Trends for the DevOps Security. A Systematic Literature Review

Tiina Leppänen[1], Anne Honkaranta[2], Andrei Costin[3]

University of Jyväskylä, Finland
leppatii@gmail.com[1], anne.honkaranta@gmail.com2, ancostin@gmail.com[3]

Abstract. Due to technical advances, old ways for securing DevOps software development have become obsolete. Thus, researchers and practitioners need new insights into the security challenges and practices of DevOps development. This paper reviews the data extraction and analysis phase and results of a Systematic Literature Review (SLR) study that was carried out in 2019. The outcome is an updated list of security challenges and practices for DevOps software development. Both reviews shows that the most essential challenges for the DevOps security deal with the complexity of the development pipelines and the overall complexity of the cloud and microservice environments. The security activities identified were classified by using the BSIMM maturity model for software security as a framework. Our review shows that DevOps security research focuses mostly on deployment phase and technical aspects of software security. We compared the security activities identified in our study with the ones identified by the BSIMM development company in their 2020 review of 128 practitioners' security practices and found matching practices and similar trends.

Keywords: DevOps, Security, Systematic Literature Review.

1 Introduction

Over the past decade, the increasing "need for speed" for faster software release cycles has increased the popularity of Agile methodologies [1]. DevOps, *"a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality"* [2], is the cornerstone for agile and fast software development. One enabler for DevOps's fast deliveries is automated delivery pipelines [4, 5, 6]. In addition, DevOps blends the development and operations (e.g., maintenance of the software) together [3].

The software community has largely adopted DevOps; it was estimated that in 2020, even 74% of database professionals used it. In an article from the *Harvard Business Review* 80% of the respondents considered DevOps as essential for their organizations, and 69% reported using DevOps either selectively or solely for their software development [8]. DevOps also appeals to the builders of software because of its emphasis on the collaborative culture and positive impact on career, and for providing the tools to build higher quality software [9].

Security breaches are reported daily on the news [10], and several identified security incidents as well as the rate of cyber-attacks are rising, which underlines the need for

enhancing security of the contemporary software [4]. Traditional security approaches have focused on "gluing" software security as the final step in software development [11]. This approach turns the focus of the security on the firewalls, intrusion detection/prevention and antivirus software, instead of incorporating the security into the software itself [11]. Building secure software is not a “glue-on” activity but requires complex operations to ensure that software cannot be easily attacked [12] and does not contain vulnerabilities [3].

Software configurations have grown more complex than ever, and the contemporary approaches for securing software have become outdated [3]. Thus, developers and researchers lack information about 1) security challenges and 2) new features that their peers are utilizing in secure software development. This paper provides information about the topics by presenting updated results of a Systematic Literature Review (SLR) which was conducted in 2019 [13]. The challenges for DevOps security were identified using typification, whereas the activities for securing software were presented by using the Building Security in Maturity Model (BSIMM) by Synopsis Corporation as a framework [15, 21].

The current paper contributes by:

- reviewing the data extraction phase and findings of an original SLR study,
- discussing the findings, i.e., challenges and practices for DevOps security, and
- updating the security activities identified in primary studies to the latest version of the BSIMM software security maturity model.

The current paper also compares updated DevOps security activities from the research literature with the findings of BSIMM project study in 2020, which covered 128 organizations and their security postures in practice.

Section two introduces the SLR and snowballing methods, and BSIMM security maturity model, which is used as a framework for presenting the security activities identified from the preliminary studies. Section three describes the original SLR and the review process. The challenges for DevOps security and the security activities identified in the SLR are presented in section four. Section four also provides a comparison between the BSIMM project [15] Top 10 security activities and security trends identified by the BSIMM project [48] and the Top 14 list of security activities from our SRL study. Section five discusses the findings and proposes avenues for further research.

For the remainder of the current paper, the original SLR by [13] is referred to as “the original SLR study,” and our review of the study is referred as “the review of the original SLR study.”

2 Systematic Literature Review (SLR), Snowballing, and the BSIMM Framework

2.1 Systematic Literature Review (SLR)

An SLR provides a way to review the results provided by the research in a rigorous way [14]. An SLR is not focused on gathering relevant findings related to a research

question; it also seeks to provide evidence-based guidelines for practitioners. Systematic reviews are laborious by their very nature, but they may provide accumulative information about the phenomena under scrutiny.

According to Kitchenham's original guidelines [14,16], the objective of an SLR is to identify all relevant research regarding the research questions. Unlike other literature review methods, an SLR may be utilized throughout the entire research process, including during formulating the research questions, searching for the relevant research, extracting data from the previous research, analyzing the findings, and providing guidelines for practitioners [16]. SLRs have been adopted in other scientific domains, such as criminology, social policy, economics, nursing, and software engineering [16].

An SLR uses specific concepts to separate the research papers that are analyzed and the literature reviews, such as the SLR itself. The research papers that are scrutinized in the SLR are called the *primary studies*, while the SLR itself is called *the secondary study* [17].

MacDonell et al. [18] evaluated the reliability of SLRs by comparing the results of two studies carried out by two independent groups of researchers. The SLR was proven to be a robust and trustworthy method for literature reviews. According to [17] an SRL study starts with identification of what needs to be known and summing it up into the form of the research questions. After that, a method for searching for the primary studies is formed. The search method should contain clear inclusion and exclusion rules. For example, it may be stated that certain digital libraries are included, while some others are excluded. After the search has been carried out the initial result set of primary studies is narrowed down by carefully examining the names, abstracts, and contents of the primary studies [17].

Once the result set is selected, an analysis tool should be prepared for data extraction [17]. It is important that the data analysis and extraction is tested by multiple researchers to ensure that the findings are valid and repeatable, and that the data extraction framework is properly understood and used [17]. The results of the analysis should be revisited and compared with other similar studies to ensure validity. Ways to enhance the analysis should also be carefully considered during the analysis.[17]

The SLR methodology is developed by medical practitioners thus it has an unordinary step in analysis: the researchers should develop and share practical guidelines from the study with other practitioners [17]. The outcome of the study is the study report or a research paper.

The SLR methodology also emphasizes that the research report should include the method and search strategy that was used, how the inclusion and exclusion rules were utilized, and how the fellow researchers extracted the data and other potentially relevant aspects of the study. [17]

It may be difficult to identify all relevant research by manual or automated database searches [19]. Many expert researchers know that a key for finding the most relevant research is to follow the references used by other researchers. The more particular research item is referenced by others, the more valuable it may be for the research. *The snowballing* method proposes that the list of references within each primary study should be further explored, and potential references listed from it should be examined. This is called backward snowballing [19, 20]. Forward snowballing aims to identify

those papers that cite papers in the results. The cited papers may be identified, for example, by using Google Scholar. The set of papers citing the original reference paper is subjected to a second round of backward and forward snowballing [19]. The snowballing goes on for as long as a strong paper is present [19, 20].

2.2 The BSIMM Framework

Measuring software security only by "look and feel" is a challenging task. Instead of trying to compare software that solves the same problem, models for identifying and measuring activities supporting security have been developed [12]. The Building Security in Maturity Model (BSIMM) [15, 21] by Synopsis Corporation is a framework that can be used as a tool for measuring the security of software, to compare as security plan with other organizations' security initiatives, and for building a roadmap for enhancing security measures. BSIMM also provides a vocabulary for describing security activities. [21]

BSIMM contains four high-level domains that, in turn, contain three practices each. These top levels of the framework are illustrated in Table 1 [15]. Each practice contains 7–12 observed and related activities, comprising of 122 security activities in total. All the activities can be used in an organization rarely or intensively [15]. Frequently observed activities are designated as level 1, less frequently observed activities are designated as level 2, and infrequently observed activities are designated as level 3. [15, 21]

Table 1. The domains and practices of the BSIMM model [15].

Domain 1: Governance	**Domain 2: Intelligence**	**Domain 3: SSDL Touchpoints**	**Domain 4: Deployment**
Strategy & Metrics	Attack Models	Architecture Analysis	Penetration Testing
Compliance & Policy	Security Features & Design	Code Review	Software Environment
Training	Standards & Requirements	Security Testing	Configuration Management/ Vulnerability Management

The first version of BSIMM was created in 2008. The original SLR mapping was done using BSIMM version 9, which was released in 2018. The BSIMM framework has evolved since then from version 9 to version 12. The main changes include the addition of DevOps in version 10 [22] (i.e., DevOps was not a part of the BSIMM framework used in the original SLR study), "shift left" (an emphasis on the application security at the earliest stages of the software development) transforming to "shift everywhere" in version 11 [23], and additional activities regarding vulnerabilities and malicious code with automated security tools in version 12 [15].

The total amount of security activities in the BSIMM framework has grown from 116 in version 9 to 122 in version 12. In addition to the increase in quantity, activities have been modified to reflect the advances in technology [15].

In addition to the BSIMM, several other frameworks provide a common measuring stick for security. Two were developed by the OWASP Foundation (Open Web Application Security Project®), a non-profit foundation that works to improve the security of software and is familiar to most application security professionals.

The "OWASP Software Maturity Model" [24], which has been referred to as SAMM or OpenSAMM, is an open framework for organizations to analyze and improve their software security posture. Although the BSIMM model is descriptive by its very nature, the SAMM model measures maturity against a prescriptive set of security practices. Because the BSIMM can be used to understand how organizations can introduce security into their processes, SAMM supports the understanding of how security level can be improved in organizations' products. [25] The DSOMM (the OWASP Devsecops Maturity Model) [26] aims to tackle security in agile and DevOps software development.

The BSIMM was chosen as the framework for presenting the research results related to the security activities by [13, pp 23] because "it has been developed using the largest set of data collected about software security anywhere." [12] also utilized the BSIMM on his security assessments covering 20 public and six private sector organizations.

3 The Original Literature Review and the Review Process

[13] had a broader scope in her research than the one selected for the current paper. The original research questions were defined as follows [13]:

- RQ1: What are the challenges of security in DevOps as reported by the authors of primary studies?
- RQ2: Which security activities are associated with DevOps in the literature?
- RQ3: How are the CAMS (culture, automation, measurement and sharing) principles reflected in secure DevOps research?

Our research was limited by time and scope; thus, the third original research question was not reviewed in the review.

3.1 The Search Process and Selected Primary Studies

The original SLR study proceeded using the SLR guidelines given by [14] and [17]. To find relevant primary studies for analysis, [13] created a search strategy based on search terms that were then used as a search string. She decided to include research articles and conference papers in the result set and exclude books and writings of opinion. The search terms used by the researcher are presented in Table 2.

Table 2. The keywords and phrases used in the search for primary studies [13, pp. 28].

Topic	Search terms derived from topics
Main research topic	devops & secur*
Variations	devsecops, secdevops, devopssec

The search for primary sources was conducted in April 2019 in four digital libraries: Science Direct, ACM Digital Library, IEEE Xplore, and Springer Link. [13] justified this selection by pointing out that these digital libraries were utilized for other literature review studies of software development and design, for example, [27, 28]. [13] reported that she inserted the search terms in title and/or in abstract data fields. The publication year of the search results was not limited. The result of the first search round was 292 articles. [13]

[13] narrowed the resulting set of primary studies down twice. In the first round, the articles' titles and abstracts were scrutinized regarding the research questions. The resulting primary studies were narrowed down to 38 primary studies. In the second round, all the articles in the result set were read, and the inclusion and exclusion criteria mentioned above was used to select the remaining 16 articles. Backward and forward snowballing processes following the guidelines provided by [19] produced two new primary studies. In the review process, this phase was not repeated because the original result set of primary studies was not available.

The final set for review in the original SLR study consisted of the 18 primary studies. They are referenced in this review as [3, 12, 29-44]. As our review of the SLR was limited by time and scope, the review started by adopting the result set of the 18 primary studies from the original SLR study.

3.2 Data Extraction from the Primary Studies

The original SLR study was conducted in a following way. First, [13] analyzed the primary studies and extracted the challenges by using typification. Typification is a method in which specific keywords and phrases are searched from the text. There were 27 challenges identified in the first round of analysis, and after a review of the list of the challenges and wordings in the research papers, the researcher typified the challenges into nine separate themes. [13]

The second research question was studied using the BSIMM framework. Even though the BSIMM is a maturity model, it was used as a framework for plotting security activities identified from the primary studies to the BSIMM security activities. [13] crafted a spreadsheet containing the BSIMM domains, related security practices, and activities.

After all primary studies were read and analyzed, there were total of 139 distinct security activities recorded, and these were sorted into 47 groups. These groups, in turn, were plotted against the security practices and activities of the BSIMM framework. If the counterpart for an activity was found in the BSIMM framework, the number of the study was added to the BSIMM practice's activity field, along with possible notes from [13]. The result was a table containing the BSIMM domains and practices; each practice field contained a list of the BSIMM security activities identified from the studies.

The review of the original SLR study was carried out as follows. First, the researchers prepared an Excel spreadsheet for data extraction containing a list of the original primary studies ordered as in the original SLR study. In addition, two columns were created: one for the most essential findings extracted by the researchers and another in

which the security challenges and activities identified in the original SLR study were copied. Finally, two additional columns provided space for the review comments and change proposals related to the original challenges and identified security activities.

Two primary studies were randomly selected for review. A trial data extraction comparing the results with the original SLR study was carried out by two authors of this paper. Some remarks for data interpretation and documenting were formulated for enhancing the process. The researchers split the papers in the result set into half, and each performed an individual review of the papers.

First, the findings related to the security challenges for DevOps were reviewed and analyzed together. Three new challenges were identified, as described in the following chapter. Second, the findings related to the security activities were scrutinized. It was soon noticed that something did not add up with the findings, and it was discovered that the BSIMM framework had been changed from to the one that has been used in the original study; hence the findings regarding BSIMM version 9, and the current version (12) were studied for gathering the findings. Data extraction was laborious, and the use of the BSIMM framework interpretation was quite challenging in some cases. The findings were analyzed together, and the differences were documented on a spreadsheet. The differences in our analysis are presented and discussed in the following chapter, which shows the findings of the original SLR study, and the findings of the review as plotted to the newest version of the BSIMM framework.

4 The Findings

This chapter discusses our findings about the similarities and differences between the original and new findings, along with the possible causes for these differing interpretations.

4.1 Challenges for DevOps

Figure 1 summarizes the challenges identified in the original SLR study and the current review. New challenges are marked with *.

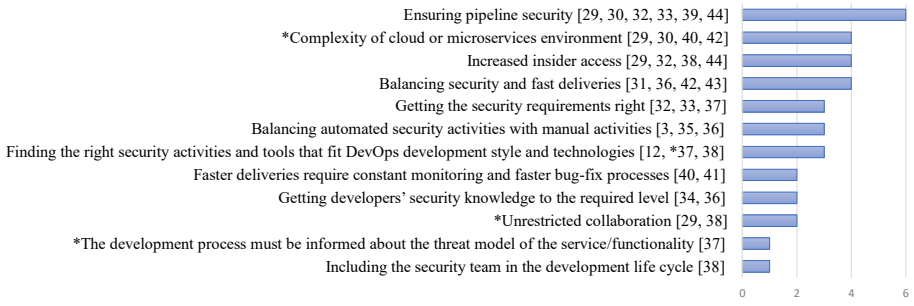

Figure 1. Security challenges in DevOps after the review of the original SLR study.

Many of the primary studies were carried out in cloud environments, which are complex and fast evolving by their nature. Fast development stresses the need for balancing between quality, security, and speed. Many papers noted that new technologies and tools, such as microservices and containers, are not designed primarily for security in mind, and their use in the pipelines requires that novel kinds of security risks related to them are identified appropriately. Therefore, *Ensuring pipeline security* was the challenge that was brought up in six articles. The *Complexity of cloud or microservices* environment is related to the complexity of the contemporary cloud services, too, and it is apparent in four of the primary studies. Although these challenges are remarkable, they are not specific to the DevOps method.

Balancing security and fast deliveries, on the other hand, is quite tightly tied into the DevOps method. Fast deliveries are the core and foremost reason for the creation of the DevOps method. Automation is required for quickening security procedures, but the outcome of this automation may not be as good as anticipated [38]. One challenge with fast deliveries is also how to get the security team to do their work at the right time and promptly.

Primary study [29] focused on an insider threat, which is obviously a security concern, while the team accessing the development environment is larger with the appearance of the operations team. Furthermore, the DevOps culture embraces the idea that developers and operations personnel both become multi-talented and may work in both roles, creating an increased number of insiders in the environment who also have larger access to the components in the environment. The insider threat was bought up by many other papers, too (e.g., primary studies [3] and [38]).

Insider threat caused by the co-existence of both “Devs” and “Ops” personnel having broad access to the software development environments and the increased speed for development are perhaps the most prominent features of DevOps development leading to characterizing the security practices in preliminary studies [12] and [38] as “Finding the right security activities and tools that fit the DevOps development style and technologies”.

4.2 Security Practices and Activities for DevOps

The security activities identified in the original SLR study and updated and mapped to BSIMM framework version 12 [15] during the research review by the current authors are presented in the tables below. The total amount of security practices (139) and ranking between the BSIMM domains did not change as a result of this review. The numbers in brackets after the BSIMM practice column name show the total number of primary studies mentioning the security practice. The numbers in brackets after the BSIMM activity names refer to the identifier of the primary study (listed in Chapter 3.1). An article with no equivalency to the original SLR is marked with strikethrough (e.g., ~~40~~). A new finding is emphasized (e.g., **3**), and an article not openly available is underlined (e.g., <u>35</u>).

The *Governance* domain presented in Table 3 includes activities that belong to the organization, management, and measurement practices of a software security initiative. Because primary study [3] emphasized the meaning of education and training as being

especially important in secure agile development during planning and getting input from the learning phase, that article was added in *Conduct software security awareness training activity* (T1.1).

Table 3. Security activities used in the BSIMM *Governance* domain.

BSIMM practice (23) > (24)	**BSIMM observed and related activities**
Strategy and Metrics (8) *No change*	[SM1.4] Implement lifecycle instrumentation and use to define governance. [34, 36, 37, 43] [SM2.2] Verify release conditions with measurements and track exceptions. [37] [SM2.3] Create or grow a satellite. [3, 12] [SM2.6] Require a security sign-off prior to software release. [37]
Compliance & Policy (13) *No change*	[CP1.1] Unify regulatory pressures [37, 38] [CP1.3] Create policy. [31, 32, 38, 42, 43, 44] [CP2.1] Build PII data inventory. [12] [CP2.3] Implement and track controls for compliance. [37, 38] [CP2.4] Include software security SLA in all vendor contracts. [40] [CP3.2] Impose policy on vendors. [40]
Training (2) > (3)	[T1.1] Conduct software security awareness training. [**3**, 38] [T3.5] Establish SSG office hours. [12]

The *Intelligence* domain contains software security practices and activities whose results end up in collections of corporate knowledge. This domain received the least mentions of all the domains. The results are presented in Table 4. Two articles were removed because they did not cover the attack patterns nor open-source related activities directly. The changes in BSIMM activity names and their purpose were also reanalyzed.

Table 4. Security activities used in the BSIMM *Intelligence* domain.

BSIMM practice (21) > (18)	**BSIMM observed and related activities**
Attack Models (10) > (9)	[AM1.2] Create a data classification scheme and inventory. [29, 41] [AM1.3] Identify potential attackers. [29, 36] [AM1.5] Gather and use attack intelligence. [35, 36] [AM2.1] Build attack patterns and abuse cases tied to potential attackers. [29, ~~41~~] [AM2.2] Create technology specific attack patterns. [30] [AM2.7] Build an internal forum to discuss attacks. [12, ~~29~~]
Security Features & Design (2) *No change*	[SFD1.1] Integrate and deliver security features. [42] [SFD2.1] Leverage secure-by-design components and services. [42]
Standards & Requirements (9) > (7)	[SR1.3] Translate compliance constraints to requirements. [32, 38, 41] [SR2.4] Identify open source. [35, 36, ~~38~~] [SR3.1] Control open-source risk. [12, 35, ~~38~~]

The *Secure Software Development Lifecycle (SSDL) touchpoints* domain covers practices included in all software security methodologies. In our review of the original

SLR study, the total number of articles in the *Intelligence* domain increased by three (see Table 5). Primary study [34] proposed several security tools relating to the use of architecture analysis processes and automated deployment that was also covered in the research by primary study [3]. On the other hand, primary study [43] mentioned difficulties with automated testing and presented CAVAS workflow with automated steps but did not explicitly cover the usage of automated tools. This article was excluded from the results.

Table 5. Security activities used in the BSIMM *SSDL touchpoints* domain.

BSIMM practice (36) > (39)	**BSIMM observed and related activities**
Architecture Analysis (13) > (14)	[AA1.1] Perform security feature review. [29, 30, 33, 36, 37, 38, 40, 41, 43] [AA2.1] Define and use AA processes. [**34**, 37, 38, 41] [AA3.3] Make the SSG available as an AA resources or mentor. [37]
Code Review (12) > (14)	[CR1.4] Use automated tools along with manual review. [**3**, 30, 36, 37, 38] [CR1.5] Make code review mandatory for all projects. [36, 37] [CR1.6] Use centralized reporting to close the knowledge loop. [36] [CR2.6] Use automated tools with tailored rules. [**3**, 31, **34**, 35, 36, ~~43~~] [CR2.7] Use a top N bugs list (real data preferred). [30]
Security Testing (11) *No change*	[ST1.3] Drive tests with security requirements and security features. [30, 31, 33, 34, 38, 43] [ST2.4] Share security results with QA. [36] [ST2.5] Include security tests in QA automation. [34, 36] [ST3.3] Drive tests with risk analysis results. [41] [ST3.5] Begin to build and apply adversarial security tests (abuse cases). [34]

The domain with the most mentions, *Deployment*, includes practices that relate to traditional network security and software maintenance. The most often mentioned practice and activity are also in this domain (see Table 6). *Use application behavior monitoring and diagnostics* (SE3.3) was mentioned in 10 of the primary studies. This result is well aligned with DevOps' principle of measuring what is needed to keep the heart of DevOps beating. The value of monitoring and using data from deployed applications is crucial in automated pipelines. Article [44] was added in the *Penetration testing* practice. The research in this article focused on continuous delivery, and penetration testing activity was recommended for assessing the security of web applications. In the *Software Environment* practice, two articles were removed because they did not cover activities of concern.

Table 6. Security activities used in the BSIMM *Deployment* domain.

BSIMM practice (59) > (58)	**BSIMM observed and related activities**
Penetration Testing (4) > (5)	[PT1.2] Feed results to defect management and mitigation system. [31] [PT1.3] Use penetration testing tools internally. [31, 36, 37, **44**]

BSIMM practice (59) > (58)	**BSIMM observed and related activities**
Software Environment (43) > (41)	[SE1.1] Use application input monitoring. [12, ~~38~~, 40, 42] [SE1.2] Ensure host and network security basics are in place. [29, 30, 31, 32, 33, 39, 40, 42, 43] [SE2.5] Use application containers to support security goals. [29, 32, 39, ~~42~~, 43] [SE2.6] Ensure cloud security basics. [3, 40, 41, 42, 43, 44] [SE2.7] Use orchestration for containers and virtualized environments. [32, 34, 39, 40, 42, 43, 44] [SE3.3] Use application behavior monitoring and diagnostics. [31, 33, 34, 12, 3, 38, 40, 41, 42, 43] [SE3.6] Enhance application inventory with operations bill of materials. [30, 35]
Configuration Management & Vulnerability Management (12) *No change*	[CMVM1.1] Create or interface with incident response. [12] [CMVM1.2] Identify software defects found in operations monitoring and feed them back to development. [3, 12, 36, 40] [CMVM2.1] Have emergency codebase response. [12] [CMVM2.2] Track software bugs found in operations through the fix process. [3, 36] [CMVM2.3] Develop an operations inventory of applications. [12, 35] [CMVM3.3] Simulate software crises. [12, 33]

The BSIMM security activities that received more than four mentions were introduced in more detailed level in the original SLR study [13] on a list of the top 13 activities. In this review, that list has been updated to contain 14 items (see Figure *2*). There were no changes in the top seven security activities. *Use automated tools along with manual review* (CR1.4) and *Use automated tools with tailored rules* (CR2.6) got both one new mention improving their rankings compared with the original top 13 list. In turn, *Use application containers* (SE2.5) and *Use application input monitoring* (SE1.1) were activities that lost one mention each.

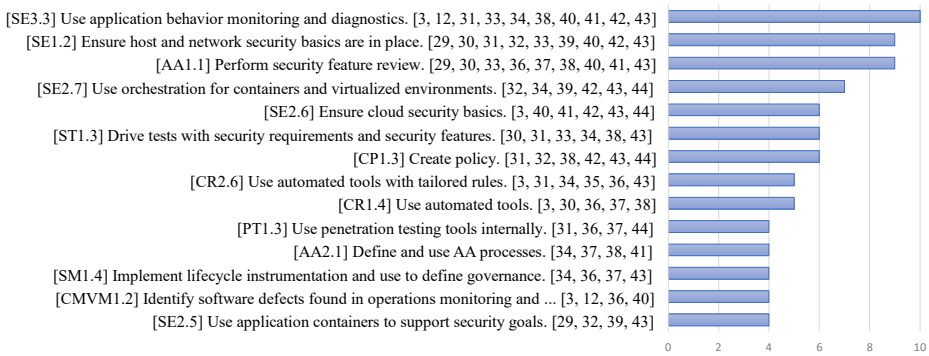

Figure 2. The top 14 BSIMM security activities after the review of the original SLR.

The results of the current review align with the original SLR study [13] that contemporary research has focused strongly on technology and how it is secured in the DevOps infrastructures. Some of the selected research articles were purely concentrated on

certain cloud computing technologies (e.g., containers, multicloud environments). As in the original SLR study, the results of the current review showed that 42% of the security activities applied to the primary studies were related to the deployment domain.

Adopting DevOps has been found to be a challenging task because of the vast amount of information, practices, and tools related to it [46]. DevOps heavily emphasizes the relevance of building a collaborative, sharing, and transparent culture. Regardless of these principles, *Training and Security* and *Design* practices received the fewest mentions.

4.3 Findings of the SLR Review Versus the Findings of the BSIMM Top 10 Security Activities

Synopsys Software, the company that created the BSIMM framework, assessed the security of hundreds of companies by using the BSIMM framework. In its latest research project, 128 organizations were scrutinized using the BSIMM framework [15]. The BSIMM project published an overview containing statistics and findings for the whole project [15].

The security activities top 14 list resulting from our SLR review has four matches with the BSIMM Top 10 list identified from the 128 participating organizations. The matching security activities are presented in Table 7.

Table 7. BSIMM12 top 10 activities compared with the findings in the current review.

#	**BSIMM12 Top 10**	**This review**
1.	[SM1.4] Implement lifecycle instrumentation and use to define governance	11th
2.	[SE1.2] Ensure host and network security basics are in place	2nd *(Exact match)*
3.	[CP1.2] Identify PII obligations	-
4.	[AA1.1] Perform security feature review	3rd *(Good match)*
5.	[PT1.1] Use external penetration testers to find problems	-
6.	[CMVM1.1] Create or interface with incident response	-
7.	[SFD1.1] Integrate and deliver security features	-
8.	[CR1.4] Use automated tools	8th *(Exact match)*
9.	[ST1.1] Ensure QA performs edge/boundary value condition testing	-
10.	[SR1.3] Translate compliance constraints to requirements	-

Implement lifecycle instrumentation and use to define governance (SM1.4) which was ranked as 11th in the current SLR review is in the first place in the BSIMM Top 10 list of practitioner's security activities. *Ensure host and network security basics are in place* (SE1.2) was an exact match in second place in both lists. *Perform security feature review* (AA1.1) was ranked in fourth place in the BSIMM Top 10 list, and in third in the current SLR review. This confirms that the most appreciated BSIMM security activities by the practitioner organizations were also considered important by the

researchers. For example, activity *Use automated tools* (CR1.4) was an exact match in these rankings.

The first two security activities in the BSIMM12 top 10 list, *Implement lifecycle instrumentation and use to define governance* (SM1.4) and *Ensure host and network security basics are in place* (SE1.2) reside in the *Governance* domain. This indicates the importance of practices that help organize, manage, and measure secure software processes in a proactive way and throughout the lifecycle of the software. *Identify PII obligations* (CP1.2) was not covered in any of the primary studies of our SLR review, even though the implementation of the GDPR received a lot of attention and forced software providers to develop novel protective measures for personal information processing.

Both rankings emphasize that the most utilized security activities focus on the technology domains of the BSIMM framework. To be more exact, the top 10 of the most mentioned BSIMM activities in our SLR review can be divided into two categories: *Deployment* and *SSDL Touchpoints* domains.

The company that published the BSIMM12 Top10 report has also published an overview of DevOps trends that were identified as emerging in the BSIMM12 project. The review reminds that too few security tools allow security gaps, but too many are a burden for the developers, and may freeze the pace of software development which remains as the ultimate target for DevOps. [48]

The new BSIMM review [48] identifies some interesting, totally new trends for security as:

- ransomware and supply chain disruptions, which call upon for increased scrutiny of software security
- increasing the capabilities for cloud security
- security teams are increasingly collaborating with DevOps practitioners, lending staff and knowledge to them instead of mandating security postures, and
- security testing as an automated activity seems to have doubled its size.

The new BSIMM review [48] points out activities that have gained a remarkable growth in the past 24 months when compared with our SRL review top 14 list: 1) *Use orchestration for containers and virtualized environments* (SE2.7) – over 500% increase in mentions in the security activities, ranked as fourth in our SLR review top 14 list, 2) *Ensure cloud security basics* (SE2.6) – over 500% increase in mentions in the security activities, which was fifth in our SRL review top 14 list, and 3) *Use application containers to support security goals* (SE2.5) – over 200% increase in mentions in the security activities, which was 14th in our SRL review top 14 list. In overall it seems that research on DevOps security is more focused on *Software development environment* practice, while the practitioners' security activities cover more of the BSIMM framework's different practices.

5 Discussion and Avenues for Further Research

The current study reviewed the method and findings of an original SLR study conducted in 2019 which discovered the challenges and the cures for DevOps security. Since the original study was carried out, the BSIMM framework has changed mainly in the areas of DevOps and automated security tools. New activities were added to reflect the impact of DevOps security, and existing ones were updated to reflect how organizations are implementing them.

The contribution of the current review was the cross-validation and assessment of the original SLR study by reviewing the original 18 primary research studies identified by Koskinen [13], and as a result updating the challenges for DevOps, and mapping the security activities identified to the newest version of the BSIMM software security maturity model as a framework. We also mapped our findings with the BSIMM project's results from a study covering 128 organizations.

Mapping the security activities identified from the primary studies to the BSIMM framework was challenging. Our findings were well aligned with the original SLR study. As the main finding and common trend of the SLR and the BSIMM analysis of the organizations was the technology focus, the key differences were the diversity of domains and practices. The results of SLR indicate strong focus on software environment practices, while the BSIMM analysis of the organizations covers 10 different practices in the list of top 10 activities. The current study showed that research on DevOps security is mostly challenged by complexity in the management of the pipelines and overall security of the complexity of cloud and microservice environments. The BSIMM framework security activities *Use application behavior monitoring and diagnostics* and *Ensure host and network security basics are in place* were the most mentioned security activities on the primary studies. The BSIMM top 10 security activities list from 2020 show that *Implement lifecycle instrumentation and use to define governance* was the most popular security activity while *Ensure host and network security basics are in place* came in second place. Interestingly, our SLR study's top 14 and the BSIMM project top 10 security activities had three matches; in addition to *Ensure host and network security basics are in place*, *Implement lifecycle instrumentation and use to define governance* and *Use automated tools* were mentioned in both listings. The BSIMM review [48] had identified ransomware and collaboration with security practitioners and DevOps specialists as activities that are gaining attention. It still seems that both the practitioners and researchers were mostly focused on the technical aspects of software security, the researchers perhaps even more than the practitioners.

We identified several potential avenues for further research. [47] carried out an SLR for detecting the challenges for security in the DevOps work. They applied the Preference Ranking Organization Method for Enrichment Evaluation (PROMETHEE) technique to prioritize the 18 challenges identified in the study. A framework like PROMETHEE would help to organize the challenges identified by our study in a more ordered way. It would also be interesting to compare the challenges of our SLR study with the challenges identified by [47], which would be possible if both listings of challenges were presented with the same framework.

The original SLR study was conducted in 2019. We conducted a new search on the research databases that were used to identify the preliminary studies for this SLR, finding 27 potential new studies that could be analyzed to update the findings.

The extraction of data from the primary sources was conducted by two researchers in our review study. Yet there is a possibility to false interpretations, as always when humans are analyzing abstract evidence such as research papers. Our SLR was also limited by time, thus we did not have the chance to update and enlarge the primary studies with the new 27 studies that were identified.

We hope that practitioners and researchers interested in SLR, DevOps and security find the current paper useful for their endeavors.

Acknowledgements

Authors wish to thank the two anonymous reviewers for their invaluable comments and feedback that helped greatly to improve the quality of paper. Specifically, the other reviewer offered a very thorough review with many good comments, ideas, and suggestions.

References

1. Lwakatare, L., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., Kuvaja, P., Mikkonen, T., Oivo, M., Lassenius, C.: Devops in practice: A multiple case study of five companies. Information and Software Technology, vol. 114, pp. 217–230 (2019).
2. Bass, L., Weber, I., Zhu, L.: DevOps: A software architect's perspective. Addison-Wesley Professional, (2015).
3. Jaatun, M., Cruzes, D., Luna, J.: DevOps for better software security in the cloud. Invited paper. In: ARES '17. Proceedings Of The 12th International Conference On Availability, Reliability And Security 2017, No 69, pp. 1–6. ACM (2017).
4. Hsu, T.: Hands-On Security in DevOps: Ensure continuous security, deployment, and delivery with DevSecOps. Packt Publishing Ltd, (2018).
5. Humble, J., Farley, D.; Continuous delivery: reliable software releases through build, test, and deployment automation. Pearson Education, (2010).
6. Konersmann, M., Fitzgerald, B., Goedicke, M., Holmström Olsson, H., Bosch, J., Krusche, S.: Rapid continuous software engineering-state of the practice and open research questions: Report on the 6th international workshop on Rapid Continuous Software Engineering (RCoSE 2020). ACM SIGSOFT Software Engineering Notes, 46(1), pp. 25-27 (2021).
7. RedGate Software The state of database devops 2021 report. https://www.red-gate.com/solutions/database-devops/report-2021
8. Vizard, M.: Survey finds wide gap between DevOps adoption and success. https://devops.com/survey-finds-wide-gap-between-devops-adoption-and-success/
9. Atlassian survey 2020 - DevOps trends. https://www.atlassian.com/whitepapers/devops-survey-2020
10. Williams, L., McGraw, G., Migues, S.: (2018). Engineering security vulnerability prevention, detection, and response. IEEE Software, 35(5), pp. 76-80, (2018).

11. Mohammed, N., Niazi, M., Alshayeb, M., Mahmood, S.: Exploring software security approaches in software development lifecycle: A systematic mapping study. Computer Standards & Interfaces, 50, pp. 107-115 (2017).
12. Jaatun, M.: Software security activities that support incident management in secure DevOps. In: Proceedings Of The 13th International Conference On Availability, Reliability And Security, pp. 1-6. ACM (2018).
13. Koskinen, A.: Devsecops: Building Security Into The Core Of Devops. University of Jyväskylä, Jyväskylä, Finland (2019).
14. Kitchenham, B: Procedures for performing systematic reviews. Keele University, UK, 33, pp. 1-26 (2004).
15. SynopsysSoftware: BSIMM12, 2021 Insights Trends Report. https://www.bsimm.com/
16. Kitchenham, B., Brereton, O., Budgen, D., Turner, M., Bailey, J., Linkman, S.: Systematic literature reviews in software engineering–a systematic literature review. Information and software technology, 51(1), pp. 7–15 (2009).
17. Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., Khalil, M.: Lessons from applying the systematic literature review process within the software engineering domain. Journal of systems and software, 80(4), pp. 571-583 (2007).
18. MacDonell, S., Shepperd, M., Kitchenham, B., Mendes, E.: How reliable are systematic reviews in empirical software engineering?. IEEE Transactions on Software Engineering, 36(5), pp. 676-687 (2010).
19. Wohlin, C.: Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: EASE '14. Proceedings Of The 18th International Conference On Evaluation And Assessment In Software Engineering, pp. 1-10. ACM (2014).
20. Jalali, S., Wohlin, C.: Systematic literature studies: Database searches vs. backward snowballing. In: Proceedings Of The 2012 Acm-Ieee International Symposium On Empirical Software Engineering And Measurement, pp. 29-38. IEEE (2012).
21. Glas, B.; Comparing BSIMM SAMM. https://owaspsamm.org/blog/2020/10/29/comparing-bsimm-and-samm/
22. Mello, J.: BSIMM10. DevOps is changing how software teams approach security. https://techbeacon.com/security/bsimm-10-devops-changing-how-software-teams-approach-security
23. SynopsysSoftware. https://news.synopsys.com/2020-09-15-Synopsys-Publishes-BSIMM11-Study-Highlighting-Fundamental-Shifts-in-Software-Security-Initiatives-in-Response-to-DevOps-and-Digital-Transformation
24. OWASP Foundation: OWASP software assurance maturity model. https://owasp.org/www-project-samm/
25. Pagel, T.: Overview of (DevSecOps) OWASP projects. https://owasp.org/www-chapter-germany/stammtische/frankfurt/assets/slides/48 OWASP Frankfurt Stammtisch 1.pdf
26. OWASP DevSecOps maturity model. https://owasp.org/www-project-devsecops-maturity-model/
27. Felderer, M., Fourneret, E.: A systematic classification of security regression testing approaches. International Journal on Software Tools for Technology Transfer, 17(3), pp. 305-319 (2015).
28. Souza, E., Moreira, A., Goulão, M.: Deriving architectural models from requirements specifications: A systematic mapping study. Information and Software Technology, 109, pp. 26-39 (2019).
29. Ahmadvand, M., Pretschner, A., Ball, K., Eyring, D.: Integrity protection against insiders in microservice-based infrastructures: From threats to a security framework. In: Federation Of

International Conferences On Software Technologies: Applications And Foundations, pp. 573–588. Springer, Heidelberg (2018).
30. Bass, L., Holz, R., Rimba, P., Tran, A., Zhu, L.: Securing a deployment pipeline. In: 2015 IEEE/ACM 3rd International Workshop on Release Engineering, pp. 4–7. IEEE (2015).
31. Beigi-Mohammadi, N., Litoiu, M., Emami-Taba, M., Tahvildari, L., Fokaefs, M., Merlo, E., & Onut, I.: A DevOps framework for quality-driven self-protection in Web software systems. In: Proceedings Of The 28th Annual International Conference On Computer Science And Software Engineering, pp. 270-274. IBM Corp. (2018).
32. Diekmann, C., Naab, J., Korsten, A., Carle, G.: Agile network access control in the container age. IEEE Transactions on Network and Service Management, 16(1), pp. 41-55 (2018).
33. Düllmann, T., Paule, C., van Hoorn, A.: Exploiting DevOps practices for dependable and secure continuous delivery pipelines. In: 2018 IEEE/ACM 4th International Workshop On Rapid Continuous Software Engineering (RCOSE), pp. 27–30. IEEE (2018).
34. Tigli, J. Y., Winter, T., Muntés-Mulero, V., Metzger, A., Velasco, E., Aguirre, A.: ENACT: Development, Operation, and Quality Assurance of Trustworthy Smart IoT Systems. In: Software Engineering Aspects Of Continuous Development And New Paradigms Of Software Production And Deployment: First International Workshop, DEVOPS 2018, March 5-6, 2018, Revised Selected Papers, vol. 11350, p. 112. Springer, Heidelberg (2019).
35. Mackey, T.: Building Open Source security into agile application builds. Network Security, 2018(4), pp. 5-8 (2018).
36. Mansfield-Devine, S.: DevOps: finding room for security. Network Security, 2018(7), pp. 15-20 (2018).
37. Michener, J., Clager, A.: Mitigating an oxymoron: Compliance in a DevOps environments. In: 2016 IEEE 40th Annual Computer Software And Applications Conference (COMPSAC), vol. 1, pp. 396–398. IEEE (2016).
38. Rahman, A., Williams, L.: Software security in DevOps: Synthesizing practitioners' perceptions and practices. In: 2016 IEEE/ACM International Workshop On Continuous Software Evolution And Delivery (CSED), pp. 70–76. IEEE (2016).
39. Raj, A., Kumar, A., Pai, S., Gopal, A.: Enhancing security of docker using Linux hardening techniques. In: 2016 2nd International Conference On Applied And Theoretical Computing And Communication Technology (iCATccT), pp. 94–99. IEEE (2016).
40. Rios, E., Iturbe, E., Mallouli, W., Rak, M.: Dynamic security assurance in multi-cloud DevOps. In: 2017 IEEE Conference On Communications And Network Security (CNS), pp. 467–475. IEEE (2017).
41. Schoenen, S., Mann, Z., Metzger, A.: Using risk patterns to identify violations of data protection policies in cloud systems. In: International Conference On Service-Oriented Computing, pp. 296-307. Springer, Heidelberg (2017).
42. Thanh, T., Covaci,S., Magedanz. T., Gouvas, P., Zafeiropoulos, A.: Embedding security and privacy into the development and operation of cloud applications and services. In: 2016 17th International Telecommunications Network Strategy And Planning Symposium (NETWORKS), pp. 31–36. IEEE (2016).
43. Torkura, K., Sukmana, M., Cheng, F., Meinel, C.: Cavas: Neutralizing application and container security vulnerabilities in the cloud native era. In: International Conference on Security and Privacy in Communication Systems, pp. 471–490. Springer, Cham. (2018).
44. Ullah, F., Raft, A., Shahin, M., Zahedi, M., Babar, M.: Security support in continuous deployment pipeline. In: Proceedings of 21th International Conference on Evaluation of Novel Approaches to Software Engineering, pp. 12 p. Cornell University Archive (2017).
45. SynopsysSoftware: BSIMM12.explanation of the activities. https://www.bsimm.com/framework/governance/compliance-and-policy.html

46. Luz, W. P., Pinto, G., Bonifácio, R.: Adopting DevOps in the real world: A theory, a model, and a case study. Journal of Systems and Software, 157, 110384 (2019).
47. Rafi, S., Yu, W., Akbar, M. A., Alsanad, A., Gumaei, A.: Prioritization based taxonomy of DevOps security challenges using PROMETHEE. IEEE Access, 8, pp. 105426-105446 (2020).
48. SynopsysSoftware: BSIMM12 Digest:The CISO's Guide to Next-Gen AppSec. https://www.synopsys.com/software-integrity/resources/ebooks/ciso-guide-modern-appsec.html