IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

A Survey of Security Integration Practices from DevOps to DevSecOps

[1]Shifa M. Shaikh, [2]Prof. Soma Ghosh

[1] Student, Department of Computer Engineering. and IT, COEP Technological University Pune, India

[2] Assistant Professor, Dept. of Computer Engineering. and IT, COEP Technological University Pune, India

***Abstract**:* DevSecOps approach based upon the security testing covers a range of testing methods, from static and dynamic analysis to container scanning and compliance checks, ensuring comprehensive security throughout the development lifecycle. By integrating these practices, organizations can proactively identify and address vulnerabilities, enhancing the resilience of their web applications in the face of evolving threats. This approach's value lies in its complexity and proactiveness. By adopting a DevSecOps approach, organizations can increase the reliability of their web applications by proactively blocking potential security holes. This summary explores the transformative impact of this methodology and highlights its ability for improving resilience for web applications leading ever evolving and complex threats of the digital age. By identifying real-time vulnerabilities and strategically remediating vulnerabilities, organizations can confidently navigate the dynamic cybersecurity landscape to ensure the security and stability of their web applications.

Index Terms - DevOps, DevSecOps, Software Vulnerability, Security, Literature Review, Methodology

I. INTRODUCTION

The "DevOps" (Development and Operations) method seeks to erase customary boundaries, or "silos," between both the development and use of software, resulting in Shift reduced the time required to commit a system change and deploy it in an actual production setting. The DevOps paradigm is currently widely used in the software development industry. Academic academics and industry practitioners have documented gains in business value, which explains the desire for adoption. The capacity to provide updates more rapidly and more frequently is the most noted advantage. However, organizations now confront extra challenges because of the rapid delivery procedures.

II. OVERVIEW OF DEVSECOPS AND ITS SIGNIFICANCE IN MODERN SOFTWARE DEVELOPMENT

DevSecOps is a way to integrate security into DevOps processes that have gained significant attention in recent years. It emphasizes collaboration, communication, and automation of software development, IT operations, and security teams. By incorporating security into the development lifecycle, organizations can detect and fix vulnerabilities early, making their web applications more secure.

Detecting the vulnerabilities early can help in, leading to more secure web applications. The key principles of DevSecOps include continuous security, shifting security left, and automating security testing. Continuous security involves integrating security at every stage of the software development lifecycle, ensuring that security measures are continuously applied and monitored. Shifting security left means addressing security concerns early in the development process, rather than as an afterthought. This proactive approach helps in identifying and addressing security vulnerabilities at an early stage, reducing the likelihood of costly fixes later in the development process.

Automating security testing involves using tools and scripts to automatically test and validate security measures, enabling faster and more efficient security assessments. By adopting DevSecOps practices, organizations can improve the overall security posture of their web applications, reduce security risks, and increase the speed of delivering secure software products to market. This proactive and collaborative approach to security aligns with the fast-paced nature of modern software development, contributing to a more secure digital environment for businesses and individuals alike.

DevSecOps, a practice that combines security testing into the software development process. It highlights the significance of automating security testing to improve the security of applications. DevSecOps aims to enhance app security by integrating security practices seamlessly into the development lifecycle. By incorporating security unit tests and automation tools, developers can create more secure applications without slowing down the development process.

III. BACKGROUND

3.1 DevSecOps Principles and Practices

This section defines key words and topics from the study. Finally, we compare our findings to relative reviews.

3.1.1 Continuous Software Engineering and its Techniques (CSE)

Continuous Software Engineering aims to sustain a cycle which is continuous of software engineering operations. Other than a sequential separate action carried out by multiple groups or divisions[1]. CSE leverages approaches like Continuous Integration to encourage continuous mobility. Identify continuous processes for Growth of the company, processes, scheduling, and strategy. This sector, development methodologies are widely employed and well established[2]. Investigation into continuous practices has concentrated primarily on development practices. [3][4][5]and[6]. According to studies, the most often utilized continuous processes in the domain are Continuous integration, Continuous Delivery, or Continuous Deploying [7]. As a result, we end up choosing 3 practices to evaluate in our study.
The following outline the meaning and connections in between these activities:

1. **Continuous integration (CI)**

CI, also known as Continuous Integration, is a development technique in which developers often merge their work (such as code changes) through the main branch of code. Typically, this occurs every day [7]. These improvements are confirmed by automated builds and testing [8]. This allows developers to notice and fix integration errors as soon as feasible [16][19].

2. **Continuous Delivery (CDE)**

This seeks to always maintain reliability and production readiness [9][3]. t To do this, code has to pass required assessments and undergo quality checks in a testing environment (like acceptance tests). Pull-based method: A team member with the necessary permissions determines which production-ready deliverables should be sent to the consumer and when. Deployment to the production setup, on the other hand, is done manually[5].

3. **Continuous Deployment (CD)**

When all necessary quality gates are cleared, CD expands on CDE by continually and automatically deploying software outputs to a production environment, [3]. Software updates are automatically distributed to production via the deployment pipeline using CD, a push-based methodology, without the need for human participation [5].

3.1.2 Development and Operations (DevOps)

By encouraging cooperation, collaboration, and integration between teams working on development and operations, the DevOps paradigm seeks to lessen the gap between them [11].DevOps is becoming more prevalent in the industry, but no universal definition exists [12]. Because of the significant overlap with continuous techniques, defining DevOps is challenging Stahl et al [7] provide meaning to distinguish between Automation continuous approaches as a result. Since they have tried to represent the collective understanding of these concepts, we use their definition of DevOps in our research. Throughout the product lifecycle, developers and IT operations collaborate to improve the pace and caliber of software deployment in a DevOps team. There has been a cultural shift in the way people work, and it has an enormous impact on teams and the companies they work for.

3.1.3 Development, Security and Operations (DevSecOps)

The application development practice known as "DevSecOps," or "development, security, and operations," automates integration of security practices at every stage of a development lifecycle, via the design phase to integration, testing, delivery, and deployment. The "DevSecOps paradigm" refers to the enhanced integration, cooperation, and collaboration among the development, operations, and security teams that results in the assimilation of security practices and concepts into DevOps. Researchers and professionals in the field have utilized variants of this phrase, such Security in DevOps[14] and DevOpsSec. [15] Nevertheless, the primary goal of these statements is to keep security at the forefront of the DevOps cycle.
The taking after preparing empowers accomplishing the guideline of DevSecOps:

Threat Displaying and Chance Evaluation: This is the handle of the ceaseless revelation of security danger outside and inner, secure against them and finding arrangements for these dangers & hazard and that as well proactively some time recently harm happens.

Risk Modeling: Risk Modeling is a preparation where you preempt an assault by making a demonstrate to see at the deviation in the handle, see for powerlessness, and insider assault by coordinating risk intel, consider all conceivable way an assault can happen and moderate the same.

IV. REVIEW METHODOLOGY

This section presents an overview of our research technique and the methodical strategy used to compile pertinent literature. We could not uncover a significant amount of scholarly research on the subject of DevSecOps after conducting an initial search of the literature. Consequently, we made the decision to carry out a multivocal literature review (MLR). All easily accessible material on a subject is referred to as multivocal literature [16]. This includes scholarly literature, white papers, blogs, and articles, among other things. This includes scholarly literature, white papers, blogs, and articles, among other things. The conclusions will offer a more thorough examination of the topic by using this variety of literature since they consider the opinions and viewpoints of specialists including academics, practitioners, independent investigators, developmental businesses, and others[16].
Among the previously released Reviews restricted to [17] is an MLR, which is on automation testing for software and the automating criteria that practitioners and researchers have proposed for what to automate and when. A Review giving a summary of DevOps is [18]. Though it's not the first for DevOps as far as we are aware, this has been an initial MLR on the topic. The fact that software

engineering (SE) practitioners generate a large amount of multivocal literature but do not publish it in scholarly journals [20] highlights the significance of MLRs (Multivocal Literature Review) in SE fields. But they draw attention to the fact that researchers are missing out on important current developments in SE practice by leaving this type of data out of systematic reviews.

4.1 Research Questions

Table 1 summarizes the research questions and their motivations. The research topics and process for performing the search were considered and agreed upon by all authors. The first author searched and screened studies with guidance from experienced experts. The steps involved are outlined in the following.

4.2 Search Approach

The steps are: First, utilize Google Scholar and Google Search to do an advanced search. The search engine Google will give priority to the various RQs by splitting the phrase into sections, each focusing on a different aspect of DevSecOps and a related word. The five search strings may be viewed in one. Then thoroughly reviewed the literature for each search, selecting only pertinent information for our main study by utilizing the other qualification and exclusion criteria. Procedure is depicted in Figure

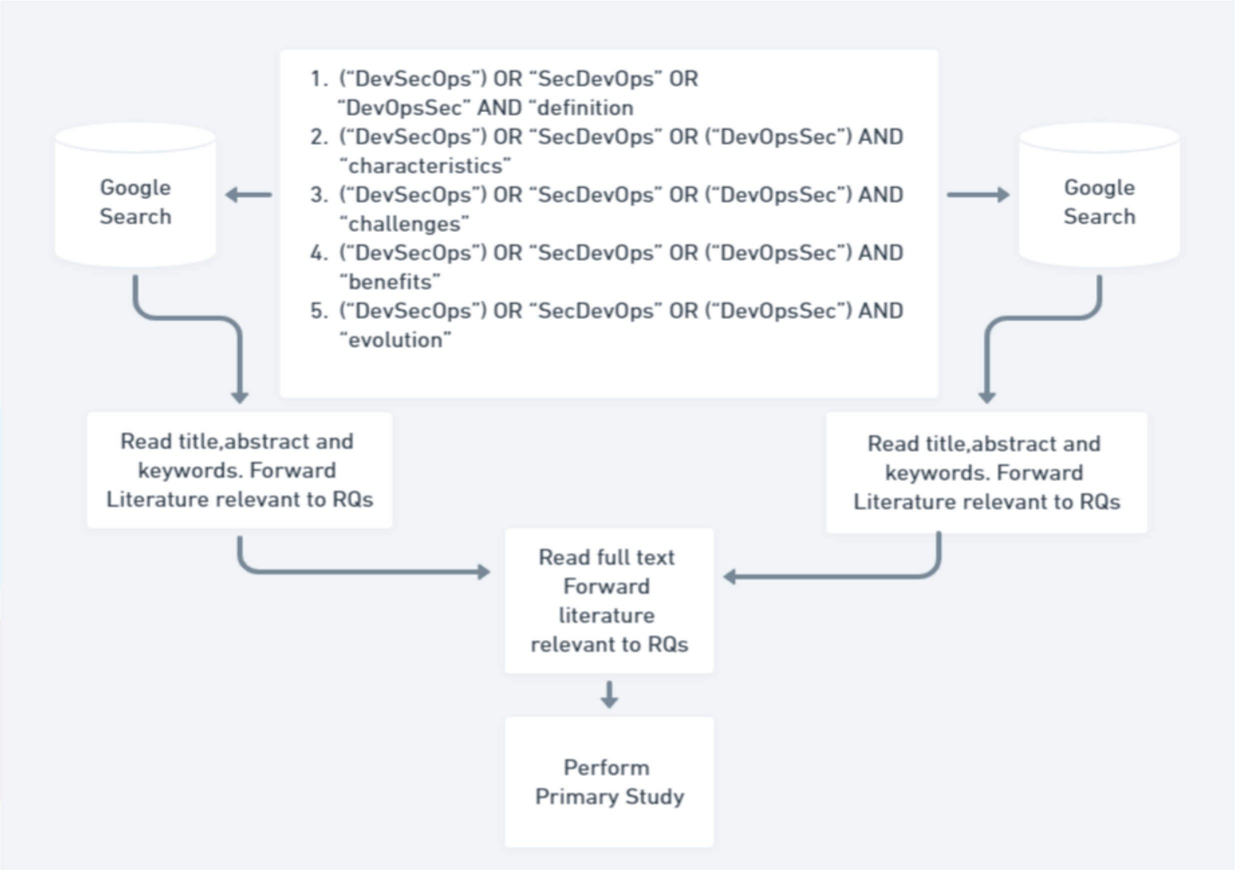
1. ("DevSecOps") OR "SecDevOps" OR "DevOpsSec" AND "definition
2. ("DevSecOps") OR "SecDevOps" OR ("DevOpsSec") AND "characteristics"
3. ("DevSecOps") OR "SecDevOps" OR ("DevOpsSec") AND "challenges"
4. ("DevSecOps") OR "SecDevOps" OR ("DevOpsSec") AND "benefits"
5. ("DevSecOps") OR "SecDevOps" OR ("DevOpsSec") AND "evolution"
Google Search
Google Search
Read title,abstract and keywords. Forward Literature relevant to RQs
Read title,abstract and keywords. Forward Literature relevant to RQs
Read full text Forward literature relevant to RQs
Perform Primary Study

Figure 1: Search Strategy on finding the relevant sources on DevOps Security.

Table 1: The research concerns.

Research Questions	Motivation
RQ1	In terms of security testing, what are the main pillars and practices of DevSecOps?
RQ2	What makes DevSecOps what it is?
RQ3	What are the expected advantages and difficulties of implementing DevSecOps?
RQ4	How do organizations adopt DevSecOps strategies for security testing and still preserve regulatory compliance?
RQ5	What are some of the solutions for challenges in DevSecOps?

V. RESULTS AND DISCUSSION

Findings of our research are presented in this section. First give a synopsis of original study before talking about the responses to our research questions.

Our outcomes may be divided into the four major categories shown below:

- People: This subject addresses problems with knowledge and abilities, the cooperation of DevSecOps diverse team members, and corporate culture (e.g., problems with team collaboration which leads to difficulties in adopting DevOps Security).
- Practices: The subject deals with problems combining security practices (such the challenges of merging manual security measures into DevSecOps) with DevOps or continuous processes (like CI/CD).
- Tools: This subject addresses problems with the tools used in DevSecOps, including use scenarios and pipeline security issues (such container vulnerabilities).
- Infrastructure: This subject addresses challenges associated with implementing DevSecOps in many infrastructure types, such as intricate cloud settings.

5.1 RQ1: In terms of security testing, what are the main pillars and practices of DevSecOps?

5.1.1 Test-driven security:

The idea that hackers can use their cell phones to crack encryption or get past many security system levels is a fantastic plot device for movies, but it seldom works in real life. Attackers typically target simple targets, such as outdated systems, web platforms with security vulnerabilities, administrative pages that are accessible online with passwords that are easy to guess, and security credentials that have been inadvertently disclosed in open-source code.

Application Security:

Numerous threats can target modern online apps. Every three years, the Open Web Applications Security Project (OWASP) publishes a top-10 list of the most frequently occurring assaults, which includes never-ending brute-force attacks, SQL injection, cross-site request forgeries, and cross-site scripting. Fortunately, the appropriate security measures may be applied at the appropriate locations to counteract any attack vector. We'll look more closely at the safeguards a DevOps team must have in place to make web applications secure in chapter 3, which addresses application security.

Infrastructure Security:

Safety of the infrastructure is important to a DevOps team even if they rely on IaaS to operate applications. Every system has access points, such as management panels, SSH gateways, and VPNs, that provide enhanced rights. As a company expands, extra caution must be taken to maintain system and network security while granting additional access and integrating more components.

Pipeline security:

The automated product delivery methods used by DevOps differ from the conventional operations that most security teams are accustomed to. A CI/CD pipeline compromise can provide an attacker complete control over the applications used in production. Integrity controls such as commits, or container signing can be used to secure the automated processes involved in delivering code to production systems. I'll go over ways to ensure the integrity of the code that runs in production and strengthen trust in the CI/CD pipeline.

Testing Continuously:

The security measures that have been put in place in each of the three regions I just described are still easy to deploy separately. The challenge arises from the need to constantly test and apply them. Test-driven security can help in this situation. Like test-driven development (TDD), test-driven development (TDD) advises developers to design tests that first simulate desired behavior before writing the code to put the tests into practice. TDS suggests creating security tests as a representation of the desired state first, and then putting the measures that pass the tests into action.

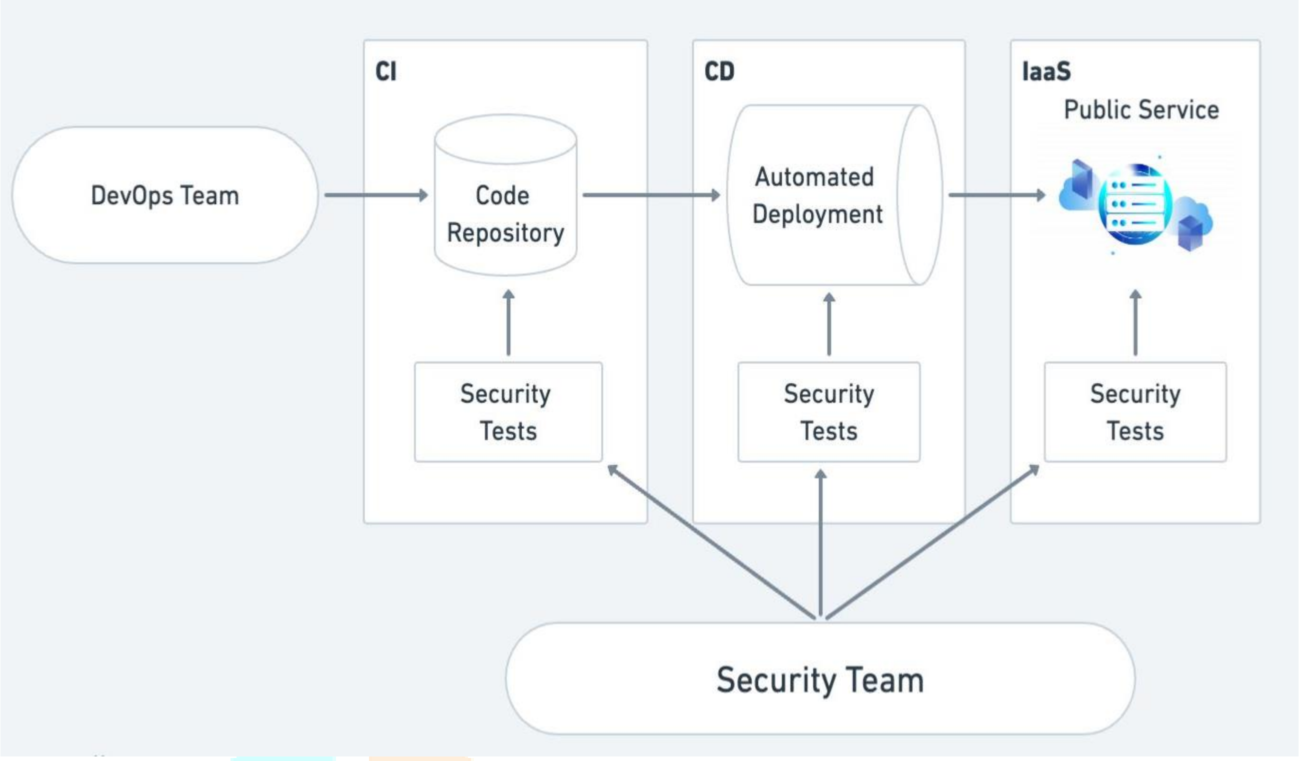

Figure 2: Test-driven security interfaces with CI/CD to execute security tests before deploying to production infrastructure.

5.1.2 Monitoring and responding to attacks:

Malware producers' scanners would quickly identify the machine and deliver one of the several exploit codes that Windows XP was susceptible to. The system was compromised in hours, and a loophole was left open to allow further infections to infect it. Not only was it entertaining to watch, but it also contributed to the teaching of a crucial lesson: no system on the internet is immune from assault at some point. Because of their complexity, modern businesses sometimes cannot afford to cover every area. Teams in charge of security must choose priorities. Three areas are the focus of our monitoring and attack response strategy.

Logging and Detecting Fraud:

In terms of security, we concentrate on two requirements:

- Finding irregularities in security
- Supplying forensic expertise for incident investigations

Detecting Intrusions:

It's challenging to detect breaches, and operations and security teams frequently need to collaborate closely. These systems can use bandwidth that should be allocated to running production services when implemented incorrectly. You'll learn how intrusion detection is efficiently integrated into DevOps using an incremental and conservative strategy.

Incident Response:

Organizations should follow six processes when responding to a security event. They are as follows.

- Preparation - Ensure that you have the basic minimum of incident response procedures in place.
- Identification - Determine quickly if an anomaly constitutes a security issue.
- Containment - Block the breach from spreading any further.
- Eradication refers to the removal of dangers from an organization.
- Recovery - Return the organization to regular functioning.
- Lessons learned -- Reflect on the situation and learn from it.

5.1.3Assessing risks and maturing security

A comprehensive continuous security plan extends beyond the technical requirements of setting security measures and responding to occurrences. Although it appears throughout the book, the "people" part of continuous security is extremely important when considering risk management.

Assessing Risks:

A solid risk management method must achieve three objectives:

- Run in tiny iterations, frequently and rapidly. Software and infrastructure are continually evolving, and companies must be able to communicate concerns without requiring weeks of procedures.
- Automate! This is DevOps, so carrying out things by hand should be an exception rather than the rule.
- Encourage everyone in the organization to participate in risk talks. Making safe goods and maintaining security is a collaborative effort.

Security Testing:

- Internally assessing the security of applications and infrastructure by techniques like checking for vulnerabilities, fuzzing, static code analysis, and configuration auditing. We'll go over numerous strategies that may be integrated into a CI/CD pipeline and used as part of a DevOps strategy's software development lifecycle (SDLC).
- Having external businesses examine the security of essential services. When correctly targeted, security audits add significant value to a company by introducing innovative ideas and perspectives to the security program. We'll talk about how to employ external audits and "red teams" effectively and make the most of their engagement.
- Create a bug bounty program. DevOps firms frequently embrace open source and make major portions of their source code public. These are excellent resources for outside security investigators who, for a few thousand dollars, can test your applications and report any security findings to you.

5.2 RQ2: What makes DevSecOps what it is?

Upon reviewing the research, the ideas that formed the basis and logic of development security operations as well as the strategies used for integrating security into software development processes immediately stood out as defining DevSecOps. Automation and the CAMS for short concepts form the foundation of DevSecOps[42][34], which incorporate sharing, measurement, automation, culture, and security from the outset:

Culture: Developers and operational teams work together more frequently in an environment that uses DevOps[42], with everyone accepting responsibility for the delivery of software to end users [44]. DevOps security refers to cooperating with a security team and encouraging an environment where development and operations cooperate to include security onto their job [33][36]. This entails embracing the security personnel in the planning phases and making sure that each is aware of their shared duty for maintaining privacy. [37][38] Practitioners recommend using agreed-upon criteria to create a collective accomplishment mindset with the objective of boosting security[38] and coordinating secure and business initiatives to make sure broad support and execution [39][40].

Automation: In DevOps, build, deployment, and testing automation is critical for quick development, deployment [42][6], and end-user feedback [44]. DevSecOps emphasizes the importance of automating security to stay up with DevOps' pace and scalability. The goal should be to fully automate security measures, allowing them to be implemented and managed without human intervention [43]. It has been critical to deploy automated in a way that doesn't impede swiftness, that might generate pause [23][40][41].

Measurement: In DevOps, measurement is keeping an eye on financial information and KPIs to evaluate the effects of freshly made launches on system dependability and pinpoint areas in need of development[42][44]. Using and developing metrics to keep an eye on threats and vulnerabilities during the process of building software is encouraged by security operations. [10]. Over the software development process, automatic security controls offer metrics for monitoring dangers and weaknesses in context, enabling companies to instantly verify an application's quality [34].

Sharing: Developers and operators share data, development instruments, and process management methods in DevOps [42][44]. The security team's involvement in the exchange of data that takes place in an automation environment is encouraged as DevSecOps. Notifying security teams about security measures helps improve security making practices ,the problems that operational and developers are having, and vice versa. [34].

Shift security to the left: The typical development process, securing, is the last stage. Privacy should move to the left, according to DevSecOps, which calls for its integration throughout every phase of development lifecycle. [36]. This requires that security teams participate in all iterations of the construction cycle, beginning at the planning stage. [23][42]. It also suggests that security advice is readily accessible to help operators and developers with security-related issues [43] [43] [37].

5.3 RQ3: What are the expected advantages and difficulties of implementing DevSecOps?

5.3.1 Advantages:

The following section presents a summary, depending to the information available, the pros obtained by DevOps Security and associated with:

Shifting Security to the Left:

It is easier to plan and carry out the introduction of safeguards across the development procedure without experiencing delays or generating issues by introducing security controls once networks are operational when security specialists are involved from the start. [23]

Automating security:

Making automated security measures makes error detection, alerting, correction, countermeasures, and forensics swift, scalable, and efficient. This lowers the chance of faults and the amount of spent time on mistakes. It also makes it easier to assess risk and create guidelines and processes[38].Scripting automates logging and documentation, standardizes processes, and generates predetermined results for tests that are related. [39] Furthermore, allowing security tests automatic execution frees up the developer's time so they can write code instead of performing tests [40]. This lowers the danger of manual error [39].

Security teams may avoid laborious, repetitive, and ineffective duties because they can store security policy templates created during development in a shared repository and use them sans requiring spending time setting up all the environments when initiating a new project. [36]

Value addition:

[41] [38] that mistakes may be costly, and it is less expensive to deploy security right away than to wait for whatever to happen [38] alludes to another poll that indicates high-performing companies utilize twenty-two percent less time on rework and unscheduled work. By discovering and assessing problems early on, tracking and quantifying vulnerabilities in security avoids implementation delays [38]. This reduces the cost of producing, identifying, and fixing mistakes. [36]

5.3.1 Difficulties:

Not willing to modify and adapt:

The first obstacle is usually the 'people challenge' that comes with every move. Most companies and workers are accustomed to the conventional way of doing things. Convincing them to embrace the new DevSecOps approach will take time. The security team and the development and operational groups must work effectively together. This will assist in improving the whole project.

Practice challenge:

Traditional DevOps techniques prioritize speed to get projects into production faster. By "shifting left" and including additional security checks at all levels of the SDLC, the pace of the DevOps platform is unavoidably reduced. This may cause conflict between the DevOps and security teams. To strike a balance between speed and security, teams must be patient enough to make informed judgments.

Challenges with tool integration and documentation:

The third obstacle is called the 'tool challenge'. Working with existing toolsets in the DevOps methodology is challenging. Furthermore, combining security tools into established corporate practices is more complex than one may expect. In addition, the team has another challenge: a lack of appropriate documentation.

You may overcome this obstacle by generating better documentation. This will allow the teams to refer and integrate the technologies more effectively into the business context.

Multi-cloud environment difficulties:

The fourth difficulty is the 'infrastructure challenge'. Shifting resources to the cloud is a widely adopted and current trend in the software business. The migration to the cloud occurs for distinct reasons. However, safeguarding resources in a multi-cloud context is a complex procedure. Data that is continuously changing in the cloud and must be safeguarded is an extremely complex issue. This is just another barrier in the shift to a DevSecOps framework.

This difficulty may be solved by concentrating on data security in addition to SDLC and implementing hybrid life cycles.

Cannot completely automate:

The fifth obstacle that we will explore is connected to the 'practice challenge'. DevOps procedures are primarily automated to allow for faster releases. However, when security is included, the processes slow down since most security measures require human input. One solution is to employ DevSecOps tools across the SDLC to avoid completely slowing down the process. These are some of the issues that businesses experience when attempting to migrate to a DevSecOps environment. While shifting presents numerous issues, there are only a handful of them, as well as solutions to them!

5.4 RQ4: How do organizations adopt DevSecOps strategies for security testing and still preserve regulatory compliance?

Adopting DevSecOps methodologies for security testing while maintaining regulatory compliance requires many essential practices:

5.4.1 Integrate security early:

Integrating security methods early in the development process, such as security testing, code evaluation, and vulnerability scanning, to detect vulnerabilities sooner and reduce rework.

5.4.2 Automated Security Testing:

To ensure continuous security assessment without slowing development, use automated security testing tools and processes including static application security testing (SAST), dynamic application security testing (DAST), and interactive application security testing (IAST).

1. **White Box Testing-SAST: Static Application Security Testing (SAST):**

Also known as white box testing. SAST is a free-box testing method that can analyze source code, bytecode, or binary code for vulnerabilities. It works in the preliminary stages of development and allows developers to identify problems in the source code before the application is compiled or run. The SAST tool scans for potential vulnerabilities such as SQL injection, cross-site scripting (XSS), and vulnerabilities. SAST allows developers to fix code-level vulnerabilities by providing detailed instructions, thereby increasing the overall security of the application. White Box Security Testing Testers have access to principles, design, and implementation. Such tests represent the developer’s approach.

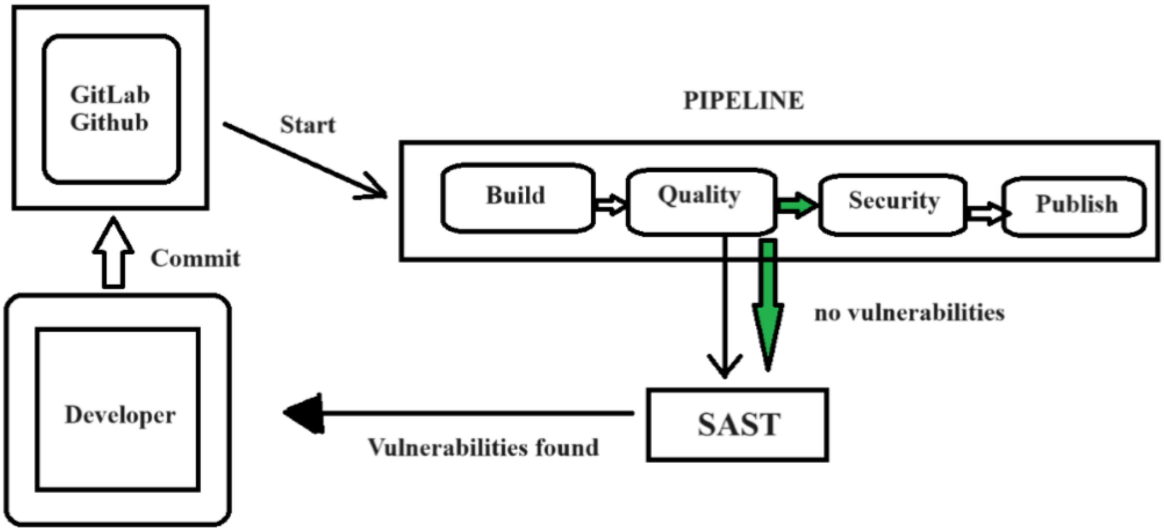

Figure 3: Static Application Security Testing (SAST)

2. **Black Box Testing: Dynamic Application Security Testing (DAST)**

DAST, compared to SAST, is a black box testing method that tests externally running applications. The DAST tool simulates a real-world attack by sending requests to the web and analyzing the responses. By interacting with applications, DAST can identify vulnerabilities that may not be apparent in the source code, such as configuration errors, validation issues, and error messages. DAST provides insight into application security from an external perspective, helping organizations understand how their applications behave in different scenarios. DAST works by simulating an attacker's behavior to simulate automated attacks against your application. The goal is to find unexpected results or results that an attacker can use to compromise the application. Because DAST tools have no internal information about the application or source code, they attack in the same way as external hackers. That is, it uses the same limited knowledge and information about the application.

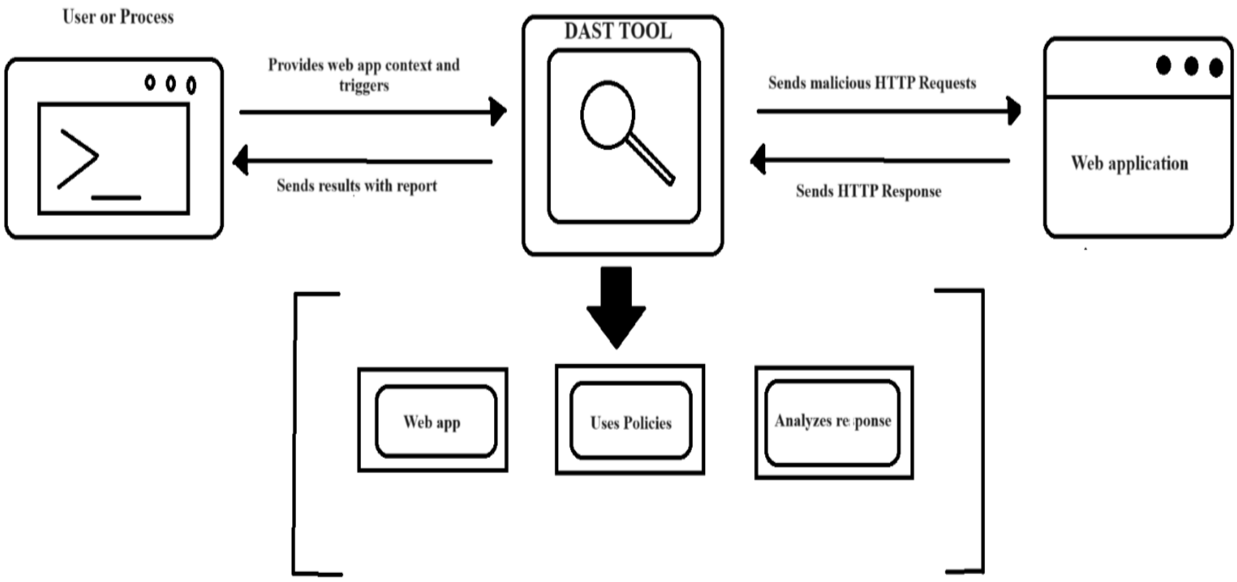

Figure 4: Dynamic Application Security Testing (DAST)

3. **Interactive Application Security Testing (IAST)**

IAST combines elements of SAST and DAST by embedding security sensors in the application at runtime. Sensors monitor the application's behavior and interactions with underlying policies. The IAST tool provides instant feedback to developers to identify real-time vulnerabilities in their code. IAST provides information about security issues, allowing developers to understand root causes and fix vulnerabilities. This protection improves the development process and ensures that security is seamlessly integrated into the application architecture. Integrating these security measures into the DevSecOps pipeline allows organizations to build and deploy web applications securely. By leveraging the power of SAST, DAST, and IAST, teams can identify, measure, and remediate vulnerabilities to improve the security of web applications. This article will take a closer look at each experiment, examining their approaches, results, and best practices for effective DevSecOps frameworks.

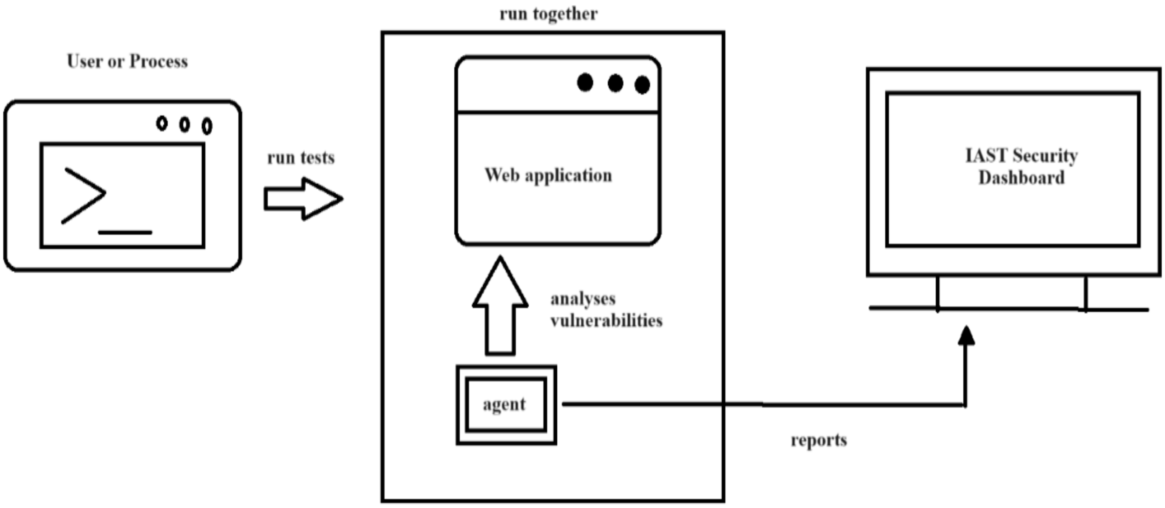

Figure 5: Interactive Application Security Testing (IAST)

These mechanized tests ought to be conducted for checking to distinguish inconsistencies in code.

(A) Monitoring: There needs to be persistent checking of logs, peculiarities. The observation ought to incorporate all assets counting Character and Get to for insider risk, PAM, Clients, Resources, instruments, and integration with risk displaying, Risk Intel, UEBA and Security Orchestration.

(B) Operations: This would incorporate Danger Intel, Occurrence reaction, Measurable, Survey of Powerlessness, Interruption Location, App Assault Location, Holder Security.

(C) Automation: Approaches once drafted should be held and overhauled sometimes. These arrangements ought to be computerized, and these ought's to be enacted when required.

(D) Red Groups: There ought to be a ruddy to test the organization's security development and help organizations distinguish crevices in their security pose, in terms of individuals prepare and system.

(E) Train the coach: Presently you have security implanted in Advancement and Operation, you require somebody in the group who can help when designers have an issue, distinguish a malevolent insider from a blameless designer and take remedial action.

(F) DevSecOps empowers high-quality application advancement without compromising the speed of conveyance. The blend of the right culture, element, Computerization, Sharing, and Moving security cleared out would permit association embrace DevSecOps and benefit from same. Comparison of traditional DevOps and DevSecOps approaches.

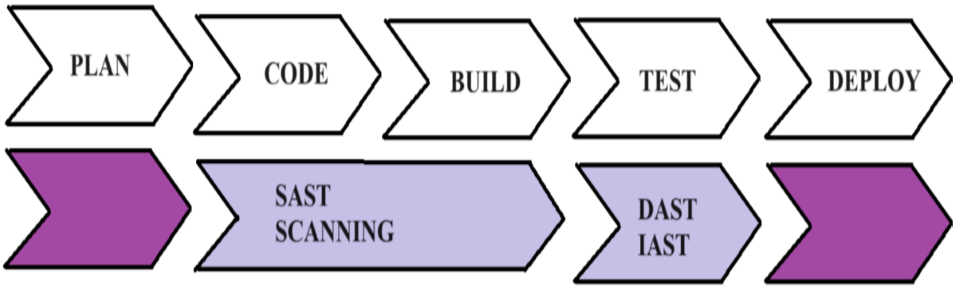

Figure 6: Workflow of SAST, DAST, IAST

5.4.3 Compliance as Code:

Use infrastructure as code (IaC) and compliance as code (CaC) techniques to automate compliance tests and verify that security and regulatory standards are satisfied throughout the development process.

5.4.4 Continuous Monitoring and Compliance:

Use continuous monitoring tools and techniques to detect and respond to security risks in real time while maintaining regulatory compliance.

5.5.5 Cross-Functional cooperation:

Encourage cooperation across development, operations, and security teams to ensure that security is integrated into all stages of the development and deployment processes.

5.5.6 Risk Assessment and Management:

Conduct frequent risk assessments to identify security vulnerabilities, prioritize them based on their effect and likelihood, and then manage them appropriately. Organizations can successfully implement DevSecOps techniques for security testing whereas upholding regulatory compliance by adhering to these guidelines.

5.5 RQ5 What are some of the solutions for challenges in DevSecOps?

The following are some ways to tackle the DevSecOps difficulties mentioned:
Encourage cooperation and dialogue amongst the teams working on development, operations, and security. Promote a common knowledge of the advantages of DevSecOps.

- Provide team members with training, workshops, and seminars to help them understand the value of using DevSecOps techniques.
- To reduce conflict between the DevOps and security teams, prioritize and balance security and speed.
- To find problems early and make sure security is a crucial component of the development process, use "shift-left" testing and integrate security tests at each stage of the software development lifecycle (SDLC).
- Enhance documentation to make it easier to integrate security tools into current DevOps practices. Investigate and implement DevSecOps-specific tools that make it easier to manage security aspects.
- Pay attention to data security in conjunction with the SDLC and adopt hybrid lifecycles to make sure that security measures are in place when moving resources to the cloud.
- Secure transient data in a multi-cloud environment by implementing encryption, access controls, and robust monitoring.
- Take advantage of DevSecOps tools that automate security processes while enabling necessary human input. This allows developers to maintain speed without compromising security.

V. Future Research

In the future, it would be intriguing to poll companies in order to maybe expand the breadth of this research's coverage of DevSecOps. It's also interesting to look at the suggested practices from this study and see how they impact the surrounding environments (business, customers, development, and operations) to determine best practices. Examining and recommending platforms or architectures for implementing DevSecOps would be the next stage. One replacement of "security as code" in DevSecOps is the microservices architecture used in [32], which examines continuous software engineering.

Table 2: Future Research on the areas that are explored in this study

Tools	Need for security tools targeted at developers as opposed to security professionals, or developer-centered safety measures (i.e., developer-centered security) The need for safeguards that facilitate the rapid deployment cycles of DevSecOps Security testing for applications as a service
Practices	Continuous methods for identifying and managing vulnerabilities Verified empirical metrics for security in DevOps Agreement on shift left and ongoing security
People	Definition of security responsibilities in DevOps Security Need for socio technical study in DevSecOps on concerns pertaining to people
Infrastructure	Paradigms validated empirically in a range of environmental settings

VI. Conclusion

This survey summarizes the research we conducted on DevSecOps to determine its definition, what DevSecOps indicates for an organization in terms of the values and procedures they should follow, what obstacles they would encounter when trying to implement DevSecOps, the advantages if it is successful.
We listed the advantages and difficulties of putting DevSecOps into practice. The difficulties we found are a sign of DevSecOps' infancy rather than a reason to avoid deploying it.However, better methods, technologies, etc., are likely to overcome these as [1]DevSecOps grows. Benefits that have been identified show that it is maturing for instance, by leading to a reduction in physical labor and less unforeseen work.

[1]

REFERENCES

[1] Fitzgerald, Brian, and Klaas-Jan Stol. "Continuous software engineering: A roadmap and agenda." *Journal of Systems and Software* 123 (2017): 176-189.

[2] Bosch, Jan. "Continuous software engineering: An introduction." *Continuous software engineering*. Cham: Springer International Publishing, 2014. 3-13.

[3] Shahin, Mojtaba, Muhammad Ali Babar, and Liming Zhu. "Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices." ("Continuous Integration, Delivery and Deployment: A Systematic Review on ...") ("Continuous Integration, Delivery and Deployment: A Systematic Review on ...") *IEEE access* 5 (2017): 3909-3943.

[4] Schermann, Gerald, et al. *An empirical study on principles and practices of continuous delivery and deployment*. No. e1889v1. PeerJ Preprints, 2016.

[5] Shahin, Mojtaba, et al. "An empirical study of architecting for continuous delivery and deployment." *Empirical Software Engineering* 24 (2019): 1061-1108.

[6] Zahedi, Mansooreh, Roshan Namal Rajapakse, and Muhammad Ali Babar. "Mining questions asked about continuous software engineering: A case study of stack overflow." ("Mining Questions Asked about Continuous Software Engineering ...") ("Mining Questions Asked about Continuous Software Engineering ...") *Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering*. ("Evaluation and Assessment in Software Engineering - EASE 2020 ...") ("Evaluation and Assessment in Software Engineering - EASE 2020 ...") 2020.

[7] Stahl, Daniel, Torvald Martensson, and Jan Bosch. "Continuous practices and devops: beyond the buzz, what does it all mean?." ("Continuous practices and devops: beyond the buzz, what does it all mean ...") ("Continuous practices and devops: beyond the buzz, what does it all mean ...") *"2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*." ("2017 43rd Euromicro Conference on Software Engineering and Advanced ...") ("2017 43rd Euromicro Conference on Software Engineering and Advanced ...") IEEE, 2017.

[8] Leppänen, Marko, et al. "The highways and country roads to continuous deployment." *Ieee software* 32.2 (2015): 64-72.

[9] Chen, Lianping. "Continuous delivery: Huge benefits, but challenges too." *IEEE software* 32.2 (2015): 50-54.

[10]Checkmarx, An Integrated Approach to Embedding Security into DevOps, 2020. URL https://www.checkmarx.com/ebooks/an-integratedapproach-to-embedding-security-into-devops

[11] Bass, Len, Ingo Weber, and Liming Zhu. *DevOps: A software architect's perspective*. Addison-Wesley Professional, 2015.

[12] Leite, Leonardo, et al. "A survey of DevOps concepts and challenges." *ACM Computing Surveys (CSUR)* 52.6 (2019): 1-35.

[13] Chen, Lianipng, Muhammad Ali Babar, and He Zhang. "Towards an evidence-based understanding of electronic data sources." *14th International conference on evaluation and assessment in software engineering (EASE)*. ("(PDF) 14th International Conference on Evaluation and Assessment in ...") ("(PDF) 14th International Conference on Evaluation and Assessment in ...") ("(PDF) 14th International Conference on Evaluation and Assessment in ...") BCS Learning & Development, 2010.

[14] Mohan, Vaishnavi, and Lotfi Ben Othmane. "Secdevops: Is it a marketing buzzword? -mapping research on security in devops." ("SecDevOps: Is It a Marketing Buzzword? - ResearchGate") ("SecDevOps: Is It a Marketing Buzzword? - Mapping Research on Security ...") ("SecDevOps: Is It a Marketing Buzzword? - Mapping Research on Security ...") *2016 11th international conference on availability, reliability, and security (ARES)*. ("2016 11th International Conference on Availability, Reliability and ...") ("2016 11th International Conference on Availability, Reliability and ...") IEEE, 2016.

[15] Bird, Jim. *DevOpsSec: Delivering secure software through continuous delivery*. O'Reilly Media, 2016.

[16] Ogawa, Rodney T., and Betty Malen. "Towards rigor in reviews of multivocal literatures: Applying the exploratory case study method." ("Towards Rigor in Reviews of Multivocal Literatures: Applying the ...") ("Towards Rigor in Reviews of Multivocal Literatures: Applying the ...") *Review of educational research* 61.3 (1991): 265-286.

[17] Garousi, Vahid, and Mika V. Mäntylä. "When and what to automate in software testing? A multi-vocal literature review." *Information and Software Technology* 76 (2016): 92-117.

[18] de França, Breno B. Nicolau, Helvio Jeronimo, and Guilherme Horta Travassos. "Characterizing DevOps by hearing multiple voices." *Proceedings of the XXX Brazilian Symposium on Software Engineering*. 2016.

[19] Garousi, Vahid, Michael Felderer, and Tuna Hacaloğlu. "Software test maturity assessment and test process improvement: A multivocal literature review." ("Software test maturity assessment and test process improvement: A ...") ("Software test maturity assessment and test process improvement: A ...") ("Software test maturity assessment and test process improvement: A ...") *Information and Software Technology* 85 (2017): 16-42.

[20] Garousi, Vahid, Michael Felderer, and Mika V. Mäntylä. "The need for multivocal literature reviews in software engineering: complementing systematic literature reviews with grey literature." ("The need for multivocal literature reviews in software engineering ...") ("The need for multivocal literature reviews in software engineering ...") *Proceedings of the 20th international conference on evaluation and assessment in software engineering*. ("Past Proceedings - EASE 2020 - Evaluation and Assessment in Software ...") ("Past Proceedings - EASE 2020 - Evaluation and Assessment in Software ...") 2016.

[21] Myrbakken, Håvard, and Ricardo Colomo-Palacios. "DevSecOps: a multivocal literature review." *Software Process Improvement and Capability Determination: 17th International Conference, SPICE 2017, Palma de Mallorca, Spain, October 4–5, 2017, Proceedings*. Springer International Publishing, 2017.

[22] Goldschmidt, Murray, and Michael McKinnon. *Devsecops-agility with security*. Technical report, Sense of Security, 2016.

[23] Shackleford, Dave. "The devsecops approach to securing your code and your cloud." ("The DevSecOps Approach to Securing Your Code and Your Cloud - DevOps.com") ("The DevSecOps Approach to Securing Your Code and Your Cloud - DevOps.com") *SANS Institute InfoSec Reading Room A DevSecOps Playbook* (2017).

[24] Wettinger, Johannes, Uwe Breitenbücher, and Frank Leymann. "Dyn tail-dynamically tailored deployment engines for cloud applications." *2015 IEEE 8th International Conference on Cloud Computing*. IEEE, 2015.

[25] K. Lindros. How to craft an effective devsecops process with your team.https://goo.gl/ppWtjx, 2016.

[26] D. Shackleford. A devsecops playbook. SANS Institute InfoSec Reading Room, 32016. A DevSecOps playbook

[27]. M. Elder. Security considerations for devops adoption. https://goo.gl/b0CStP,2014.

[28] Whitehat Security. Devops invites security to "join the party".https://goo.gl/spj0wK, 2016.

[29] M. Hornbeek. Devops makes security assurance affordable. https://goo.gl/g0iKfZ,2015.

[30] A. Cureton. Building security into devops: Is devsecops the beginning of the future?https://goo.gl/Npv2Py, 2017.

[31] Trihinas, Demetris, et al. "Devops as a service: Pushing the boundaries of microservice adoption." *IEEE Internet Computing* 22.3 (2018): 65-71.

[32] O'Connor, Rory V., Peter Elger, and Paul M. Clarke. "Continuous software engineering—A microservices architecture perspective." *Journal of Software: Evolution and Process* 29.11 (2017): e1866.

[33] Claps, Gerry Gerard, Richard Berntsson Svensson, and Aybüke Aurum. "On the journey to continuous deployment: Technical and social challenges along the way." ("On the journey to continuous deployment: Technical and social ...") ("On the journey to continuous deployment: Technical and social ...") *Information and Software technology* 57 (2015): 21-31.

[34] S. Vonnegut. 4 keys to integrating security into devops 2016.

[35] Humble, Jez, and Joanne Molesky. "Why enterprises must adopt devops to enable continuous delivery." *Cutter IT Journal* 24.8 (2011): 6.

[36] S. Lietz. Shifting security to the left. https://goo.gl/sbheKS, 2016.

[37] G. Bledsoe. Getting to devsecops: 5 best practices for integrating security intoyour devops". https://goo.gl/ZPzgxa, 2016.

[38] F. Lim. Devsecops is the krav maga of security. https://goo.gl/BH4MS2, 2016

[39] S. Lietz. Principles of devsecops. https://goo.gl/N8zcXV, 2015.

[40] T. Greene. What security teams need to know about devops.https://goo.gl/c8VOn4, 2016.

[41] MacDonald, N., and I. Head. "Devsecops: How to seamlessly integrate security into devops." *Gartner Research* (2016).

[42] C. Caum. Getting started with policy-driven development and devsecops.https://goo.gl/AevVcX, 2016.

[43] Anonymous User. Security breaks devops – here's how to fix it.https://goo.gl/Yr1jk3, 2015.

[44] Fitzgerald, Brian, and Klaas-Jan Stol. "Continuous software engineering: A roadmap and agenda." *Journal of Systems and Software* 123 (2017): 176-189.