Integrating DevSecOps into the software development lifecycle: A comprehensive model for securing containerized and cloud-native environments

Bohdan Leshchenko[1,*,†], Bohdan Snisar[1,†], Anton Stupak[1,†] and Viacheslav Osadchyi[2,†]

[1] *Zhytomyr Polytechnic State University, 103 Chudnivsyka str., 10005 Zhytomyr, Ukraine*
[2] *Borys Grinchenko Kyiv Metropolitan University, 18/2 Bulvarno-Kudriavska str., 04053 Kyiv, Ukraine*

Abstract

The increased use of containerized and cloud-native environments necessitates integrating security measures throughout the entire Software Development Lifecycle (SDLC). This study proposes a comprehensive DevSecOps model designed to address modern infrastructures' security challenges. Our model prioritizes the continuous inclusion of security measures from the initial planning stages to the secure decommissioning of applications. Key elements of the model are improved governance of security, frequent auditing, disaster recovery planning, and a focus on continuous innovation within SDLC. The proposed approach offers a robust basis for protecting development processes, ensuring resilience, and maintaining compliance in rapidly evolving technological environments by integrating these activities into the DevOps framework. The practical applicability of the model is validated by comparing it against the existing frameworks and its prospective capacity to significantly enhance security posture within organizations working with containerized and cloud-native environments.

Keywords

DevSecOps, Software Development Lifecycle, SDLC, containerized environments, cloud-native security, security governance, continuous integration

1. Introduction

1.1. Relevance of the topic

In today's fast-paced IT landscape, incorporating security practices into development and operational workflows is critical, giving rise to the DevSecOps approach. DevSecOps is an evolution of traditional DevOps methodology that emphasizes embedding security controls early in the SDLC, addressing potential security issues from the start. This proactive integration is critical because it ensures that security is not an afterthought but a fundamental part of the development process.

The problems created by traditional security approaches, which frequently fail to keep up with modern IT systems' quick deployment cycles, drive the change to DevSecOps. By adding security measures into continuous integration and continuous deployment (CI/CD) pipelines, DevSecOps improves the capacity to discover and remediate vulnerabilities early, minimizing security breach risk [1, 2].

While containers provide agility and efficiency, they are vulnerable to certain security flaws. These include vulnerabilities in container images, misconfigurations, and unsafe runtime environments. Containers frequently employ images from public sources, which may include obsolete or vulnerable components. Furthermore, containers may contain unneeded software, which expands the attack surface and poses significant security threats if not adequately controlled [3].

As more organizations move to containerized systems, the requirement for integrated security solutions grows. Containers make installing and scaling programs easier but may also provide entry sites for malicious attacks if not properly secured. The significance of safeguarding these environments is clear as organizations strive to secure their applications and data from possible breaches [4].

Several issues arise when traditional security measures are combined with the DevOps model. These include incompatibilities with quick development cycles, challenges in automating security processes, tool complexity and integration concerns, configuration management issues, container vulnerabilities, and cultural and organizational hurdles. Addressing these difficulties is critical for the successful implementation of DevSecOps, which allows organizations to develop safe software quickly and effectively [5].

DevSecOps practices are essential for improving the security of containerized environments, as they address unique security challenges, ensure a more secure and

CPITS-II 2024: Workshop on Cybersecurity Providing in Information and Telecommunication Systems II, October 26, 2024, Kyiv, Ukraine
* Corresponding author.
† These authors contributed equally.
zogyyy1@gmail.com (B. Leshchenko);
b.snisar@ukr.net (B. Snisar);
s2pak.anton@gmail.com (A. Stupak);
v.osadchyi@kubg.edu.ua (V. Osadchyi)

0009-0001-5781-3518 (B. Leshchenko);
0009-0007-0091-0943 (B. Snisar);
0009-0008-1247-5990 (A. Stupak);
0000-0001-5659-4774 (V. Osadchyi)

resilient IT infrastructure, and ultimately improve cybersecurity in modern software development.

1.2. Research objectives and goals

This research aims to model all the practices under the framework and integrate them into the more extensive software development process, emphasizing cloud-native security. Our approach aims to enhance security by integrating security measures with DevSecOps in mind at each phase of development and operations. Identifying and addressing security vulnerabilities early on aims to reduce the risk of potential data breaches and other security threats. Ultimately, the research aims to establish best practices for implementing DevSecOps in a cloud-native environment. Enhancing overall security posture and resilience is crucial.

This research has several goals: identification and analysis of the current challenges. This includes examining the limitations of current security methodologies.

Additionally, the research will focus on developing strategies for integrating security measures seamlessly into the development process. By understanding the current challenges and limitations, the goal is to create a more robust and proactive security framework that can adapt to the evolving threat landscape. Ultimately, the aim is to provide organizations with practical guidance on implementing DevSecOps effectively in a cloud-native environment, ensuring that security is prioritized from the outset of the development process.

We focus on developing an expanded DevSecOps model that includes additional stages and practices such as security governance, disaster recovery, continuous innovation, and secure decommissioning.

1.3. Research methods

Our work uses a complex methodological approach to create a complete DevSecOps model. The model should be designed specifically for protecting containerized and cloud-based environments. The research methodologies were devised to address the study's objectives and goals methodically.

The first method involves a thorough literature review, which is the foundation for understanding existing DevSecOps methods and unique security problems. This review included academic journals, conference papers, and business publications. "Gray" literature and materials from third-party vendors were also used in our work. The literature research gave crucial insights into the growth of security procedures in modern IT infrastructures and gaps in existing models, which influenced the succeeding phases of the research.

Based on the findings of the literature overview, a comparative analysis was performed to assess the efficiency of existing DevSecOps models and frameworks. This investigation thoroughly evaluated various models based on factors such as security integration into the SDLC, adaptation to cloud-native environments, and scalability in complex IT ecosystems. The comparison research highlighted substantial strengths and limitations in the existing models, enabling the identification of particular areas for development. This stage was critical in determining the design of the new, expanded DevSecOps model suggested in this study.

The third part of the research concluded with the development of a new model. Drawing on the findings of the literature study and comparative analysis, the expanded model was created to fill the gaps in current frameworks.

The new model's structure is purposefully linked with the SDLC to ensure easy incorporation into existing development processes, making it both practical and successful.

The research uses these methodologies to establish a robust and adaptive DevSecOps model that organizations can use to improve the security of their software development processes, particularly in containerized and cloud-native environments.

1.4. Practical significance

This research offers a comprehensive DevSecOps model for organizations to integrate security into their software development processes. This model helps mitigate risks and enhance overall security posture in modern cloud infrastructures.

Our research will introduce an extended DevSecOps model, incorporating security governance, regular auditing, disaster recovery planning, and continuous innovation. This model ensures that security is integrated into development and maintained as the organization evolves.

Its alignment with real-world operational practices further reinforces the model's practical applicability. Its design should be easily integrated into existing DevOps workflows, allowing organizations to adopt it without significantly disrupting their current processes. This ease of integration is essential for encouraging widespread adoption, as it reduces the barriers to implementing comprehensive security measures in cloud-native environments.

Our research should provide a scalable, adaptable, and comprehensive security model for organizations to protect their containerized cloud-native environments, enhancing security during development and ensuring ongoing protection against emerging threats.

2. Literature review and theoretical foundations

2.1. Software development life cycle

Understanding the Software Development Lifecycle (SDLC) is essential as it offers an organized system for overseeing software development, guaranteeing productivity, security, and appropriate resource use at every project stage. SDLC provides an organized technique for developing software, ensuring that every stage—from design and planning to implementation and upkeep—carefully handles essential details. Researchers and practitioners can significantly improve the security of containerized applications, including security considerations at every level.

Several studies [6, 7] have examined the phases and models of various approaches of different SDLCs. Fig. 1 illustrates the salient features of these studies.

Consequently, the process used to design, develop, and test software is called the SDLC. Planning, design, implementation, testing, deployment, and maintenance are some of the phases that make it up. Specific tasks, like gathering requirements, coding, testing for bugs, and software deployment, are part of each phase.

Models such as waterfall, iterative, spiral, and agile offer different approaches to structuring these phases to maximize development. By offering a methodical framework and enhancing planning, visibility, risk management, and customer satisfaction, the SDLC helps manage software development.

By comprehending the advantages and drawbacks of each methodology, software development teams can make well-informed decisions and adopt the most sustainable approach, thereby enhancing the likelihood of successful project outcomes in the long run [8].

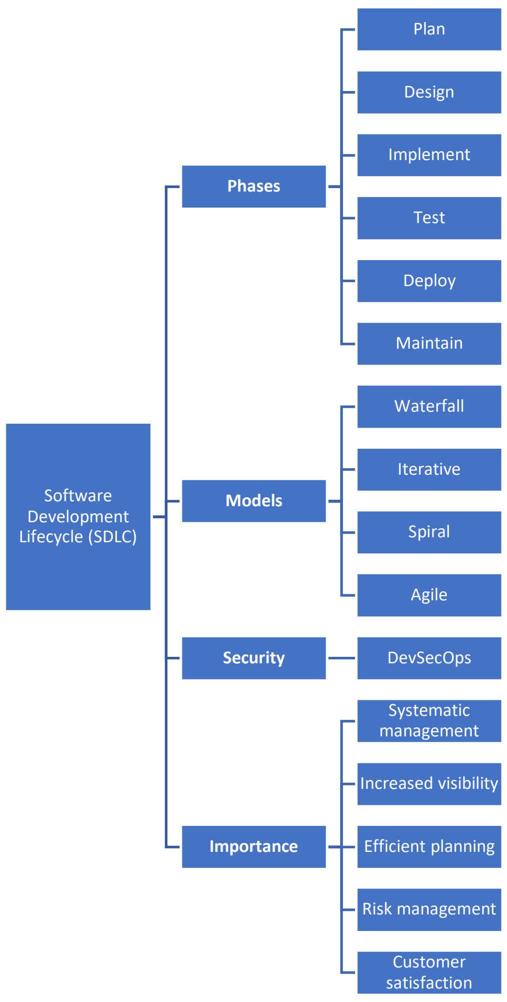

Figure 1: Key Aspects of the SDLC

As suggested by Olorunshola et al., utilizing two or more methods within a single project is recommended because choosing a specific method can be challenging for a company [9].

Unfortunately, their suggestions do not cover all modern SDLC. Despite the numerous techniques and models suggested, incorporating security remains a challenge.

2.2. DevSecOps

Today, most teams recognize that security is integral to the software development lifecycle. Security can be addressed throughout the development lifecycle by following DevSecOps practices and conducting security assessments throughout the entire SDLC process.

In their research [10], Rafiq Ahmad et al. classified significant studies through a systematic literature review. They have identified 145 security risks and 424 best practices for managing security via DevSecOps. They proposed the following six phases of DevSecOps: requirement engineering (RE), design, development/coding, testing, deployment, and maintenance.

Using these phases, researchers and practitioners can create robust security plans that tackle the difficulties presented by containerized environments, ultimately producing more resilient and secure applications.

The term *DevSecOps* (an organizational software engineering culture) means the processes of development (Dev), security (Sec), and operations (Ops). The ultimate goal of DevSecOps is to achieve safe and rapid code release. Security was traditionally seen as a distinct stage that came after the development cycle and occasionally even after deployment. However, with the introduction of DevSecOps, security procedures are now integrated into the whole software development lifecycle, completely changing the original strategy. Continuous security involves persistent monitoring and real-time insight into security vulnerabilities at every stage of the DevSecOps lifecycle [11].

DevSecOps' primary idea is based on the principle of "shifting left" [5], which involves incorporating security early in the development process. This technique enables the early discovery and remediation of vulnerabilities, lowering the cost and complexity of addressing security concerns later in the development cycle. Fig. 2 portrays DevSecOps as DevOps with continuous security assurance, where security controls may be included throughout the DevOps workflow [12].

Kumar and R. Goyal described the stages of the continuous security process in their paper [12]. Their concept consists of 12 points, which expand upon our prior SDLC phases. Fig. 3 illustrates a short explanation of their phases.

DevSecOps is a significant progression in IT operations that combines the speed and agility of DevOps with strong security safeguards. Its importance in containerized architectures cannot be emphasized, as it improves security and increases cooperation and efficiency among development and operational teams. DevSecOps adoption will be critical to ensure safe, robust, and high-performing IT systems as organizations increasingly embrace cloud-native technologies and complex microservice application architectures.

2.3. Cloud security

Containers and associated orchestration software used in cloud systems provide new security challenges. Bader Alouffi et al. [13] did a literature study underlining the critical and ongoing issues in cloud computing security, emphasizing the need for further research and development.

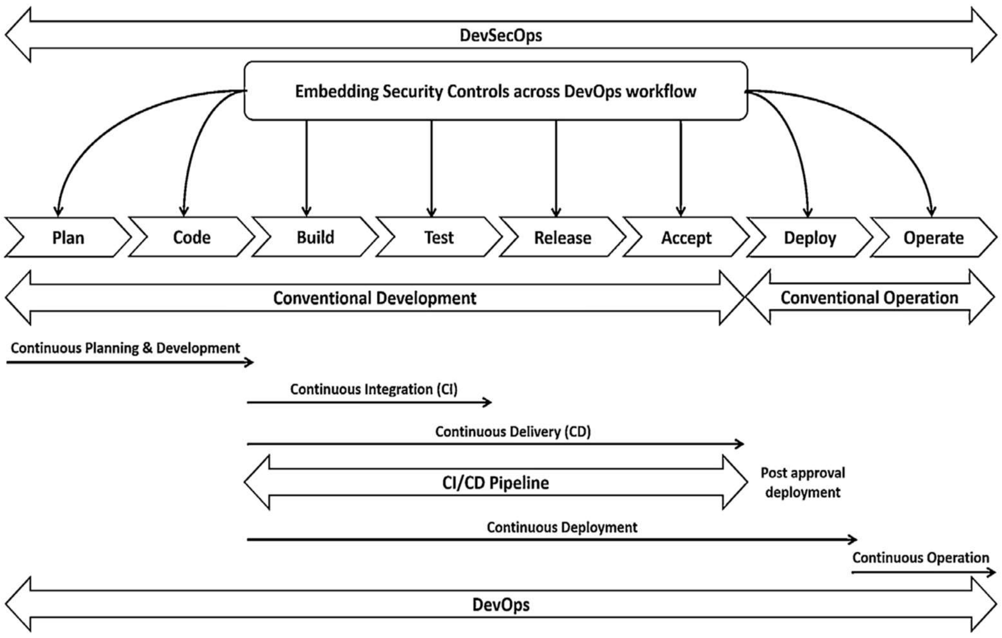

Figure 2: Common representation of DevSecOps [12]

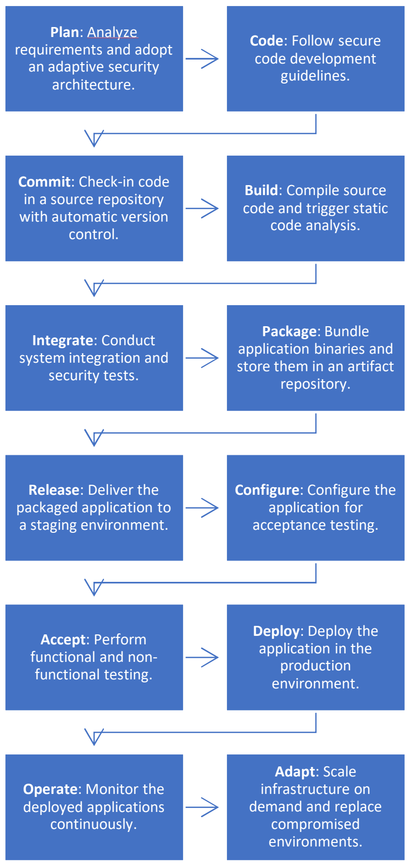
Plan: Analyze requirements and adopt an adaptive security architecture.
Code: Follow secure code development guidelines.
Commit: Check-in code in a source repository with automatic version control.
Build: Compile source code and trigger static code analysis.
Integrate: Conduct system integration and security tests.
Package: Bundle application binaries and store them in an artifact repository.
Release: Deliver the packaged application to a staging environment.
Configure: Configure the application for acceptance testing.
Accept: Perform functional and non-functional testing.
Deploy: Deploy the application in the production environment.
Operate: Monitor the deployed applications continuously.
Adapt: Scale infrastructure on demand and replace compromised environments.

Figure 3: Continuous security workflow by Kumar and Goyal [12]

Cloud-based system security becomes increasingly important as cloud computing's importance in commercial and consumer contexts grows. The analysis identified seven significant security dangers, with data manipulation and leaking being the most urgent issues. These attacks threaten the integrity and confidentiality of cloud-stored data, necessitating the development of increasingly complex security procedures [14].

Significant weaknesses exist in containerized cloud infrastructures, such as data tampering, unauthorized changes to stored data, data leaks, and the unauthorized exposure of sensitive information. The frequency of these weaknesses in the studied literature indicates that current security solutions are insufficient to properly defend cloud systems from all sorts of attackers. As a result, there is an obvious and immediate need for continued research to improve cloud security protocols.

Furthermore, the topic of data outsourcing—the transfer of control over sensitive data from users to cloud service providers (CSPs) remains a significant worry. This power transfer creates possible weaknesses, notably in data confidentiality and integrity. The evaluation emphasizes the significance of creating more robust security.

2.4. Culture

Culture is critical to effective DevSecOps workflow [15], determining how security is integrated into the software development lifecycle. The DevSecOps culture emphasizes teamwork, with development, operations, and security teams working together to integrate security into all aspects of software development. This collaborative strategy breaks down traditional silos, allowing for proactive control of security concerns.

A DevSecOps culture [16] is distinguished by continuous improvement, in which teams regularly refine their security practices in response to evolving threats, and shared responsibility ensures that all team members, not just security specialists, are accountable for upholding security standards. Communication is also essential for enabling open conversation across teams to handle security risks promptly and effectively.

DevSecOps is critical for managing the complexity of modern programs that use microservices and containers. It incorporates security into CI/CD pipelines to assure ongoing protection and promotes a shared security responsibility across development, operations, and security teams. Without an established security culture, developers will "take shortcuts" [17].

So, the keys to defining DevSecOps culture are recognising the importance of cooperation, continuous improvement, shared accountability, communication, and trust. These cultural components are vital for integrating security into the fabric of software development, ensuring that security is not an afterthought but rather a necessary component of the development process.

2.5. Regular security audits

Cybersecurity audits have evolved as a critical element of the overall cyber risk management strategy, mainly as organizations rely more on digital technologies that expose them to a broader range of cyber threats. The success of cybersecurity audits (CSA) is critical for organizations seeking to protect their digital assets' integrity, confidentiality, and availability [18].

The quality of these cybersecurity audits directly impacts the effectiveness of an organization's cyber threat defense mechanisms. Understanding and quantifying audit quality, as described in Rajgopal, Srinivasan, and Zheng's [19] study on audit quality in financial audits, can provide valuable insights into improving the robustness of cybersecurity audits.

Rajgopal et al. [19] present a comprehensive framework for assessing audit quality by examining audit problems reported in enforcement proceedings and litigation cases. Their findings highlight the importance of particular proxies for audit quality, such as restatements and the audit fee-to-total fee ratio. Using similar approaches in cybersecurity audits could help organizations detect and address flaws in their security policies more effectively.

However, organizations should realize that audits can be only one part of the cybersecurity puzzle. They must be integrated with other defensive methods to effectively combat the evolving landscape of cyber threats.

As a result, the ongoing improvement of cybersecurity audit procedures, guided by research and best practices, is crucial for organizations that strive to protect their digital resources and remain resilient to cyber threats in the long term.

2.6. Security governance and compliance

Information Security Governance (ISG) is a strategic framework that connects an organization's information security policies with its overall business objectives. It ensures essential assets are protected while protecting the organization's value and reputation [20]. This alignment is necessary to maintain organizational resilience against the evolving cyber threat.

If practical, ISG is more than just a technology requirement—it is a critical business function in which senior management has to get involved. As noted by AlGhamdi et al. [20], the research emphasizes the need for senior management support and dedication in driving the successful deployment of ISG frameworks. Information security governance at the highest organizational levels guarantees that security measures are not seen as separate IT problems but integrated into strategic decision-making, reinforcing the organization's entire risk management approach.

ISG also ensures compliance with applicable laws, regulations, and industry best practices. Compliance will ensure that security practices within an organization have been attested to regulatory standards and follow well-known best practices. AlGhamdi et al. [20] emphasize that compliance is critical for legal protection and improving the organization's reputation and credibility.

Organizations can reduce the risks of noncompliance, such as legal penalties and reputational damage, by adhering to regulatory frameworks and ensuring that security controls are consistently deployed and reviewed [5].

2.7. Secure decommissioning

Securely decommissioning IT assets, particularly storage devices containing sensitive information, is crucial to data security. Unfortunately, it is often overlooked and can pose significant security risks and legal obligations.

As the literature further shows, one of the key challenges in IT asset disposal (ITAD) is ensuring that all data stored on devices is irretrievably erased before the hardware is repurposed, sold, or disposed of. The study by Debnath et al. [21] emphasizes the potential threats posed by improper decommissioning of IT assets, particularly among middle card players, such as small- and medium-sized enterprises (SMEs), institutions, and individuals. These entities often lack the resources or expertise to implement stringent data sanitization processes, making them particularly vulnerable to data breaches when their IT assets enter the e-waste supply chain [21].

In the e-waste supply chain, improper decommissioning can escalate the risk even further. IT assets are difficult to track, and unauthorized parties may be able to retrieve sensitive information due to a lack of proper disposal methods, including the physical components that can retain recoverable data even after standard deletion procedures.

Integrating secure decommissioning practices into Information Security Management Systems (ISMS), such as ISO/IEC 27001, is critical for managing information security risks, including those related to IT asset end-of-life, and preventing unauthorized data recovery.

Secure decommissioning is a governance challenge, not just a technical issue. Organizations should establish clear policies for ITAD, including certified data destruction services and certificates, to ensure sensitive data is securely handled and reduce legal risks associated with data breaches.

Secure decommissioning is an essential part of IT asset management. It ensures an organization's security from breaches and unauthorized access. Compliance with set standards and a strong policy on ITAD can ensure that data security prevails throughout the lifecycle of IT assets.

2.8. Continuous innovation in security

Innovation within DevOps is continuous and pervasive throughout the software development and operation lifecycle. Continuous innovation in DevOps is the ongoing ability to respond to new requirements and market changes while keeping the software up-to-date, efficient, and competitive [22].

DevOps practices, including CI/CD, automation, and cross-functional collaboration, help organizations maintain agility and innovate quickly to respond to changing market conditions [23].

3. Existing classifications

There are several kinds of literature on continuous security frameworks. The most important among them is presented by Xiaofan Zhao et al.'s study [24]. Their Challenge-Practice-Tool-Metric (CPTM) approach offers a complete framework for successfully integrating security into DevOps operations. This model is developed based on a Multi-vocal Literature Review (MLR) and illustrates the correlations between challenges, processes, tools, and metrics in the DevSecOps lifecycle.

Key obstacles to adopting DevSecOps identified using the **CPTM model** are organizational resistance, integration complexities, and the need for perpetual compliance. These struggles are further classified under organizational, procedural, technological, and business-related difficulties, providing a solid understanding of what stands in the way of successfully deploying DevSecOps.

The model emphasizes integrating these technologies into current processes to improve efficiency and security. It provides a variety of commercial and open-source solutions that enable these techniques. This guarantees that the right instruments are applied to successfully handle certain security requirements.

The **ADOC model** [12] introduces a framework that integrates development, security, and operational activities to ensure that security practices are incorporated at each stage of the DevOps pipeline. OSS and cloud technologies enable this model, which incorporates six dimensions, nine guidelines, a twelve-stage process, and seven practice areas.

The ADOC model presented in Fig. 3 provides a way forward for organizations intending to apply DevSecOps principles through OSS when deploying in the cloud. This ensures built-in security at every stage of development and is applied through automation, making it far more achievable to deliver securely and cost-effectively developed high-quality software. The twelve-step process ensures that security does not become an afterthought but is considered an integral part of the DevOps lifecycle, thus raising the organization's overall security posture.

Another great source of categorization information is the GitHub repository by Sottlmarek, a popular and well-organized resource in the **DevSecOps community library** [25] with over 5,300 stars. It offers a vast library of tools and approaches for integrating security into the DevOps lifecycle, primarily focused on cloud cybersecurity and DevSecOps best practices. It categorizes tools into pre-commit time, secret management, OSS and dependency management, supply chain security, SAST, DAST, continuous deployment security, Kubernetes, and container security.

The OWASP® Foundation supports software security through community-led projects, global chapters, and conferences. Agile frameworks and DevOps practices drive the software development industry, which often fails to integrate security concerns during deployment. Standard safety measures are sometimes ignored in continuous integration settings, leading to insecure Docker registries and the potential theft of a company's entire source code. The foundation aims to address these challenges by promoting open-source software projects, ensuring security, and supporting collaborative conferences. They have created the DevSecOps Maturity Model for better security planning.

The DevSecOps Maturity Model, as shown in Fig. 4, demonstrates the security controls applied when implementing DevOps practices and how they can be benchmarked. DevOps practices can also enhance security by evaluating each part of a Docker image, such as application and operating system libraries, for known vulnerabilities. Attackers are intelligent and creative, constantly evolving with new technologies and goals. Guided by the visionary DevSecOps Maturity Model, relevant ideas and actions are being implemented to mitigate threats [26].

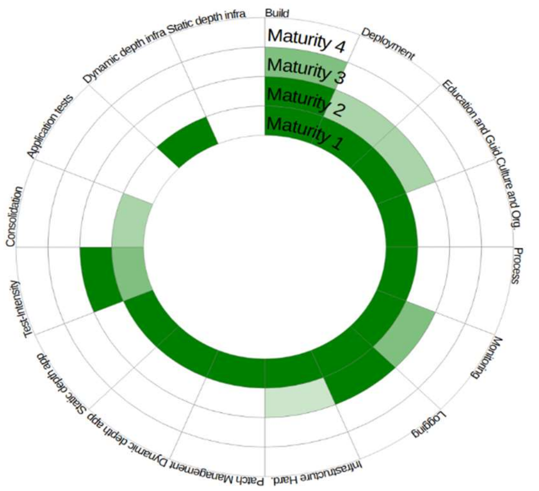

Figure 4: DevSecOps Maturity Model example of Identification of the degree of the implementation [26]

As we can see from the information provided, "white" and "grey" researchers are not adhering to a single stagnant classification for the Software Development Lifecycle in securing containerized and cloud-native environments. This suggests that diverse perspectives and orientations are being pursued in this research, leading to a broader model. It also indicates that the process of integrating DevSecOps into software development lifecycles is complex and multifaceted.

3.1. Comparative analysis of models

In this section, we conduct a comparative analysis of four prominent models: the Continuous Planning and Testing Model (CPTM), the DevSecOps Maturity Model, the Application Delivery and Operations Control (ADOC) model, and a generic DevSecOps library approach.

Table 1
Comparison of existing models

CPTM model [24]	The DevSecOps Maturity Model [26]	ADOC [12]	DevSecOps library [25]
Plan	Requirements Gathering	Plan	Plan
Create	Design	Code	Code
Verify	Development	Commit	Build
Preprod	Testing	Build	Test
Release	Deployment	Integrate	Release
Prevent	Maintenance	Package	Deploy
Detect		Release	Operate
Respond		Configure	Monitor
Predict		Accept	
Adapt		Deploy	
		Operate	
		Adapt	

Continuous Planning and Testing Model (CPTM) illustrates a cyclical approach to security that underlies the SDLC. It begins with the ***Planning phase***, where security requirements are specified, followed by the ***creation phase***, which focuses on the design of secure systems. The model emphasizes continuous ***Verification*** during the ***Implementation phase***. In the ***Testing phase***, the need to identify security vulnerabilities is prioritized before deployment, and the ***Release phase*** highlights secure release practices. CPTM presents a continuous loop of ***Prevent, Detect, Respond, Predict***, and ***Adapt*** activities during the ***Maintenance phase***, underscoring the importance of ongoing vigilance and adaptability to maintain secure operations.

DevSecOps Maturity Model provides a structured approach to scaling security practices in the SDLC. It starts with ***requirements gathering*** in the ***planning phase***, where security needs are deeply embedded in project planning. In the ***Design phase***, security is integrated into system architecture, making it a fundamental part of the design process. ***The Implementation phase*** focuses on secure Development, emphasizing secure coding and regular security assessments. Rigorous ***Testing*** ensures that security testing is an integral part of quality assurance. ***The Deployment phase*** incorporates security into the deployment pipeline, making security checks a continuous part of software releases. Finally, ***Maintenance*** ensures that security remains a top priority throughout the software's lifecycle.

Application Delivery and Operations Control (ADOC) introduces a unique approach concentrating on delivery and operational management. The ***Planning phase*** aligns with strategic security planning, while the ***Code phase*** emphasizes secure coding practices as part of the design. During the ***Implementation phase***, ***Commit*** activities ensure that code commits are secure and reliable. ***Build*** and ***Integrate*** activities in the ***Testing*** and ***Deployment phases*** to ensure that security is integrated into the continuous integration/continuous deployment (CI/CD) pipeline. The ***Maintenance phase*** is comprehensive, covering ***Package***, ***Release***, ***Configure***, ***Accept***, ***Deploy***, ***Operate***, and ***Adapt*** activities, ensuring security is maintained and adapted to changing environments and threats.

The **DevSecOps Library** takes a broader approach to DevSecOps, reflecting its practices in a more generalized way. It closely aligns with the traditional Software Development Life Cycle (SDLC) stages while strongly focusing on integrating security measures throughout each phase. ***Planning*** involves the identification and integration of security requirements. ***Code*** reflects secure coding practices, while ***Build*** focuses on embedding security into the build process. The ***Test phase*** emphasizes thorough security testing, ensuring vulnerabilities are identified and addressed before deployment. ***Release*** in this approach involves the secure deployment of the software. ***Maintenance*** includes ***deploying, operating, and monitoring*** activities to ensure continuous security throughout the software's operational life.

The comparative examination (Table 1) of these models demonstrates that, while each framework has distinct strengths and emphasis areas, they all strive for the same goal: seamless security integration into the SDLC. By comparing these models, we obtain significant insights into how security might be systematically integrated into software development processes, improving the security posture of modern software systems. As the threat landscape evolves, adopting and enhancing these models will be critical for organizations seeking to maintain strong security across the software lifecycle.

3.2. Expanding existing models

Following our research, we identified additional areas for improvement in existing models and their classification. Extra phases can be introduced to address the problems and practices discussed earlier. One key recommendation is to emphasize the importance of **continuous education**, ensuring all team members are regularly trained on security best practices, tools, and emerging threats. This helps a security-conscious culture within the organization, where every stakeholder plays a role in maintaining security.
Incorporating **disaster recovery** and business continuity measures is also essential. Developing and testing plans that allow business operations to recover swiftly after a significant incident or system failure ensures resilience.

Additionally, **regular security audits** should be conducted to assess the effectiveness of security measures.

These audits should be automated and integrated into the CI/CD pipeline whenever possible to maintain alignment with current standards and compliance requirements.

As applications grow, security measures must **scale** accordingly. Adapting security strategies to the increasing complexity of container orchestration and cloud environments is vital.

Moreover, **implementing governance frameworks** helps ensure that all security practices comply with regulatory and organizational standards. These policies and procedures are crucial for guiding applications' secure development, deployment, and maintenance.

Secure decommissioning processes are equally important when applications or components end their lifecycle. This involves securely removing data, dismantling infrastructure, and ensuring no residual vulnerabilities are left behind.

Lastly, encouraging **continuous innovation** in security practices and technologies keeps the organization ahead of emerging threats. Adopting new tools, methodologies, and approaches helps to navigate the ever-evolving challenges within the DevSecOps landscape.

Our extended model (Table 2) contains 20 elements, which cover the whole range of actions required for the safe development, deployment, and operation of contemporary software systems. By including extra stages such as monitoring, reaction, recovery, auditing, and education, this model provides a complete framework that handles the technical components of security and the cultural and procedural factors required to maintain a strong security posture.

Table 2
Proposed extended model

Phase	Description
Educate	Continuously train and educate team members.
Plan	Strategic planning, defining project objectives, gathering requirements, and identifying security risks.
Govern	Establish security governance frameworks and ensure compliance with regulatory standards.
Code	Implement secure coding practices.
Commit	Use secure version control practices, protecting code commits.
Build	Integrate automated security testing into the build process.
Integrate	Ensure secure integration of components with automated testing and validation.
Package	Package the application securely.
Configure	Manage configurations securely, applying best practices to ensure consistency and security.
Release	Conduct final security checks and validations before releasing the application to production.
Deploy	Automate deployment with integrated security checks, ensuring secure, validated code reaches production.
Operate	Implement continuous monitoring and real-time security operations.
Monitor	Automated tools are used for continuous security monitoring and anomaly detection [27].
Respond	Establish incident response protocols.
Audit	Perform regular security audits to assess the effectiveness of security measures and ensure ongoing compliance.
Accept	Conduct security acceptance testing.
Scale	Adapt security strategies to accommodate growth and increased complexity.
Adapt	Regularly review and update security practices.
Innovate	Adopting new technologies and methodologies.
Decommission	Securely retire applications or components.

This expanded model is especially well-suited for addressing the complexities of containerized, cloud-native environments, ensuring that security is integrated throughout the entire software lifecycle. It can also be used to develop and automate deployments of enterprise security subsystems, as in our previous research [28].

The model addresses all phases of the SDLC and integrates security considerations at every stage, ensuring a comprehensive approach to secure software development in modern, cloud-native environments.

4. Conclusions

This study proposes a complete DevSecOps model designed to solve the security problems of containerized and cloud-native environments. By incorporating security practices throughout the SDLC, the model provides a solid foundation for organizations to improve their security posture. The suggested model incorporates essential aspects such as security governance, disaster recovery planning, frequent audits, and secure decommissioning to ensure that security is an ongoing and integrated part of the development process.

The comparison with existing frameworks shows that the expanded model fills gaps in current practices and provides a scalable solution that syncs with the dynamic nature of modern IT environments. The model's emphasis on continual innovation and adaptation helps organizations stay ahead of emerging threats and changing security requirements.

Empirical validation of the effectiveness and scalability of the suggested DevSecOps approach through implementation in actual applications across several sectors. Automated security tool integration is another possible area to explore how these tools might be integrated into the suggested paradigm, particularly inside the CI/CD pipeline. Potential pathways for further study might include examining organizational and cultural obstacles to adopting DevSecOps methods and how training initiatives and strategic change management can help remove these barriers.

As technology advances, future studies should assess the model's adaptability to new frameworks, such as serverless computing and AI-driven development processes.

Another approach is to examine how the suggested model affects efficiency and performance, especially in terms of development teams' resource allocation and time to market.

Lastly, future research must focus on matching the model with different regulatory frameworks and investigating the possibility of automating compliance checks inside DevSecOps procedures.

Once addressed, these research topics will enrich and modify the proposed model to suit the changing needs of the software development industry and, hence, go a long way toward contributing to the development of secure and resilient IT infrastructures.

References

[1] D. Berestov, et al., Analysis of Features and Prospects of Application of Dynamic Iterative Assessment of Information Security Risks, in: Cybersecurity Providing in Information and Telecommun. Systems, CPITS, vol. 2923 (2021) 329–335.

[2] S. Shevchenko, et al., Information Security Risk Management using Cognitive Modeling, in: Cybersecurity Providing in Information and Telecommun. Systems II, CPITS-II, vol. 3550 (2023) 297–305.

[3] B. Kaur, et al., An Analysis of Security Vulnerabilities in Container Images for Scientific Data Analysis, GigaScience, 10(6) (2021). doi: 10.1093/gigascience/giab025.

[4] F. Khan, et al., Data Breach Management: An Integrated Risk Model, Inf. Manag. 58(1) (2021) 103392. doi: 10.1016/j.im.2020.103392.

[5] R. N. Rajapakse, et al., Challenges and Solutions when Adopting DevSecOps: A Systematic Review, J. Inf. Software Technol. 141 (2022) 106700. doi: 10.1016/j.infsof.2021.106700.

[6] N. Dwivedi, D. Katiyar, G. Goel, A Comparative Study of Various Software Development Life Cycle Models, Int. J. Res. Eng. Sci. Manag. 5(3) (2022). 141–144.

[7] B. Acharya, P. K. Sahu, Software Development Life Cycle Models: A Review Paper, Int. J. Adv. Res. Eng. Technol. 11 (2020). 169–176. doi: 10.34218/ijaret.11.12.2020.019.

[8] S. Pargaonkar, A Comprehensive Research Analysis of Software Development Life Cycle (SDLC) Agile & Waterfall Model Advantages, Disadvantages, and Application Suitability in Software Quality Engineering, Int. J. Sci. Res. Publ. 13 (2023) 120–124. doi: 10.29322/ijsrp.13.08.2023.p14015.

[9] O. E. Olorunshola, F. N. Ogwueleka, Review of System Development Life Cycle (SDLC) Models for Effective Application Delivery, Information and Communication Technology for Competitive Strategies (ICTCS 2020), LNNS 191 (2021) 281–289. doi: 10.1007/978-981-16-0739-4_28.

[10] R. A. Khan, et al., Systematic Literature Review on Security Risks and Its Practices in Secure Software Development, IEEE Access 10 (2022) 5456–5481. doi: 10.1109/access.2022.3140181.

[11] Solutions — DevSecOps — Addressing Security Challenges in a Fast Evolving Landscape White Paper (2022). URL: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/devsecops-addressing-security-challenges.html

[12] R. Kumar, R. Goyal, Modeling Continuous Security: A Conceptual Model for Automated DevSecOps using Open-Source Software over Cloud, Comput. Secur. 97 (2020) 101967. doi: 10.1016/j.cose.2020.101967.

[13] B. Alouffi, et al., A Systematic Literature Review on Cloud Computing Security: Threats and Mitigation Strategies, IEEE Access 9 (2021) 57792–57807. doi: 10.1109/access.2021.3073203.

[14] P. Anakhov, et al., Protecting Objects of Critical Information Infrastructure from Wartime Cyber Attacks by Decentralizing the Telecommunications Network, in: Cybersecurity Providing in Information and Telecommun. Systems, vol. 3050 (2023) 240-245.

[15] N. Tomas, J. Li, H. Huang, An Empirical Study on Culture, Automation, Measurement, and Sharing of DevSecOps, in: International Conference on Cyber Security and Protection of Digital Services, UK (2019) 1–8. doi: 10.1109/cybersecpods.2019.8884935.

[16] M. Sánchez-Gordón, R. Colomo-Palacios, Security as Culture: A Systematic Literature Review of DevSecOps, in: IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20). Association for Computing Machinery (2020) 266–269. doi: 10.1145/3387940.3392233.

[17] S. Sultan, I. Ahmad, T. Dimitriou, Container Security: Issues, Challenges, and the Road Ahead, IEEE Access 7 (2019). 52976–52996. doi: 10.1109/access.2019.2911732.

[18] H. Hulak, et al., Dynamic model of guarantee capacity and cyber security management in the critical automated systems, in: 2nd International Conference on Conflict Management in Global Information Networks, vol. 3530 (2022) 102-111.

[19] S. Rajgopal, S. Srinivasan, X. Zheng, Measuring audit quality, Review of Accounting Studies 26 (2021). 559–619. doi: 10.1007/s11142-020-09570-9.

[20] S. AlGhamdi, K. T. Win, E. Vlahu-Gjorgievska, Information Security Governance Challenges and Critical Success Factors: Systematic Review, Comput. Secur. 99 (2020). 102030. doi: 10.1016/j.cose.2020.102030.

[21] B. Debnath, et al., An Analysis of Data Security and Potential Threat from IT Assets for Middle Card Players, Institutions and Individuals, Sustainable Waste Management: Policies and Case Studies (2019) 403–419. doi: 10.1007/978-981-13-7071-7_36.

[22] A. Wiedemann, et al., Implementing the Planning Process within DevOps Teams to Achieve Continuous Innovation, in: 52nd Hawaii International Conference on System Sciences (2019) 7017–7026. doi: 10.24251/hicss.2019.841.

[23] G. Auth, R. Alt, C. Kögler, Continuous Innovation with DevOps: IT Management in the Age of Digitalization and Software-defined Business, Springer Cham (2021). doi: 10.1007/978-3-030-72705-5.

[24] X. Zhao, T. Clear, R. Lal, Identifying the Primary Dimensions of DevSecOps: A Multi-Vocal Literature Review, J. Syst. Software 214 (2024) 112063. doi: 10.1016/j.jss.2024.112063.

[25] GitHub. sottlmarek/DevSecOps: Ultimate DevSecOps Library. URL: https://github.com/sottlmarek/DevSecOps

[26] OWASP Devsecops Maturity Model | OWASP Foundation. URL: https://owasp.org/www-project-devsecops-maturity-model/

[27] O. V. Talaver, T. A. Vakaliuk, Telemetry to Solve Dynamic Analysis of a Distributed System, J. Edge Comput. 3 (2024) 87–109. doi: 10.55056/jec.728.

[28] B. Leshchenko, et al., Model of a Subsystem for Securing E-Mail Against Loss using Mail Transport Agents based on Containerized Environments, in: Cybersecurity Providing in Information and Telecommunication Systems II co-located with International Conference on Problems of Infocommunications. Science and Technology (PICST 2023), vol. 3550 (2023) 14–28.