrastructure (Monitoring as Code) through virtualization and containerization technology. This self-service monitoring/alerting allows breaking silos between dev, ops, and sec teams by opening access to key security metrics, which enables a sharing culture and continuous improvement.	
128	10.1109/ access.2019.2930000	Design factor	To that end, we have defined and formalized an activity that supports 'Fast and Continuous Feedback from Ops to Dev' by providing a flexible monitoring infrastructure so that teams can configure their monitoring and alerting services according to their criteria (you build, you run, and now you monitor) to obtain fast and continuous feedback from the operation and thus, better anticipate problems when a production deployment is performed. This activity has been formalized using the Software & Systems Process Engineering Metamodel by OMG and its instantiation is described through a case study that shows the versioned and repeatable configuration of a cybersecurity monitoring infrastructure (Monitoring as Code) through virtualization and containerization technology. This self-service monitoring/alerting allows breaking silos between dev, ops, and sec teams by opening access to key security metrics, which enables a sharing culture and continuous improvement.	Design factor for self-service monitoring and alerting
129	10.1109/ access.2019.2930000	Activity	Many of these principles and practices can be applied to security by building security in from the start (shifting security left), automating security test and monitoring, and making collaboration between developers and security engineers effective, aka. DevSecOps.	
130	10.1109/ access.2019.2930000	Principle	Many of these principles and practices can be applied to security by building security in from the start (shifting security left), automating security test and monitoring, and making collaboration between developers and security engineers effective, aka. DevSecOps.	

ID	DOI	Code	Excerpt	Notes
131	10.1109/ access.2019.2930000	Practice	Many of these principles and practices can be applied to security by building security in from the start (shifting security left), automating security test and monitoring, and making collaboration between developers and security engineers effective, aka. DevSecOps.	
132	10.1109/ access.2019.2930000	Best practices	In fact, the edition of a new standard on DevOps is expected for December 2020 (IEEE P2675 DevOps Standard for Building Reliable and Secure Systems Including Application Build, Package and Deployment)	
133	10.1109/ access.2019.2930000	Design factor	Security metrics related to the logical security of the information systems, i.e. cybersecurity, and specifically IoT-based systems. Example are: Declared Vulnerabilities or CVEs (Common Vulnerabilities and Exposures) that are the measures taken from the so-CVSSs (Common Vulnerabilities Scoring Systems) that indicate levels of vulnerability of the elements that make up a system (HW and SW). System Availability measurements that indicate whether the system, from the point of view of users and external applications, is responding to their requests—they can provide information for the detection of denial of service attacks. Finally, intrusion, such as abnormal behavior of devices, detection of signatures, DNS-based intrusion detection.	Design factor for continuous security monitoring
134	10.1109/ access.2019.2930000	Tool category	Commonly these measures are taken from SIEMs (Security Information and Event Management) that collect data from system logs (e.g., antivirus, servers, network appliances, software components, etc.), analyze and correlate data from these logs, and detect and report security events.	Tool category for security monitoring
135	10.1109/ ms.2017.3571578	Opportunity	Raynaud emphasizes the importance of building security in from the start, because treating security as a "bolt-on" to the end of the process is far costlier and can damage the relationship between security and development teams.	Incident response oriented
136	10.1109/ ms.2017.3571578	Activity	Many DevOps principles—such as test automation—can easily be applied to security, and the adoption of these principles can help products and businesses succeed securely.	Incident response oriented
137	10.1109/ ms.2017.3571578	Activity	DevSecOps puts security at the forefront of requirements to avoid the costly mistakes that come from treating security as an after-thought.	Incident response oriented
138	10.1109/ ms.2017.3571578	Design factor	We want security to be included in the nonfunctional requirements	Incident response oriented

ID	DOI	Code	Excerpt	Notes
139	10.1109/ms.2017.3571578	Design factor	Successful implementation [of DevSecOps] happens when the security team provides knowledge and tools and the DevOps team runs them. There's no reason for a security team to run the tooling as a completely out-of-band management process. Use the tools you have at your disposal already.	Incident response oriented
140	10.1109/ms.2017.3571578	Design factor	The CI/CD (continuous integration / continuous delivery) process, for example, is fantastic from a security point of view.	Incident response oriented
141	10.1109/ms.2017.3571578	Activity	The [improved] results you get from penetration testing ... after security training for developers is really impressive.	Incident response oriented
142	10.1109/ms.2017.3571578	Activity	Start by sitting with each other. I've done lots of incident response and forensics sitting in a glass box, where nobody can actually see what you're doing. Why should you hide everything? Working in silos never works.	Incident response oriented
143	10.1109/ms.2017.3571578	Practice	Use the methodology of automation for the benefit of security. When [incident response teams] realize attacks are coming against a particular aspect of your website or application, include that as part of the QA process.	Incident response oriented
144	10.1109/ms.2017.3571578	Activity	Use the methodology of automation for the benefit of security. When [incident response teams] realize attacks are coming against a particular aspect of your website or application, include that as part of the QA process.	Incident response oriented
145	10.1109/ms.2017.3571578	Activity	Give the attack pattern to your developers, so that they can actually change the application accordingly.	Incident response oriented
146	10.1109/ms.2017.3571578	Activity	Having a "security champion" is one way to do it. This is where the security team teaches one of the developers about security, and then [that person] disseminates the information to the rest of the team. It's really about knowledge sharing.	Incident response oriented
147	10.1109/ms.2017.3571578	Design factor	I used to work in a company that was doing high-frequency trading. They had gamified finding bugs. Two years in, we had five known issues that we wanted the developers to discover. One of the guys came back with 10 of them. At this point we said, "Wow, they got it."	Incident response oriented
148	10.1109/ms.2017.3571578	Principle	[Shifting security left] is when you start from the nonfunctional requirements. For example, in a financial company you explain [at the start], "We have to think about PCI requirements." That's the essence of it: start from the beginning with all the different teams.	Incident response oriented

ID	DOI	Code	Excerpt	Notes
149	10.1109/ms.2017.3571578	Challenge	Initially there will be a learning curve, where suddenly this security person is asking lots of questions.	Incident response oriented
150	10.1109/ms.2017.3571578	Opportunity	But if you think about the costs of implementing security later on, that's completely different. [Think about] fixing a bug in production. That'll cost you a fortune. You'll have to stop production, redo QA, [rebuild] all your artifacts, do all the version control again, and [update] all your documentation. If you do it at the beginning, once everything is being built, you can reduce these costs. By shifting security left, by discovering issues and bugs at an earlier stage, you can easily incorporate it as part of your QA process. The lag you'll experience will go down, and the cost of fixing security will be much lower.	Incident response oriented
151	10.1109/ms.2017.3571578	Activity	DevSecOps emphasizes threat modeling, which is quite fun.	Incident response oriented
152	10.1109/ms.2017.3571578	Activity	incorporate those (security) tests as part of QA. It's easy. Your OWASP Top 10: incorporate those as part of your QA process. Give developers the ability to test for [those issues] themselves.	Incident response oriented
153	10.1016/j.jss.2018.06.073	Design factor	Studies have shown (e.g. Yuan et al., 2015; Wang et al., 2017; Williams, 2015) that including knowledge base (e.g. taxonomies, catalogs of misuse and abuse cases, attack scenarios and trees, etc.) helps the analyst to identify and analyze threats. Therefore, we were interested to record which existing techniques provide a knowledge base.	Design factor for threat modeling
154	10.1016/j.jss.2018.06.073	Tool category	We have assessed the techniques as knowledge based if they are supported by some external source of information which helps raise the quality of outcomes. For instance, some techniques provide a catalog of example threats (e.g. STRIDE Shostack, 2014; Torr, 2005), templates (e.g. misuse cases) or even use one of the existing databases (such as CAPEC4, CWE5, CVE6) to compute threat suggestions.	Tool for threat modeling
155	10.1016/j.jss.2018.06.073	Design factor	Misuse cases (MUC) are derived from use cases in requirements engineering. In the form of templates, they are used to capture textual descriptions of threat paths, alternative paths, mitigations, triggers, preconditions, assumptions, attacker profiles, etc.	Design factor for threat modeling
156	10.1016/j.jss.2018.06.073	Design factor	The literature also mentions abuse cases, MUC maps and MUC scenarios. The difference between misuse and abuse cases is subtle and the two terms are sometimes used interchangeably. Strictly speaking, abuse is misuse with malicious intent. MUC maps and scenarios both focus on representing chained attacks, from start to the end of vulnerability exploitation.	Design factor for threat modeling

ID	DOI	Code	Excerpt	Notes
157	10.1016/j.jss.2018.06.073	Design factor	Another way of identifying alternative paths of attack is by using attack (or threat) trees, where the root node is refined into leaves re-presenting all possible attacker actions. Therefore an attack path is a single path starting at leaf node leading to the root node. Attack trees are commonly adopted in a combination with other techniques.	Design factor for threat modeling
158	10.1016/j.jss.2018.06.073	Tool category	For instance, LINDDUN Deng et al. (2011) proposes a combined analysis by first mapping the threats to (DFD) elements, using threat tree patterns and usage scenarios in order to identify MUC scenarios.	Tool for threat analysis (types of input)
159	10.1016/j.jss.2018.06.073	Design factor	Much like threat patterns, problem frames are used to describe problems in software engineering. They define an intuitively identifiable problem class in terms of its context and the characteristics of its domains, interfaces and requirements (Jackson, 2001). As such problem frames are rather general in scope, therefore conceptualized security problem frames were soon introduced (Hatebur and Heisel, 2005; Beckers et al., 2013b; Lin et al., 2004).	Design factor for threat modeling
160	10.1016/j.jss.2018.06.073	Design factor	Goal-oriented requirements engineering (GORE) perceives systems as a set of agents communicating in order to achieve goals. In GORE goals (or anti-goals) are refined until finally requirements (or anti requirements) are achieved.	Design factor for threat modeling (types of input)
161	10.1016/j.jss.2018.06.073	Design factor	Finally, several software-centric techniques are well recognized in the software engineering community, particularly in the industrial space, such as STRIDE (Shostack, 2014; Torr, 2005), CORAS (Lund et al., 2011), P.A.S.T.A (UcedaVelez and Morana, 2015), DREAD (Owasp, 2017), Trike (Saitta et al., 2005), to name a few.	Design factor for threat modeling (types of input)
162	10.1016/j.jss.2018.06.073	Tool category	Finally, several software-centric techniques are well recognized in the software engineering community, particularly in the industrial space, such as STRIDE (Shostack, 2014; Torr, 2005), CORAS (Lund et al., 2011), P.A.S.T.A (UcedaVelez and Morana, 2015), DREAD (Owasp, 2017), Trike (Saitta et al., 2005), to name a few.	Design factor for threat modeling (types of input)
163	10.1016/j.jss.2018.06.073	Design factor	Risk-centric threat analysis techniques focus on assets and their value to the organization. They aim at assessing the risk and finding the appropriate mitigations in order to minimize the residual risk. Their main objective is to estimate the financial loss for the organization in case of threat occurrence (e.g. CORAS Lund et al., 2011). Therefore, when risk-centric techniques are used assets dictate the priority of elicited security requirements.	Design factor for threat modeling

ID	DOI	Code	Excerpt	Notes
164	10.1016/j.jss.2018.06.073	Design factor	On the other hand, attack-centric threat analysis techniques focus the analysis around the hostility of the environment. They put emphasis on identifying attacker profiles and attack complexity for exploiting any system vulnerability (e.g. Attack trees Mauw and Oostdijk, 2005). Their main objective is to achieve high threat coverage and identify appropriate threat mitigations.	Design factor for threat modeling
165	10.1016/j.jss.2018.06.073	Design factor	Finally, the literature also mentions so-called software-centric threat analysis techniques. This group includes techniques that focus the analysis around the software under analysis. For example, in STRIDE Shostack (2014); Torr (2005) the analysis is performed on DFDs, which provide a high-level architectural view of the software.	Design factor for threat modeling
166	10.1016/j.jss.2018.06.073	Tool category	For instance, Almorsy et al. (2013); Berger et al. (2016); Chen et al. (2007) and Tøndel et al. (2010) present formalized rules to extract knowledge from public repositories of threats and vulnerabilities namely Common Weakness Enumeration (CWE) (Martin, 2007) , Common Attack Pattern Enumeration and Classification (CAPEC) (Barnum, 2008), Open Web Application Security Project (OWASP) (Category: attack - owasp, 2017;Category:vulnerability - owasp, 2017).	Tool for threat modeling
167	10.1016/j.jss.2018.06.073	Design factor	In our opinion, there are three areas where existing techniques could be improved in order to cater to the needs of DevOps. First, it is important that the information that was gained from threat analysis is automatically propagated to source code level (and vice-versa). It might be beneficial to assure the traceability between the threats and corresponding security requirements at the level of implementation. This might facilitate a more efficient reuse of analysis outcomes in the fast-changing code base. Establishing a traceable link between architectural design and implementation can be achieved with a "top-down" or "bottom-up" approach.	Design factor for threat modeling
168	10.1016/j.jss.2018.06.073	Activity	In a "top-down" approach, the architectural design decisions need to be annotated in the source code (e.g. as presented by Abi-Antoun and Barnes, 2010). Such annotations may have to be added manually by developers themselves, which could render the technique unreliable. Therefore, there are existing approaches to extract the architecture from the code base (i.e. Software Architecture Reconstruction (SAR)) by employing dynamic and/or static reverse engineering techniques (e.g. as presented by Granchelli et al., 2017). To the best of our knowledge, the existing tools supporting SAR have limitations and are not commonly applied to practice. From a usability perspective, practices such as continuous deployment cause uncertainty in the security implications of modified code base. For instance, it would be beneficial for developers to get instant feedback on how their contribution impacts the security of the code base (e.g. one threat is mitigated).	

ID	DOI	Code	Excerpt	Notes
169	10.1016/j.jss.2018.06.073	Design factor	Second, the existing techniques would benefit from guidelines of how to compose the analysis outcomes. In practice, the software systems under analysis are too large and complex to be analyzed at once. Therefore, organizations are forced to scope the system into sub-systems and assign the analysis to several teams of experts to be analyzed simultaneously. As a result, border elements are either analyzed multiple times, or overlooked. One possible solution could be to scope the system according to assets. In this case, elements handling certain assets would be analyzed together in an end-to-end manner. To facilitate the composability of analysis outcomes, a level of formalism could be beneficial. For example, taint analysis has been used to analyze applications in order to present potentially malicious dataflows to the human analyst. The analyst (or automated malware detection tool) is able to decide whether particular flows constitute a policy violation.	Design factor for threat modeling
170	10.1016/j.jss.2018.06.073	Design factor	Third, the analysis performed for one subsystem is related to security assumptions, which may not be in line with the security assumptions of another subsystem. Further, threats with high impacts to the organization are typically prioritized. Threat prioritization is commonly still performed manually, which demands a lot of resources. Therefore, existing analysis techniques need to invest in impact analysis automation.	Design factor for threat analysis
171	10.1016/j.jss.2018.06.073	Design factor	To conclude, in light of DevOps and Agile, where software development is driven by change, there are three important aspects where existing analysis techniques have yet to mature: (i) traceability of analysis in the code base, (ii) composability of analysis outcomes and (iii) threat impact analysis automation.	Design factor for threat analysis
172	10.1049/iet-sen.2018.5293	Activity	With the aim to address (multi)Cloud risk challenges, we adopt the SLA-based approach where we rely on the existence of a Security SLA (and PLA) associated to each application component that is not under control of the designer and is offered as a service (i.e. is a Cloud service consumed by the application).	
173	10.1049/iet-sen.2018.5293	Framework	The risk assessment process in MUSA is considered the key driver to Cloud services selection decision support.	
174	10.1049/iet-sen.2018.5293	Design factor	The risk assessment process in MUSA is considered the key driver to Cloud services selection decision support. Depending on whether the multiCloud application processes PII, risks may involve not only security concerns but also risks to data privacy. MUSA promotes that risk evaluation is a continuous task where multiple perspectives of organization roles should take part.	
175	10.1049/iet-sen.2018.5293	Activity	risk model based on the OWASP threat risk modelling	

ID	DOI	Code	Excerpt	Notes
176	10.1049/iet-sen.2018.5293	Tool category	These threats may be chosen from a threat catalogue such as that included in the MUSA Security Metrics Catalogue, which describes potential threats to different service types taken from expert sources such as the OWASP TOP 10 threats catalogue.	
177	10.1049/iet-sen.2018.5293	Tool category	The security threats selected are classified in the STRIDE categories (Spoofing identity, Tampering, Repudiation, Information disclosure, Denial of service and Elevation of privilege).	
178	10.1049/iet-sen.2018.5293	Design factor	The DevOps team is required to provide the likelihood and impact of each threat and the composite risk index (CRI) of each threat ...	Design factor for threat modeling
179	10.1049/iet-sen.2018.5293	Tool category	In our approach, we have leveraged the ROAM model risk mitigation classification. ROAM is a common agile management risk mitigation classification that, according to the countermeasures applied, classifies each threat as: (i) Resolved, in case the risk has been answered, avoided or eliminated; (ii) Owned, for risks that have been allocated to someone who has responsibility for addressing them; (iii) Accepted, if the risk has been accepted and no further actions are required to address it; (iv) Mitigated, if an action has been taken (i.e. controls are set) to mitigate the risk, either reducing its likelihood or reducing its impact.	
180	10.1049/iet-sen.2018.5293	Activity	Cloud service selection on the basis of offered controls. Once the risk profile is finished, a Cloud service match-making process starts to select the services that best match the controls and the requirements in the CPIM model.	
181	10.1049/iet-sen.2018.5293	Activity	Creation of security SLA for multiCloud	
182	10.1049/iet-sen.2018.5293	Activity	Continuous monitoring of composed SLA fulfilment. The security and privacy levels promised to multiCloud application customers are continuously under scrutiny by DevOps team who keep tracking whether the metrics defined for the controls are reaching the target levels (SLOs).	
183	10.1109/smartcloud.2017.21	Design factor	These mechanisms are employed to identify and resolve vulnerabilities in computer systems and applications e.g. the number of vulnerabilities in applications could be reduced by integrating security assessments techniques in Continuous Development (CD) pipelines	Design factor for security automation
184	10.1109/smartcloud.2017.21	Design factor	SecaaS follows after the Software as-a-Service (SaaS) model by leveraging the cloud to provide security services. Hence, the services are self-managed, automated and scalable.	Design factor for security automation
185	10.1109/smartcloud.2017.21	Activity	A possible approach consists in training DevOps on security roles. But this approach is not optimal and may be limited given the requirement for security expertise.	

ID	DOI	Code	Excerpt	Notes
186	10.1109/ smartcloud.2017.21	Design factor	SecaaS could be integrated into CD pipelines to resolve the aforementioned concerns (see Figure 1). API calls against SecaaS could automate security tasks, including security analysis and reports generation.	Design factor for security automation
187	10.1109/ smartcloud.2017.21	Activity	For example, security scanners can be used to scan target VMs for check for vulnerabilities prior to application deployment	
188	10.1109/ ares.2016.92	Design factor	automated monitoring, automated deployment pipeline, and automated testing contribute positively to the security of the software	DevOps generic
189	10.1109/ ACCESS.2017.26856 29	Activity	Having proposed a secure build server, they encapsulated build jobs using virtualization environment with snapshot capability to prevent one project's security attacks from infecting other projects' build jobs in multitenant CI systems.	DevOps generic
190	10.1109/ ACCESS.2017.26856 29	Activity	which integrates security design fragments (i.e., security patterns) through four compassion primitives namely connect tactic, disconnect tactic, create tactic, and delete tactic to secure deployment pipelines	DevOps generic
191	10.1145/3098954.31 03170	Activity	Institute security awareness program	
192	10.1145/3098954.31 03170	Activity	Monitor security practices	
193	10.1145/3098954.31 03170	Activity	Perform security analysis of requirements	
194	10.1145/3098954.31 03170	Activity	Specify resource-based security properties	
195	10.1145/3098954.31 03170	Activity	Requirements inspection	
196	10.1145/3098954.31 03170	Activity	Risk estimation	
197	10.1145/3098954.31 03170	Activity	Threat modeling	
198	10.1145/3098954.31 03170	Design factor	Detail misuse cases	
199	10.1145/3098954.31 03170	Activity	Perform security analysis of system design	
200	10.1145/3098954.31 03170	Activity	Coding standards	
201	10.1145/3098954.31 03170	Activity	Pair programming	

ID	DOI	Code	Excerpt	Notes
202	10.1145/3098954.31 03170	Activity	Integrate security analysis into build processes	
203	10.1145/3098954.31 03170	Activity	Perform software security function usability testing	
204	10.1145/3098954.31 03170	Activity	Perform SW security fault injection testing	
205	10.1145/3098954.31 03170	Activity	Perform source level security reviews	
206	10.1145/3098954.31 03170	Activity	Perform code signing	
207	10.1145/3098954.31 03170	Activity	Repository improvement	

# Appendix C: thematic analysis details

## Thematic analysis on security activities

Activity	Excerpt IDs
Performing continuous feedback from production to development	99,100,104,117,122,123,144
Provide security training	94,106,185
Establish security satellites	111,115,146
Practice incident response	74,126,142
Performing automated security testing	6,14,70,75,76,77,82,85,89,129,136
Performing automated run-time testing	10,12,78
Performing automated static testing	37,85,89,114
Integrate security tests in unit testing	41
Performing automated software composition analysis	6,47,91,119,125
Implement automated remediation	25,33
Implement automation of software licensing	40
Performing security configuration automation	39,73,76,114
Performing security requirements analysis	23,46,51,91,114,116,137,180
Performing threat modeling	54,69,114,151,175
Performing risk analysis	7,23,53,68,102,103
Establishing security SLA's for cloud providers	23,28,30,172,181
Performing continuous monitoring	77,85,129
Performing continuous monitoring of security SLA's	32,34,182
Performing continuous monitoring of security metrics throughout the SDLC using CI/CD tooling	62,63,101
Performing continuous monitoring of system metrics using automated tools	38
Performing continuous monitoring of security controls	71
Performing continuous monitoring of application behaviour	120,121
Provide self-service monitoring capabilities to dev and ops	127
Performing continuous assurance	48

Activity	Excerpt IDs
Performing manual security testing	
Performing manual penetration testing	12,52,114,141
Performing manual security review	89,114
Performing automated security testing of the CI/CD pipeline	189,190

## Thematic analysis on design factors

Activity	Design factor	Excerpt IDs
Performing automated Security Testing	Leverage SecaaS by using cloud provided self-managed, automated and scalable security services	184, 186
Performing automated Security Testing	Integrate the security tools in an automated deployment pipeline	140, 183, 188
Performing automated Security Testing	Automate as many security controls and verifications as possible	60, 61
Perform automated run-time testing	Perform automated run-time testing at four levels: (1) pre-authentication scanning, (2) post-authentication scanning, (3) independent backend scanning and (4) complete workflows	79
Perform automated run-time testing	Ensure automated run-time testing is implemented for a broad scope of test scenarios	18
Perform automated run-time testing	Ensure proper unit tests are in place to optimise run-time testing efficiency	20
Performing automated static testing	Minimise the number of false positives resulting from static testing	16
Performing security requirements analysis	Treat security requirements as nonfunctional requirements	138
Performing security requirements analysis	Leverage metrics gathered during the security requirements analysis phase to evaluate the security level of alternative designs	97
Performing security requirements analysis	Enable the evaluation of alternative designs through suitable metrics during security requirements analysis to determine variations in security levels of a given design and make appropriate choices	97



Activity	Design factor	Except IDs
Performing security requirements analysis	Leverage goal-oriented requirements analysis (GORE) to perform security requirements analysis	160
Performing threat modeling	Perform threat modelling from a risk-centric perspective	163
Performing threat modeling	Perform threat modelling from an attack-centric perspective	164
Performing threat modeling	Perform threat modelling from a software-centric perspective	161, 165
Performing threat modeling	Ensure compatibility of threat modelling outcomes from a scope and result perspective	169, 171, 178
Performing threat modeling	Introduce abuse cases and problem frames to perform threat modeling	153, 156, 159
Performing threat modeling	Make use of attack or threat trees to perform threat modelling	153, 157
Performing threat modeling	Implement traceability of threat modelling (results) in the code base	167, 171
Performing threat modeling	Automate threat impact analysis	170, 171
Performing risk analysis	Performing risk analysis continuously before each iteration	26
Performing risk analysis	Performing risk analysis during the design phase	174
Performing continuous monitoring	Ensure continuous monitoring covers a wide range of resources and metrics including logical security, availability and intrusions	72, 133
Performing continuous monitoring	Leverage monitoring as code to establish a versioned and repeatable deployment of monitoring infrastructure	128
Performing manual security testing	Limit manual penetration testing to critical components or perform in parallel to reduce impact on deployment lead times	13
Performing continuous feedback from production to development	Set a time limit for all lower-priority security defects	105
Performing continuous feedback from production to development	Give attack patterns to your developers	145

Activity	Design factor	Except IDs
Performing continuous feedback from production to development	Build an internal forum to discuss attacks	118
Performing continuous feedback from production to development	Establish emergency code base response	124
Performing continuous feedback from production to development	Incorporate security tests as part of QA for detected incidents	152
Providing security training	Teach every developer enough to enable them to identify areas where they would benefit from the advice of an expert	107
Performing threat modeling	Perform threat modelling from a software-centric perspective	161, 165
Performing threat modeling	Ensure compatibility of threat modelling outcomes from a scope and result perspective	169, 171, 178
Performing threat modeling	Introduce abuse cases and problem frames to perform threat modeling	153, 156, 159
Performing threat modeling	Make use of attack or threat trees to perform threat modelling	153, 157
Performing threat modeling	Implement traceability of threat modelling (results) in the code base	167, 171
Performing threat modeling	Automate threat impact analysis	170, 171
Performing risk analysis	Performing risk analysis continuously before each iteration	26
Performing risk analysis	Performing risk analysis during the design phase	174

Activity	Design factor	Except IDs
Performing continuous monitoring	Ensure continuous monitoring covers a wide range of resources and metrics including logical security, availability and intrusions	72, 133
Performing continuous monitoring	Leverage monitoring as code to establish a versioned and repeatable deployment of monitoring infrastructure	128
Performing manual security testing	Limit manual penetration testing to critical components or perform in parallel to reduce impact on deployment lead times	13
Performing continuous feedback from production to development	Set a time limit for all lower-priority security defects	105
Performing continuous feedback from production to development	Give attack patterns to your developers	145
Performing continuous feedback from production to development	Build an internal forum to discuss attacks	118
Performing continuous feedback from production to development	Establish emergency code base response	124
Performing continuous feedback from production to development	Incorporate security tests as part of QA for detected incidents	152
Providing security training	Teach every developer enough to enable them to identify areas where they would benefit from the advice of an expert	107

## Appendix D: Expert interviews

### Interview 1: Cyber Security Designer

**Organisation:** Belgium-based financial services company that specialises in the settlement of securities transactions as well as the safekeeping and asset servicing of these securities. >1000 staff members.

**Role:** Cyber Security Designer. Aligns security practices to agile methodology, including DevOps.

**Experience:** More than 10 years of experience in Security and Architecture.

The interviewee mentions that his organisation is a highly regulated organisation requiring compliance with a wide variety of different norms and standards: CSTR (Cloud Security Thread Report), SWIFT standards, GDPR, French Military Law, etc. Requirements defined by these norms are collected in a compliance rule database, where they are combined with specific risk controls.

The organisation is at the beginning of its DevOps journey, today application development does not utilise Continuous Integration (CI) practices and is starting to apply agile methodologies throughout its activities. There is pressure to speed up current software delivery process to become faster and more responsive. At this point there are no clear practices related to Dev(Sec)Ops defined within the organisation, from a security perspective all projects are treated in a similar fashion. The interviewee also warns that whichever practices and activities are selected and implemented the challenge of proving regulatory compliance remains due to the complexity of mapping outcomes of security activities to these requirements.

The interviewee mainly considers DevSecOps from the point of view of automation, the path forward for the organisation is to incorporate automated security activities (static analysis, penetration testing, etc.) in the existing development activities allowing faster and more efficient feedback loops. Some of the activities which are considered valuable in the context of the organisation are:

- Automation of code review: The interviewee believes this is fairly easy to do however humans will still be required for more in depth tasks
- Automation of testing: The interviewee mentions that automated boundary testing would be beneficial to security
- Gathering and tracking of compliance requirements
- Definition of security policies and acceptance criteria
- Design review and threat modelling

Today the organisation already invests in the software security education of developers by providing secure coding standards & guidelines and by providing secure coding training materials. However

due to the fact that development activities are performed by external parties they have little control during the development cycle itself.

While discussing the diagram the interviewee pointed out that the ideation phase during which many of the security architecture activities are performed can be considered outside of the infinite loop used to represent the Dev(Sec)Ops process. It is in this step that activities such as compliance requirements and risk assessments are performed. He also stresses that common knowledge dictates that about half of the application security issues are related to errors and mistakes during the development phases and the other half is due to design and architecture.

The interviewee also mentioned that the organisation has no current plans to integrate the operations team and activities in this approach as this is perceived as being too high risk and outside of the comfort zone of the organisation. As this may be the case in the future, he also points out that the separation between development and operations teams used to be a security control within organisations and that a tight integration may also introduce new security challenges. The interviewee assumes that a DevSecOps approach will not have impact on the high-level (Security) Architecture activities being performed today.

When asked the interviewee confirms that for the organisation and the industry at large security is perceived as an obstacle for successful DevOps implementations. The interviewee also points out that security was not really part of the Agile movement and that it is only now

Today the organisations' security architecture team focusses its efforts on adapting the existing security-related processes to the Agile way of working. The traditional approach within the organisation was to prepare a detailed security architecture document at the beginning of each project. This document was then circulated between the different stakeholders for review and approval. In each Agile iteration, this document needed to be refined and reviewed, which resulted in a cumbersome process, requiring a lot of effort and resulting in time delays.

The security architecture team changed its methodology to a model-based architecture following the principle of "documenting as far as you can see" and refine through iterations. Security architecture is now generated from models, which reduces documentation overhead. They use the ArchiMate modelling language (Archi as a tool) to define the models.

Another challenge for the organisations transition to Agile methodologies are contracts with external suppliers. The contracts are defined based on the fixed scope for 5 years, which is difficult to determine because in Agile the scope is supposed to be flexible.

The interviewee recognises that a Dev(Sec)Ops culture could further help optimising the use of available resources. For instance, expensive external pen testers today are mainly occupied with the detection of trivial coding errors (e.g., checking input boundaries). He expects that by applying (automated) security practices throughout the development cycle these "low-hanging fruits" can be covered earlier on in the process allowing security experts to dedicate their efforts to detecting complex security issues or dependent to the business process dependent weaknesses.

The interviewee also mentions that the most important challenge faced by the organisation today is the organisational culture concerning security. There is a constant struggle between business stakeholders and security architecture, even within highly regulated organisations. The business stakeholders consistently prioritise features over security in part due to the way performance is measured and rewarded. This is reinforced by Agile methodologies which enable the business stakeholders to do exactly that. As such this conflict is not new to Agile but is rather being reinforced by it. He warns however in the long term this leads to the accrual of technical debt. A lack of structure can lead to functionality (new or changed) cannot longer be implemented because of the excessive complexity. One potential way to tackle this problem according to the interviewee would be to establish rules so that a certain percentage of effort in each sprint should be dedicated to paying off technical debt.

---

## Interview 2: Cyber Security Designer

---

**Organisation:** Large governmental institution.

**Role:** Cyber Security Assurance Analyst. Assist in the definition of software security assurance and DevSecOps practices institution.

**Experience:** More than 19 years of experience in Information Security ranging from technical to governance aspects.

At the start of the interview, when asked about his take-aways related to DevSecOps, the interviewee mentions that DevSecOps is an incredibly broad field. Therefore he states that DevSecOps should be mainly characterised by the concept of automation. Activities such as developing applications or performing manual penetration testing are best compared to forms of art and therefore require considerable time and effort making them slow. DevSecOps does not aim to replace activities such as manual penetration testing but rather seeks to implement additional supporting measures which can be automated reducing the time required to test something. They should be seen as complementary to manual activities and, by performing the low level work, these automated activities may reduce the time required for the manual activities.

The real challenge in DevSecOps according to Frederic is that in general automated security testing such as performed by Dynamic Application Security Testing tools delivers very poor results. This type of technology does not play very well with newer technological evolutions such as Single Page Applications, new approaches to authentication and so on. A potential way forward would be the use of Interactive Application Security Testing (IAST) solutions however the interviewee points out that these rely on the coverage of the functional tests performed by the testing team. Testing through IAST may therefore prove incomplete as testing teams rarely cover the full breadth of functionality in an application. This leads the interviewee to the conclusion that one of the most important aspects of web application security is very difficult to cover through automation. The interviewee points out that for high value applications or components manual verification activities makes perfect sense

---

however he does not see why one would not apply the automated tests as a security baseline across all applications.

According to the interviewee it is a true challenge in larger organisations to embrace true DevSecOps practices. Concepts such as infrastructure-as-code, continuous deployments and close collaboration between teams are difficult to implement throughout large organisations. He mentions that many organisations claim to do DevSecOps while in reality few manage to get it right at this point in time.

When looking at the process part of things he states that in his current organisation the implementation of DevSecOps is being pushed bottom-up with the various teams looking to interface with each-other on the technical side of things. This leads to teams integrating aspects of their work on an ad-hoc basis which does not allow a true integration as would be possible when driven top-down. This situation can be explained by the traditional segregation between development and security / penetration testing teams in the organisation, they used to live each in their own worlds. Frederic points out that the only way forward would be to embed the security experts in the development teams.

Currently he tries to push for a ratio of 1 application security specialist for every 100 developers which he considers a minimalistic starting point. The organisation currently designates so called security experts (security champions) within some development teams who are considered responsible for security in their team. However these people can not be considered as true application security experts due to the significant knowledge gap. A transversal team of security experts supporting the security leaders across the various development teams would make sense from his perspective. The current approach of the organisation is to leverage SecureCodeWarrior (security training solution) combined with internal tournaments to create a sense of community. An important aspect to promote security practices are relational mechanisms such as meet-ups, leaderboards and tournaments.

When talking about threat modelling the interviewee points out that development teams are asking for visual tools allowing them to input data-flows and receive relevant threats and security requirements. These types of tools can gain in effectiveness when combined with security testing based on the identified security requirements. Overall the importance of security architecture should not be underestimated.

Looking at the tooling landscape in the organisation they currently implemented a secure coding training platform based on Secure Code Warrior while leveraging Fortify SAST and DAST tools for automated scanning. Software Composition Analysis is performed using OWASP dependency check and container security scanning is performed using tenable.io. The results from these tools are reported in the Fortify Software Security Center which provides vulnerability correlation and allows a link to the training platform to allow developers to obtain additional knowledge related to the findings. The findings from the vulnerability correlation platform can be exported to Jira for easy accessibility.

Overall the interviewee believes that in general developers welcome the availability of security knowledge and tools. The best approach is to provide access to security knowledge and tooling for

---

the development community while cultivating an application security mindset. Involving management by increasing their understanding and ensuring they give quality a place in the organisation is an important success factor to establish such a mindset. The interviewee wants to make an attempt at creating this management awareness by exposing management to security challenges in pseudo code during the next secure coding competition. Another enabler for this mindset could be the publication of team leaderboards.

---

### Interview 3: DevSecOps advisor

---

**Organisation:** Various customers including large financial institutions.

**Role:** DevSecOps advisor.

**Experience:** Five years of experience with information security and specialised in DevSecOps

The interviewee states that many organisation are facing challenges when implementing security in the software development cycle. They don't know how to approach this in a scalable way, an example of this is that traditional manual penetration testing does not really fit well with DevOps. Another aspect are the difficulties related to infusing existing DevOps teams with security knowledge.

This leads us to one of the obstacles for DevSecOps which, according to the interviewee, is awareness about security topics combined with the required level of knowledge to get it right. According to him the people aspect in DevSecOps should therefore be the main focus of any successful approach. In practice he sees very often that a bottom-up approach is used to influence culture, for example tools or processes are leveraged to generate quick wins. However the ultimate goal should be establish a security mindset in the organisation.

The lack of people with security knowledge is an important driver for automation of security activities. It is crucial to leverage automation to reduce the skill gap. The key to effective automation is to start from the process and build from there. The tools should be there to facilitate and automate the process instead of processes being built around tools which does not work very well.

Coming back to culture and mindset it is important to emphasise that the developers understand the why. Without this mind-shift there will be friction with the development and operations teams leading them to fight against the security tools. A way to achieve this is to ensure stakeholders receive relevant output from the automated processes so they can understand why something happens and to provide them relevant knowledge on the topic of security. This communication is a key enabler for adoption of DevSecOps processes.

Every organisation should have secure coding standards however when trying to create a security mindset it is also useful to convey impact by relating to real-world incidents and cases. Try to place as many things as possible in context for the participants.

When asked about threat modeling the interviewee sighs and mentions that it is a difficult subject in DevSecOps, it is hard to get it right. You need knowledgeable people to do the threat modeling and

you need attention and time from the team(s) to make it valuable. Theoretically the approach to start from threat modeling makes sense however in practice it is not easy to implement. Threat modeling automation could be a way of facilitating this. The technique is promising if it becomes feasible to do with limited time available. Overall he sees issues in the scalability aspects of threat modelling.

The same issue applies to security requirements analysis, how to scale this? In many cases security requirements are poorly documented and if document rarely consulted by the people doing the actual implementation. The key is to find an approach to consume security requirements for a DevSecOps team. A combination of automated validation and stop on fail can push teams to implement the required security measures to successfully complete the build. For example open policy agent can be leveraged to automate decision making in automated tests leading to compliance as code. Solutions such as Open Policy Agent make sense because they operate on a sufficiently abstract level allowing them to cover a wide range of technologies.

When asked about important DevSecOps enablers the interviewee mentions that SAST and SCA are cornerstones of security automation. DAST however is very difficult to automate leading to limited automation capabilities. It is an essential component but as it stand it's not sufficiently mature to be integrated in a DevSecOps pipeline. DAST relies on a complete coverage of the applications functionality through tests, without proper functional testing there is no good way to leverage DAST. AIAST may be more a marketing approach to distinguish themselves from other DAST solutions.

Another aspect which deserves attention are deployment aspects such as cloud security, infrastructure as code and secret management are an integral part of DevSecOps. Container security is an important aspect which is regularly neglected. The increasing complexity of the overall stack increases risks from a security perspective.

Getting and maintaining an overview of cloud resources is a huge challenge. A lot of security knowledge is required to understand which controls you need, how they need to be configured and maintaining a view on their effectiveness. There is knowledge available on cloud controls however it remains a manual and research heavy process. There are quite some open-source and commercial tools available however they usually cover only part of the solution resulting in a mesh of many different tools and solutions. The networking aspect of multi-hybrid clouds is very complex and pushed towards teams who were previously not concerned with these aspects.

With all these tools in place maintaining oversight can be quite challenging. Correlating the wide range of metrics gathered throughout the CI/CD pipeline is another current aspect. There is hope that machine learning can play an important role here. Therefore solutions such as vulnerability correlation engines are best created as bespoke solutions tailored to the organisation using them.

Bringing the various pieces of DevSecOps together is the most challenging of all. DevSecOps can only be feasible by bringing together experts in the different domains and providing this knowledge to a wide range of actors (development and operations). Training the people involved in the DevSecOps process is key. Overall the interviewee considers security to be a quality attribute, without quality security is not achievable.

## Appendix E: knowledge nomination worksheet

ID	Name	Category	Experience	Iteration	Survey		Interview		GSS	
					I	P	I	P	I	P
1	-----	Practitioner	+10 years experience MsC	1	Y	Y	Y	Y	Y	N
2	-----	Practitioner	+10 years experience MsC	1	Y	N	Y	N	N	N
3	-----	Practitioner	+10 years experience MsC	1	Y	Y	Y	Y	Y	Y
4	-----	Academic	Author of gold set paper	2	Y	N	Y	N	N	N
5	-----	Academic	Author of gold set paper	2	Y	N	Y	N	N	N
6	-----	Academic	Author of gold set paper	2	Y	N	Y	N	N	N
7	-----	Academic	Author of gold set paper	2	Y	N	Y	N	N	N
8	-----	Practitioner	+10 years experience MsC	3	Y	Y	Y	N	Y	N
9	-----	Practitioner	+10 years experience MsC	3	Y	Y	Y	N	Y	N
10	-----	Practitioner	+10 years experience MsC	3	Y	N	Y	N	N	N
11	-----	Academic	Author of gold set paper	3	Y	Y	Y	N	Y	N
12	-----	Practitioner	+10 years experience MsC	3	Y	N	Y	N	Y	Y
13	-----	Practitioner	+10 years experience MsC	3	Y	Y	Y	N	Y	N
14	-----	Practitioner	+10 years experience MsC	3	Y	N	Y	N	N	N
15	-----	Practitioner	+10 years experience MsC	3	Y	Y	Y	N	Y	N
16	-----	Practitioner	+10 years experience MsC	3	Y	N	Y	N	N	N

ID	Name	Category	Experience	Iteration	Survey		Interview		GSS	
					I	P	I	P	I	P
17	-----	Practitioner	+10 years experience MsC	3	Y	N	Y	N	N	N
18	-----	Practitioner	+10 years experience MsC	3	Y	N	Y	N	N	N
19	-----	Practitioner	+10 years experience MsC	3	Y	Y	Y	Y	Y	Y
20	-----	Practitioner	+6 years experience Bachelor	3	Y	Y	Y	Y	Y	N
21	-----	Practitioner	+10 years experience Bachelor	3	Y	Y	Y	N	N	N
22	-----	Practitioner	+10 years experience Bachelor	3	Y	N	Y	N	N	N
23	-----	Academic	Author of gold set paper	2	Y	N	Y	N	N	N
24	-----	Practitioner	+10 years experience PhD	4	N	N	N	N	Y	Y
25	-----	Practitioner	+10 years experience MsC	4	N	N	N	N	Y	Y
26	-----	Practitioner	+10 years experience MsC	4	N	N	N	N	Y	Y
27	-----	Practitioner	+10 years experience MsC	4	N	N	N	N	Y	Y
28	-----	Practitioner	+10 years experience MsC	4	N	N	N	N	Y	Y

## Appendix F: analysis of expert elaborations

Expert input	Expert ID	Extracted	Type	Related to
i have mentioned mindset couple of times..and also management support and understanding are very important.	8	Establish a security mindset	L1 Activity	
		Management support	Enabler	
		Train management on security concepts	Design factor	Establish a security mindset
Give centralized training sessions/ workshops. Scan all of your registries (with artifacts and source code) for vulnerabilities, make centralized dashboards, reward the teams who fixes the most vulnerabilities, share best practices all across the organization. Build quality levels for all programming languages - use the same levels so all languages are treated the same.	9	Perform centralised training sessions and workshops	Design factor	Provide security training
	9	Establish artefacts and source code registries which are automatically scanned for vulnerabilities	L2 Activity	Manage Digital Supply Chain
	9	Establish centralised dashboards	L2 Activity	Performing continuous monitoring
	9	Reward the teams who fix the most vulnerabilities	Design factor	Establish a security mindset
	9	Share best practices all across the organisation	Design factor	Establish a security mindset
	9	Establish a code quality standard for each language	Design factor	Establish a security mindset
to be found at (controls with regard to DevOps): <a href="https://secursoftwarealliance.org">https://secursoftwarealliance.org</a> <a href="https://www.norea.nl/nieuws/6043/studierapport-devops-and-agile-in-control-gepubliceerd">https://www.norea.nl/nieuws/6043/studierapport-devops-and-agile-in-control-gepubliceerd</a>	10			
Maybe something around infrastructure as code / compliance as code more in depth. Also container security and secrets management more in depth.	13	container security	L2 Activity	Manage Digital Supply Chain
		Leverage compliance-as-code	Design factor	Performing continuous assurance
		Container security	L2 Activity	Manage Digital Supply Chain
		Implement secret management	L1 Activity	

Expert input	Expert ID	Extracted	Type	Related to
MBSE	14	Leverage Model-based systems engineering	Design factor	Performing security requirements analysis

Activity	Design factor	Expert ID
Establish a security mindset	Train management on security concepts	8
Establish a security mindset	Reward the teams who fix the most vulnerabilities	9
Establish a security mindset	Share best practices all across the organisation	9
Establish a security mindset	Establish a code quality standard for each language	9
Establishing security satellites (champions)	in the initial push from the current way of working to 'Security to Left' principle.	8
Establishing security satellites (champions)	Should not be forced but discovered.	13
Establishing security SLA's for cloud providers	I went for yes here because you should have a say in defining some security controls relevant (which might not be covered by the SOC2 or ISAE 3402)	8
Establishing security SLA's for cloud providers	Partly true, there is very little influence in cloud providers. Security always remains your own end-responsibility, whatever a cloud provider provides. It does not say "it's them" to handle security and the SLA protects me. If your data got stolen because of a lack of security at your cloud provider, no one will give you back your data...	9
Implement automated remediation	Always manually check the results from time to time.	9
Implement automated remediation	Not seen this effective yet in practice.	13
Implement automation of software licensing	So true of open source software.	9
Integrate security tests in unit testing	Important to focus on critical first, then highs. Consider the teams needs to have special security related knowledge. Start early on to avoid bottlenecks when deploying to production.	9
Performing automated run-time testing	Start small, extend later on.	9
Performing automated run-time testing	Quite challenging to implement properly when DevOps is not at a proper level.	13
Performing automated security testing	Security mindset of the team in my opinion is one of the most important factor in making DevSecOps team successful. Also, management support is of utmost importance.	8
Performing automated security testing	Fail fast when security stuff is not met.	9

Activity	Design factor	Expert ID
Performing automated security testing	The above should be elements of the Definition of Done. There are many more controls.	10
Performing automated security testing	Good APIs that can facilitate organizational processes. The implementation should be easy to understand by developers.	13
Performing automated security testing	Automated testing is better at finding implementation bugs than design flaws.	14
Performing automated security testing of the CI/CD pipeline	Also check for naming conventions and design best practices.	9
Performing automated software composition analysis	(Make use of) Approved libraries	14
Performing automated software composition analysis	Most critical (end-user facing applications) first.	9
Performing automated software composition analysis	Process to handle new vulnerabilities that pop up suddenly is important.	13
Performing automated static testing	Independent justification of whether a false positive is really false and whether the fix genuinely addresses the cause	14
Performing automated static testing	Ensure code coverage (applications should be tested as a whole, not scans run on separate modules, otherwise the static testing will not properly follow flows).	7
Performing automated static testing	Should prioritise the most relevant applications first since this takes a lot of time.	9
Performing automated static testing	Good process is key, build breakers are important.	13
performing continuous monitoring of application behaviour	as a DevOps team you should be aware of the normal application behaviour so as to identify any deviations there.	8
performing continuous monitoring of security controls	Especially in highly regulated environments in which controls change now and then and in which informal communication is difficult to track. Formal controls can and should be tracked.	9
performing continuous monitoring of system metrics using automated tools	Important, but not super high priority since other factors must be in place first to make this a reality (e.g. automated deployments).	9
Performing manual security testing	When having automated security at a proper level then finally the value of manual testing can really be of value	13
Performing manual security testing	since it is a very binary answer, i went for Yes. I would say that it really depends on the criticality of the application and the customisation that application has gone through.	8
Performing risk analysis	Depends on what is considered as risk analysis and scope	13
Performing security requirements analysis	Leverage process metrics	14

Activity	Design factor	Expert ID
Performing security requirements analysis	How to find the balance between functional and non-functional requirements. Now most of the time almost all of the effort goes to functional requirements/features.	9
Performing security requirements analysis	Leverage Model-based systems engineering	14
Performing threat modeling	It should be a very agile approach	13
Performing threat modeling	Quantitative vs Qualitative	14
provide self-service monitoring capabilities to DEV and OPS	security to the left	8
provide self-service monitoring capabilities to DEV and OPS	It can prevent escalations and helps the teams to be self-organised when it comes to fix any issues.	9
Providing security training	Very important..	8
Providing security training	Should be hands on.	13
Providing security training	educate teams how to refactor software, educate product owners this is very important, learn developers how to prioritise (risk factor versus impact versus likelihood of an exploit, etc). A cloud native service is not secure by default, learn that etc.	9
	Compliance	14
Provide security training	Perform centralised training sessions and workshops	9
Performing continuous assurance	Leverage compliance-as-code	13

## Appendix G: Results of prioritisation by experts

Aspect	Activity	1	2	3	4	5	6	7	8
Delay	Establish a security mindset across the organisation	1	4	4	2	4		4	3
Effectiveness	Establish a security mindset across the organisation	5	5	3	4	4	3	4	5
Financial	Establish a security mindset across the organisation	4	4	3	1	4		1	3
delay	Establish artefact and source code registries which are automatically scanned for vulnerabilities	5	4	3	3	5		3	2
effectiveness	Establish artefact and source code registries which are automatically scanned for vulnerabilities	5	4	4	3	4	4	3	2
financial	Establish artefact and source code registries which are automatically scanned for vulnerabilities	4	4		4	4		2	1
Delay	Establish security satellites	4	4	4	2	4		3	4
Effectiveness	Establish security satellites	5	4	5	4	3	5	5	
Financial	Establish security satellites	2	4	2	3	4		2	3
Delay	Establishing security SLAs for cloud providers	5	5	4	1	5	2	5	3
Effectiveness	Establishing security SLAs for cloud providers	4	2		4	2	3	2	4
Financial	Establishing security SLAs for cloud providers	4	4	4	2	4	3	3	3
delay	Implement automated container security scanning	5	4	4	3	4		4	4
effectiveness	Implement automated container security scanning	5	3	3	4	4	5	3	4
financial	Implement automated container security scanning	5	3		3	4		3	4
delay	Implement automated remediation	5	4	5	3	5		5	5
effectiveness	Implement automated remediation	5	2	5	2	3	3	3	4
financial	Implement automated remediation	4	3		2	2		3	4
delay	Implement centralised dashboards	5	4	4	3	5		5	3
effectiveness	Implement centralised dashboards	5	4	3	2	4	3	2	5
financial	Implement centralised dashboards	4	4		4	4		4	4
delay	Implement secrets management	5	4	3	3	5		3	4



Aspect	Activity	1	2	3	4	5	6	7	8
effectiveness	Implement secrets management	5	3	3	5	5	4	3	5
financial	Implement secrets management	3	3		4	5		3	5
delay	Integrate security tests in unit testing	5	1	3	2	5		1	5
effectiveness	Integrate security tests in unit testing	5	5	4	2	5	5	3	2
financial	Integrate security tests in unit testing	1	1		1	3		1	4
delay	Performing automated run-time testing	5	2	4	4	4		4	3
effectiveness	Performing automated run-time testing	5	3	3	3	5	3	4	3
financial	Performing automated run-time testing	1	4		3	5		3	2
delay	Performing automated software composition analysis	5	4	3	3	3		4	2
effectiveness	Performing automated software composition analysis	4	3	3	2	3	5	4	2
financial	Performing automated software composition analysis	3	4		2	4		3	2
delay	Performing automated static testing	1	4	2	4	4		3	3
effectiveness	Performing automated static testing	1	4	3	5	5	3	3	5
financial	Performing automated static testing	1	2		3	5		3	2
Delay	Performing continuous assurance	5	4	3	2	4	2	3	2
Effectiveness	Performing continuous assurance	5	2		2	4	3	4	5
Financial	Performing continuous assurance	3	4	2	3	3	3	3	1
Delay	Performing continuous feedback from production to development	2	3	3	5	4		5	4
Effectiveness	Performing continuous feedback from production to development	5	4	3	5	5	5	4	2

Aspect	Activity	1	2	3	4	5	6	7	8
Financial	Performing continuous feedback from production to development	3	4	2	4	5		5	4
delay	Performing continuous monitoring of application behaviour	5	4	3	3	5		4	2
effectiveness	Performing continuous monitoring of application behaviour	5	4	4	3	4	4	4	3
financial	Performing continuous monitoring of application behaviour	4	2		2	4		2	2
delay	Performing continuous monitoring of security controls	5	4	3	3	5		4	4
effectiveness	Performing continuous monitoring of security controls	5	3	4	2	5	4	4	4
financial	Performing continuous monitoring of security controls	4	3		2	4		2	3
delay	Performing continuous monitoring of security metrics throughout the SDLC using CI/CD tooling	5	4	3	2	5		4	3
effectiveness	Performing continuous monitoring of security metrics throughout the SDLC using CI/CD tooling	5	3	3	4	4	3	4	4
financial	Performing continuous monitoring of security metrics throughout the SDLC using CI/CD tooling	4	3		2	5		3	4
delay	Performing continuous monitoring of security SLAs	5	5	3	2	5		4	3
effectiveness	Performing continuous monitoring of security SLAs	5	1	4	2		3	2	4
financial	Performing continuous monitoring of security SLAs	4	4		1	5		3	2
delay	Performing continuous monitoring of system metrics using automated tools	5	4	3	2	5		4	3
effectiveness	Performing continuous monitoring of system metrics using automated tools	5	2	2	4	5	4	4	5
financial	Performing continuous monitoring of system metrics using automated tools	4	3		2	4		3	3
Delay	Performing manual penetration testing	1	2	2	4	4	2	1	1
Effectiveness	Performing manual penetration testing	2	5		3	4	4	4	2

Aspect	Activity	1	2	3	4	5	6	7	8
Financial	Performing manual penetration testing	1	2	2	2	4	3	1	3
Delay	Performing manual security review	3	2	2	3	3	2	2	2
Effectiveness	Performing manual security review	4	3		2	2	4	3	2
Financial	Performing manual security review	1	3	3	4	4	3	2	2
Delay	Performing risk analysis	4	2	5	3	4	3	3	4
Effectiveness	Performing risk analysis	4	4		5	4	4	3	3
Financial	Performing risk analysis	4	4	4	4	4	3	4	3
delay	Performing security configuration automation	5	4	4	3	5		5	3
effectiveness	Performing security configuration automation	5	4	4	4	3	5	4	4
financial	Performing security configuration automation	4	2		4	4		2	2
Delay	Performing security requirements analysis	5	3	3	1	3	2	2	5
Effectiveness	Performing security requirements analysis	3	3		2	3	4	3	4
Financial	Performing security requirements analysis	5	4	4	2	2	3	4	2
Delay	Performing threat modeling	3	3	5	3	3	2	4	4
Effectiveness	Performing threat modeling	5	3		3	3	3	4	2
Financial	Performing threat modeling	4	3	4	3	3	3	3	2
Delay	Practice incident response	4	3	4	5	5		5	3
Effectiveness	Practice incident response	5	3	5	4	4	3	2	4
Financial	Practice incident response	4	3	3	3	5		3	5

Aspect	Activity	1	2	3	4	5	6	7	8
Delay	Provide security training	1	4	5	4	5		4	2
Effectiveness	Provide security training	5	3	3	4	4	3	4	4
Financial	Provide security training	1	3	3	3	4		3	2
delay	Provide self-service monitoring capabilities to dev and ops	5	4	3	3	5		5	4
effectiveness	Provide self-service monitoring capabilities to dev and ops	5	2	3	4	3	4	3	2
financial	Provide self-service monitoring capabilities to dev and ops	5	4		4	4		2	1
Delay	Securing the CI/CD pipeline	5	4	3	5	4	2	5	4
Effectiveness	Securing the CI/CD pipeline	4	4		5	3	3	3	5
Financial	Securing the CI/CD pipeline	5	4	3	4	4	3	2	2

---

## List of figures

Figure 1: Comparison between the typical process steps in waterfall and agile development as explained by Abrahamsson et al. [4]	2
Figure 2: Relationship between continuous integration, delivery and deployment according to M. Shahin et al. [5]	3
Figure 3: Mapping of IATF Information System Security Engineering Process (ISSE) to waterfall and agile process steps	5
Figure 4: Visualisation of the “shift left on security” concept based on the premises of [23] and [27]	6
Figure 5: Overview of design science as defined by Hevner [37]	8
Figure 6: Schematic overview of the structure for this research	10
Figure 7: Schematic overview of the research design	12
Figure 8: Schematic overview of the literature research	17
Figure 9: Screenshot of GSS session	34
Figure 10: Ranking scales used for prioritisation of security activities	35
Figure 11: Map of DevSecOps activities identified during this research	95

---

## List of tables

Table 1: Literature review search results	17
Table 2: Overview of codes used for thematic analysis	20
Table 3: Results of thematic analysis of security activities	21
Table 4: Results of thematic analysis of design factors	23
Table 5: Selection criteria for DevSecOps experts	26
Table 6: Results of contextual questions	28
Table 7: Results on the validation of the proposed definitions for DevOps and DevSecOps	28
Table 8: Results of validation of security activities by expert panel	30
Table 9: Results of validation of design factors by expert panel	32
Table 10: Results of the prioritisation of security activities by the expert panel	36
Table 11: Prioritised list of security activities	89

## References

- [1] Tallon, P., Queiroz, M., Coltman, T., Sharma, R. (2019). Information technology and the search for organizational agility: A systematic review with future research possibilities *The Journal of Strategic Information Systems* 28(2), 218-237. <https://dx.doi.org/10.1016/j.jsis.2018.12.002>
- [2] Forsgren, N., Humble, J. (2015). The Role of Continuous Delivery in it and Organizational Performance SSRN Electronic Journal <https://dx.doi.org/10.2139/ssrn.2681909>
- [3] Martin, F., Jim, H. (2001). Agile-Manifesto *Software Development* 9(8), 28-35.
- [4] Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J. (2017). Agile Software Development Methods: Review and Analysis <https://arxiv.org/abs/1709.08439>
- [5] Shahin, M., Babar, M., Zhu, L. (2017). Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices *IEEE Access* 5(), 3909-3943. <https://dx.doi.org/10.1109/access.2017.2685629>
- [6] Almeida, F., o (2017). Challenges in migration from waterfall to agile environments *World Journal of Computer Application and Technology* 5(3), 39-49.
- [7] Moe, N., Dings, T., Dyb, T. (2008). Understanding Self-Organizing Teams in Agile Software Development 19th Australian Conference on Software Engineering (aswec 2008) <https://dx.doi.org/10.1109/aswec.2008.4483195>
- [8] Artač, M., Borovšak, T., Nitto, E., Guerriero, M., Tamburri, D. (2017). DevOps: Introducing Infrastructure-as-Code 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C) <https://dx.doi.org/10.1109/icse-c.2017.162>
- [9] Kang, H., Le, M., Tao, S. (2016). Container and Microservice Driven Design for Cloud Infrastructure DevOps 2016 IEEE International Conference on Cloud Engineering (IC2E) <https://dx.doi.org/10.1109/ic2e.2016.26>
- [10] Jabbari, R., Ali, N., Petersen, K., Tanveer, B. (2016). What is DevOps?: A Systematic Mapping Study on Definitions and Practices <https://dx.doi.org/10.1145/2962695.2962707>
- [11] Ebert, C., Gallardo, G., Hernantes, J., Serrano, N. (2016). DevOps *IEEE Software* 33(3), 94-100. <https://dx.doi.org/10.1109/ms.2016.68>
- [12] Rahman, A., Williams, L. (2016). Software Security in DevOps: Synthesizing Practitioners' Perceptions and Practices <https://dx.doi.org/10.1145/2896941.2896946>
- [13] Duvall, P., Matyas, S., Glover, A. (2007). Continuous integration: improving software quality and reducing risk
- [14] Beznosov, K., Kruchten, P. (2004). Towards agile security assurance <https://dx.doi.org/10.1145/1065907.1066034>
- [15] Smeds, J., Nybom, K., Porres, I. (2015). DevOps: a definition and perceived adoption impediments
- [16] Technologies, CA. DevOps: The Worst Kept Secret to Winning in the Application Economy: 2014. <http://www.ca.com/us/~media/Files/whitepapers/devops-the-worst-kept-secret-to-winning-in-the-application-economy.pdf>
- [17] Chen, L. (2017). Continuous Delivery: Overcoming adoption challenges *Journal of Systems and Software* 128(), 72-86. <https://dx.doi.org/10.1016/j.jss.2017.02.013>
- [18] Mann, A., Brown, A., Stahnke, M., Kersten, N. (2018). Puppet - State of DevOps Report 2018
- [19] Elliot, S. (2014). DevOps and the Cost of Downtime
- [20] Mohan, V., Othmane, L. (2016). SecDevOps: Is It a Marketing Buzzword? Mapping Research on Security in DevOps 2016 11th International Conference on Availability, Reliability and Security (ARES) <https://dx.doi.org/10.1109/ares.2016.92>
- [21] McCarthy, M., Herger, L., Khan, S., Belgodere, B. (2015). Composable DevOps: Automated Ontology Based DevOps Maturity Analysis 2015 IEEE International Conference on Services Computing <https://dx.doi.org/10.1109/scc.2015.87>
- [22] Corman, J., Rice, D., Williams, J. (2010). Rugged Software manifesto
- [23] Jiménez, M., Rivera, L., Villegas, N., Tamura, G., Müller, H., Gallego, P. (2018). DevOps' Shift-Left in Practice: An Industrial Case of Application <https://dx.doi.org/10.29007/lh14>
- [24] Airaj, M. (2016). Enable cloud DevOps approach for industry and higher education: Enable cloud DevOps approach for industry and higher education Concurrency and Computation: Practice and Experience 29(5), e3937. <https://dx.doi.org/10.1002/cpe.3937>
- [25] Mansfield-Devine, S. (2018). DevOps: finding room for security *Network Security* 2018(7), 15-20. [https://dx.doi.org/10.1016/s1353-4858\(18\)30070-9](https://dx.doi.org/10.1016/s1353-4858(18)30070-9) ([https://dx.doi.org/10.1016/s1353-4858\(18\)30070-9](https://dx.doi.org/10.1016/s1353-4858(18)30070-9))
- [26] Colavita, F. (2016). Proceedings of 4th International Conference in Software Engineering for Defence Applications, SEDA 2015 [https://dx.doi.org/10.1007/978-3-319-27896-4\\_17](https://dx.doi.org/10.1007/978-3-319-27896-4_17)
- [27] Westland, J. (2004). The cost behavior of software defects *Decision Support Systems* 37(2), 229-238. [https://dx.doi.org/10.1016/s0167-9236\(03\)00020-4](https://dx.doi.org/10.1016/s0167-9236(03)00020-4) ([https://dx.doi.org/10.1016/s0167-9236\(03\)00020-4](https://dx.doi.org/10.1016/s0167-9236(03)00020-4))
- [28] Tashi, I. (2009). Regulatory Compliance and Information Security Assurance 2009 International Conference on Availability, Reliability and Security <https://dx.doi.org/10.1109/ares.2009.29>
- [29] Goel, S., Shawky, H. (2009). Estimating the market impact of security breach announcements on firm values *Information Management* 46(7), 404-410.
- [30] Sinanaj, G., Muntermann, J., Cziesla, T. (2015). How Data Breaches Ruin Firm Reputation on Social Media!-Insights from a Sentiment-based Event Study. *Wirtschaftsinformatik*
- [31] Hevner, March, Park, Ram (2004). Design Science in Information Systems Research *MIS Quarterly* 28(1), 75. <https://dx.doi.org/10.2307/25148625>
- [32] Recker, J. (2012). Scientific Research in Information Systems, A Beginner's Guide [https://dx.doi.org/10.1007/978-3-642-30048-6\\_1](https://dx.doi.org/10.1007/978-3-642-30048-6_1)
- [33] Wieringa, R. (2014). Design Science Methodology for Information Systems and Software Engineering [https://dx.doi.org/10.1007/978-3-662-43839-8\\_1](https://dx.doi.org/10.1007/978-3-662-43839-8_1)
- [34] Bobbert, Y. (2017). On Exploring Research Methods for Business Information Security Alignment and Artefact Engineering *International Journal of IT/Business Alignment and Governance (IJITBAG)* 8(2), 28-41. <https://dx.doi.org/10.4018/ijitbag.2017070102>
- [35] Bobbert, Y., Mulder, H. (2013). Group Support Systems Research in the Field of Business Information Security: A Practitioner's View <https://dx.doi.org/10.1109/hicss.2013.244>
- [36] Bruce, C. (1994). Research students' early experiences of the dissertation literature review *Studies in Higher Education* 19(2), 217-229. <https://dx.doi.org/10.1080/03075079412331382057>
- [37] Hevner, Alan R. (2007) A Three Cycle View of Design Science Research *Scandinavian Journal of Information Systems* 19(2)
- [38] Nowell, L., Norris, J., White, D., Moules, N. (2017). Thematic Analysis *International Journal of Qualitative Methods* 16(1), 1609406917733847. <https://dx.doi.org/10.1177/1609406917733847>
- [39] Linstone, H., Turoff, M., others, . (1975). The delphi method

- [40] Saunders, M., Lewis, P., Thornhill, A. (2007). Research methods Business Students
- [41] Okoli, C., Pawlowski, S. (2004). The Delphi method as a research tool: an example, design considerations and applications *Information & Management* 42(1), 15-29. <https://dx.doi.org/10.1016/j.im.2003.11.002>
- [42] Jaatun, M., Cruzes, D., Luna, J. (2017). DevOps for Better Software Security in the Cloud <https://dx.doi.org/10.1145/3098954.3103172>
- [43] Jaatun, M. (2018). Software Security Activities that Support Incident Management in Secure DevOps <https://dx.doi.org/10.1145/3230833.3233275>
- [44] Oyetoyan, T., Cruzes, D., Jaatun, M. (2016). An Empirical Study on the Relationship between Software Security Skills, Usage and Training Needs in Agile Settings 2016 11th International Conference on Availability, Reliability and Security (ARES) <https://dx.doi.org/10.1109/ares.2016.103>
- [44] Carter, K. (2017). Francois Raynaud on DevSecOps *IEEE SOFTWARE* 34(5), 93-96. <https://dx.doi.org/10.1109/ms.2017.3571578>
- [45] Torkura, K., Sukmana, M., Cheng, F., Meinel, C. (2017). Leveraging Cloud Native Design Patterns for Security-as-a-Service Applications 2017 IEEE International Conference on Smart Cloud (SmartCloud) <https://dx.doi.org/10.1109/smartcloud.2017.21>
- [46] Myrbakken, H., Colomo-Palacios, R. (2017). DevSecOps: A Multivocal Literature Review 770(), 17-29. <https://dx.doi.org/10.1007/978-3-319-67383-72>
- [47] Derksen, D., Neggens, D., Onwezen, D., Zelen, S./ (2018). Agile Secure Software Lifecycle Management Secure by Agile Design
- [48] Tuma, K., Calikli, G., Sc, ., ariato, R. (2018). Threat analysis of software systems: A systematic literature review *JOURNAL OF SYSTEMS AND SOFTWARE* 144(Acm Sigplan Notices 49 6 2014), 275-294. <https://dx.doi.org/10.1016/j.jss.2018.06.073>
- [49] Hernan, S., Lambert, S., Ostwald, T., Shostack, A. (2006). Threat modeling-uncover security design flaws using the stride approach *MSDN Magazine*
- [50] Shostack, A. (2014). Elevation of privilege: Drawing developers into threat modeling
- [51] Watson, C. (2012). OWASP Cornucopia Ecommerce Website Edition
- [52] Shostack, A. (2008). Experiences Threat Modeling at Microsoft
- [53] Rios, E., Iturbe, E., Mallouli, W., Rak, M. (2017). Dynamic security assurance in multi-cloud DevOps <https://dx.doi.org/10.1109/cns.2017.8228701>
- [54] Rios, E., Iturbe, E., Larrucea, X., ., Mallouli, W., Dominiak, J., Munteș, V., Matthews, P., Gonzalez, L. (2019). Service level agreement-based GDPR compliance and security assurance in (multi)Cloud-based systems *IET SOFTWARE* 13(3, SI), 213-222. <https://dx.doi.org/10.1049/iet-sen.2018.5293>
- [55] Siewruk, G., Mazurczyk, W., Karpinski, A. (2019). Security Assurance in DevOps Methodologies and Related Environments *International Journal of Electronics and Telecommunications* 65(2), 211-216.
- [56] Gruhn, V., Hannebauer, C., John, C. (2013). Security of public continuous integration services <https://dx.doi.org/10.1145/2491055.2491070>
- [57] Rimba, P., Zhu, L., Bass, L., Kuz, I., Reeves, S. (2015). Composing Patterns to Construct Secure Systems 2015 11th European Dependable Computing Conference (EDCC) <https://dx.doi.org/10.1109/edcc.2015.12>
- [58] Fraile, F., Flores, J., Anaya, V., Saiz, E., Poler, R. (2018). A Scaffolding Design Framework for Developing Secure Interoperability Components in Digital Manufacturing Platforms 2018 International Conference on Intelligent Systems (IS) <https://dx.doi.org/10.1109/is.2018.8710510>

- [59] Diaz, J., Perez, J., Lopez-Pena, M., Mena, G., Yague, A. (2019). Self-Service Cybersecurity Monitoring as Enabler for DevSecOps *IEEE Access* 7(), 100283-100295. <https://dx.doi.org/10.1109/access.2019.2930000>
- [60] Jaatun, M. (2018). Software Security Activities that Support Incident Management in Secure DevOps <https://dx.doi.org/10.1145/3230833.3233275>
- [61] Forsgren, N., Smith, D., Humble, J., Frazelle, J. (2019). 2019 Accelerate State of DevOps Report

---

## Credits

Image on cover and back page: ID 141352479 © Michal Balada | Dreamstime.com

Image on section one cover page: ID 168373105 © Ah Naeem | Dreamstime.com

Image on section two cover page: ID 157425409 © Vitaliy Pozdeev | Dreamstime.com

Image on section three cover page: ID 27213058 © catiamadio | Dreamstime.com

Image on section four cover page: ID 24701801 © Photomo | Dreamstime.com

Image on appendixes cover page: ID 94173390 © Andrea Simon | Dreamstime.com

