Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

Cybersecurity in DevOps Environments: A Systematic Literature Review

ROBERTO CARLOS BAUTISTA RAMOS[1], SANG GUUN YOO[1]
[1]Departamento de Informática y Ciencias de la Computación, Escuela Politécnica Nacional, Quito, Ecuador
Corresponding author: Sang Guun Yoo (e-mail: sang.yoo@epn.edu.ec).

ABSTRACT This systematic literature review provides a comprehensive analysis of the most critical cybersecurity challenges in DevOps environments. Through a rigorous examination of 62 peer-reviewed articles published between 2016 and 2025, we identified recurring threats, active attack vectors, structural vulnerabilities, mitigation strategies, and their technical impact on system performance and operational resilience. The analysis revealed that the most significant threats are related to uncontrolled automation, exposure of sensitive secrets in CI/CD pipelines, lack of mutual authentication between distributed services, supply chain attacks, and the use of unauthorized tools (Shadow IT). These threats simultaneously compromise core security principles, including integrity, confidentiality, and traceability. The most frequent attack vectors include code injection in CI/CD pipelines, unrestricted access to public repositories, remote execution via default configurations, and lateral movement in flat architectures. We identified 27 recurrent vulnerabilities throughout the DevOps lifecycle. The most critical include the absence of automated security testing, poor management of secrets, and reliance on unverified third-party components. More than 30 technical and organizational countermeasures were documented, such as SAST/DAST/IAST scans, infrastructure-as-code validation, secure credential storage via vaults, and integrated practices like DevSecOps and compliance-as-code. When properly implemented, these strategies do not degrade system performance and may even enhance resilience and stability. Nonetheless, a lack of comparative empirical validation in most reviewed studies limits the generalizability of proposed solutions. These findings establish a foundation for future research in emerging domains, such as the Internet of Things, where continuous, adaptive, and verifiable security is paramount for automated and dynamic environments.

INDEX TERMS DevOps, Cybersecurity, Threats, Vulnerabilities, Attack Vectors, Mitigation, Systematic Literature Review.

I. INTRODUCTION

Current technology enables organizations to carry out digital transformation processes, which in turn significantly affects the development and operation of software systems. Within this context, the DevOps paradigm has emerged as an optimal solution to the challenge of accelerating value delivery by improving software quality as well as collaboration between Development (Dev) and Operations (Ops) teams. The principles of DevOps include automation, Continuous Integration/Continuous Delivery (CI/CD), Infrastructure as Code (IaC), continuous monitoring, and cyclic interactions, creating an adaptive system in a highly dynamic environment. However, this operational flexibility has led to increased complexity in the attack surface, which has drawn greater attention to cybersecurity issues in these highly orchestrated and automated environments [1] [2]. DevOps environments pose a challenging problem in the development of efficient and sustainable security mechanisms due to the speed at which systems are built, tested, and deployed. Traditional security, which was applied reactively or at late stages of the software lifecycle, is insufficient in environments where deployments occur multiple times per day. Therefore, security must be transformed into an integrated discipline that is incorporated from the early stages of development and maintained throughout system operation, evolving continuously. Based on this definition, DevSecOps emerges as the new branch of DevOps that adds security as a recurring element in the software pipeline for its constant automation and application throughout the software lifecycle [3] [4]. The use of methodologies such as shift-left security allows development teams to integrate security controls from the early stages of development and greatly reduce the probability of exploiting flaws in advanced production stages. This philosophy has become highly relevant in continuous delivery environments, where the prevention of vulnerabilities in the code at early stages becomes a priority to maintain system resilience. In this way, the formal introduction of cybersecurity principles in DevOps contexts was established in 2017, when Gartner formalized the term DevSecOps in its report *Innovation Insight for DevSecOps*, providing a formally accepted framework [5]. The effective adoption of the DevSecOps approach involves combining several practices, such as static code analysis, dynamic application scanning, third-party dependency verification, secret management, policy as code, integration of access controls in automation tools, and validation of configurations in ephemeral infrastructure environments such as containers or serverless functions. These practices not only require advanced technical tools but also demand a cultural shift within the organization, where security becomes a shared responsibility rather than an isolated or subsequent role. However, this integration brings significant challenges, such as resistance to change among development teams, lack of training in secure practices, the complexity of automating controls without affecting productivity, and the need to manage a high volume of security alerts with a low false positive rate [6]. On the other hand, the arrival of new technologies such as Kubernetes, Docker, Terraform, and CI/CD platforms like Jenkins, GitLab CI/CD, GitHub Actions, or CircleCI has opened a world of possibilities for automation. However, it has also introduced new attack vectors. Security threats in DevOps are not limited to source code errors; they also include misconfigured infrastructure components, compromised supply chains, third-party libraries with known vulnerabilities, and accidental exposure of API keys, among other critical risks. In fact, recent incidents such as the SolarWinds supply chain attack and the Codecov security breach have demonstrated how malicious actors can infiltrate DevOps environments using legitimate tools, affecting not only a single organization but also multiple customers interconnected through the digital value chain [7] [8]. In this context, scientific research has produced a growing number of works addressing various facets of security in DevOps. These range from case studies on the implementation of DevSecOps practices in real organizations to proposals for maturity models, frameworks for secure integration, metrics to evaluate pipeline protection levels, and automated scanning tools for containers or source code. However, this body of knowledge is scattered, fragmented, and presents varying levels of theoretical depth and empirical validity. This study offers a systematic literature review aimed at analyzing security challenges in DevOps environments, identifying the strategies employed to mitigate threats and vulnerabilities, and examining how these strategies influence system performance. To ensure the methodological rigor of the study, the approach proposed by Kitchenham et al. is adopted, which is widely recognized in Software Engineering and provides clear guidelines for protocol formulation, primary source identification, application of inclusion and exclusion criteria, as well as extraction and synthesis of relevant data [9]. This study provides a comprehensive view of security in DevOps environments, addressing the main threats and attack vectors, along with the documented methodological approaches to mitigate their impact on security and operational efficiency. It also examines inherent vulnerabilities that can be exploited by malicious actors, analyzing effective corrective measures to reduce risks. Finally, it explores how mitigation strategies influence system performance, identifying the most efficient ones in applied scenarios. Through a systematic review, this work aims to provide a key resource for researchers and professionals, offering technical and scientific evidence to strengthen security in modern software development.

II. METHODOLOGY

This research was carried out through a Systematic Literature Review (SLR), strictly following the methodological guidelines established by Kitchenham et al. [9]. This methodology provides a structured and replicable framework to identify, assess, and synthesize the scientific evidence available on a specific topic, minimizing potential biases and strengthening the validity of the findings. The application of this technique is particularly relevant in rapidly evolving technological contexts such as cybersecurity in DevOps environments, where knowledge is scattered across multiple sources and disciplines. The review process was organized into four fundamental stages, illustrated in Figure 1 and described in detail below: (1) formulation of the research questions, aimed at defining the analytical focus of the study; (2) systematic identification and selection of relevant primary studies using reproducible search strategies in high-impact scientific databases; (3) critical appraisal of the methodological quality of the selected studies through explicit inclusion, exclusion, and quality criteria; and (4) synthesis and interpretation of the results, in order to extract patterns, trends, challenges, and relevant strategies from the analyzed corpus. This systematic approach ensures a deep, objective, and methodologically sound analysis, providing a comprehensive view of the state of the art and emerging practices related to the integration of security into DevOps development cycles. The main objective of this review was to collect, filter, and synthesize primary studies relevant to the field of cybersecurity in DevOps environments, considering as primary study any scientific publication that reports original findings, empirical results, or theoretical contributions based on evidence directly obtained by the authors. The SLR, as a consolidated technique within the evidence-based research paradigm, is designed to answer specific research questions through a rigorous, transparent, and reproducible process, based on the prior definition of a structured review protocol. This protocol encompasses aspects such as the design of search strategies, definition of Boolean strings, selection of academic sources, and cross-validation of the results obtained, thus ensuring the traceability and reliability of the entire research process.

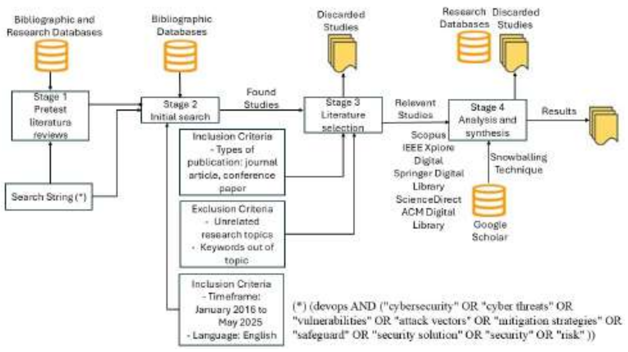
Bibliographic and Research Databases
Bibliographic Databases
Discarded Studies
Research Databases
Discarded Studies
Stage 1 Pretest literature reviews
Stage 2 Initial search
Found Studies
Stage 3 Literature selection
Relevent Studies
Stage 4 Analysis and synthesis
Results
Search String (*)
Inclusion Criteria - Types of publication: journal article, conference paper
Exclusion Criteria - Unrelated research topics - Keywords out of topic
Inclusion Criteria - Timeframe: January 2016 to May 2025 - Language: English
Scopus IEEE Xplore Digital Springer Digital Library ScienceDirect ACM Digital Library
Snowballing Technique
Google Scholar
(*) (devops AND ("cybersecurity" OR "cyber threats" OR "vulnerabilities" OR "attack vectors" OR "mitigation strategies" OR "safeguard" OR "security solution" OR "security" OR "risk"))

FIGURE 1. **Stages of the systematic literature review process.**

A. PHASE 1: INITIAL SEARCH

This phase focused on the methodological and operational planning of the systematic literature review through the structuring of a rigorous and reproducible research protocol. The process was divided into two sub-stages: (a) protocol design and (b) protocol execution. **a)** Research Protocol Design A set of Research Questions (RQs) was defined to explore in a structured manner the challenges, strategies, practices, and emerging trends surrounding security in DevOps environments. The research questions, along with their motivations, are detailed in Table 1. In parallel, a structured search string was constructed (Table 2) using Boolean operators, controlled vocabulary, and relevant synonyms, with the aim of maximizing both recall and precision in the retrieval of pertinent studies. The selection of bibliographic sources focused on highly relevant databases in the field of software engineering and cybersecurity, including: Scopus, IEEE Xplore Digital Library, SpringerLink, ScienceDirect, and ACM Digital Library. The time frame for analysis was set from 2016 to 2025, in order to capture the contemporary evolution of the DevSecOps approach, framed by two key milestones:

- In 2016, Julien Vehent published the book *Securing DevOps*, a pioneering work that laid the technical foundations for the systematic incorporation of security practices into automated development pipelines [10].
- In 2017, Gartner formally included the concept of DevSecOps in its report *Innovation Insight for DevSecOps*, recognizing it as a strategic trend in modern risk management [5].

Additionally, the protocol established a language criterion requiring that only publications written in English be included, in order to ensure an international, technical, and specialized focus aligned with the standards of high-impact scientific literature. **b)** Protocol Execution Once the protocol was defined, systematic searches were conducted in the selected databases, applying the predefined search string in the fields of title, abstract, and keywords. The result of this phase was the construction of a preliminary document corpus composed of scientific publications directly related to the subject of study: cybersecurity in DevOps environments. This initial set of studies was subsequently subjected to filtering, quality assessment, and data extraction processes, corresponding to the subsequent stages of the methodological framework.

B. PHASE 2: BIBLIOGRAPHY SELECTION

This stage included the following activity: **a)** Conducting a detailed analysis of the titles, abstracts, and keywords of the identified studies, applying inclusion and exclusion criteria. The inclusion criteria were defined as the exclusive acceptance of articles published in peer-reviewed journals and papers presented at relevant academic conferences. On the other hand, the exclusion criteria consisted of discarding studies whose research approaches were not related to the main topic or that included keywords outside the area of interest."

C. PHASE 3: ANALYSIS AND SYNTHESIS

This stage was structured as follows: **a)** The full content of the studies was accessed through highly regarded academic databases, such as Scopus, IEEE Xplore Digital Library, Springer Digital Library, ScienceDirect, and ACM Digital Library. These sources were selected due to their relevance in publishing high-quality scientific research and their coverage in areas related to technology, software development, and cybersecurity. Additionally, they ensure access to peer-reviewed articles and a wide range of specialized resources, thereby guaranteeing the scientific validity of the document corpus. **b)** The complete content of the studies was assessed based on predefined inclusion criteria. Only studies whose focus was semantically related to the DevOps paradigm and that addressed specific aspects of cybersecurity, threats, vulnerabilities, and mitigation strategies were considered relevant. **c)** To enrich the final document corpus, the snowballing technique was applied. This process involved using Google Scholar (GS) to identify studies that cited the initially selected documents, as well as conducting an exhaustive review of their references, following the methodology described by [11]. This technique allowed for the identification of relevant connections and expanded the analytical scope. **d)** A critical and thorough analysis was carried out both within each selected study and across the studies in the final corpus, with the aim of identifying recurring patterns, knowledge gaps, and complementary approaches. **e)** Finally, the research questions stated in the protocol were addressed by integrating findings obtained during the analysis and synthesis stage, ensuring a rigorous and evidence-based approach.

III. RESULTS AND ANALYSIS

This section presents the results obtained throughout the search process, including a detailed description of the final composition of the document corpus. This corpus consists of 62 relevant studies that were selected and classified according to predefined criteria, covering various topics and methodologies. The key findings that emerged from the reviewed studies are outlined below, along with prevailing patterns, trends, and research areas, with a particular focus on security practices and tools applied in DevOps environments.

TABLE 1. Research Questions and Motivations

Research Question	Motivation
RQ1. What are the main cyber threats and attack vectors impacting DevOps environments, and which methodological approaches have been documented to mitigate their impact on system security and operational efficiency?	The motivation behind this question lies in the need for a comprehensive characterization of cyber threats and attack vectors affecting DevOps environments, given their highly automated, ephemeral architecture geared toward continuous integration and delivery. The goal is to systematically analyze adversarial mechanisms that exploit attack surfaces introduced by CI/CD pipelines, containers, infrastructure as code, and orchestration tools, as well as to compile effective methodological approaches.
RQ2. What are the main inherent vulnerabilities in DevOps environments that facilitate exploitation by malicious actors, and what corrective measures have been implemented to mitigate their impact on system security?	This question arises from the need to identify systemic, technical, and procedural vulnerabilities inherent to DevOps practices which, due to their focus on agility and automation, tend to introduce non-trivial exposure conditions. The aim is to establish a taxonomy of exploitable weaknesses, including insecure configurations, secret management failures, and dependency on vulnerable software components, as well as to examine empirically validated corrective measures, such as the integration of automated controls, security as code, and early verification mechanisms via shift-left approaches.
RQ3. How do threat and attack mitigation strategies in DevOps environments influence system performance, and which have proven to be the most effective in applied scenarios?	This question addresses the inherent tension between implementing cybersecurity measures and maintaining optimal levels of performance, scalability, and operational agility in DevOps pipelines. It seeks to examine, based on empirical evidence and applied studies, the quantitative and qualitative impact of mitigation strategies on productivity, development throughput, deployment times, and environment latency. It also aims to identify practices that, due to their effectiveness and low computational cost, have been validated as optimal trade-offs between protection and efficiency.

TABLE 2. Search Strings Applied in Each Database

Database	Search String	Articles Found
Scopus	TITLE-ABS-KEY ((devops AND ("cybersecurity" OR "cyber threats" OR "vulnerabilities" OR "attack vectors" OR "mitigation strategies" OR "safeguard" OR "security solution" OR "security" OR "risk"))) AND PUBYEAR > 2015 AND PUBYEAR < 2026 AND (LIMIT-TO (DOCTYPE, "ar")) AND (LIMIT-TO (LANGUAGE, "English"))	131
IEEE Xplore Digital Library	(devops AND ("cybersecurity" OR "cyber threats" OR "vulnerabilities" OR "attack vectors" OR "mitigation strategies" OR "safeguard" OR "security solution" OR "security" OR "risk"))	25
Springer Digital Library	(devops AND ("cybersecurity" OR "cyber threats" OR "vulnerabilities" OR "attack vectors" OR "mitigation strategies" OR "safeguard" OR "security solution" OR "security" OR "risk"))	479
ScienceDirect	(devops AND ("cybersecurity" OR "cyber threats" OR "vulnerabilities" OR "attack vectors" OR "mitigation strategies" OR "safeguard" OR "security solution" OR "security" OR "risk"))	83
ACM Digital Library	[All: devops] AND [[All: "cybersecurity"] OR [All: "cyber threats"] OR [All: "vulnerabilities"] OR [All: "attack vectors"] OR [All: "mitigation strategies"] OR [All: "safeguard"] OR [All: "security solution"] OR [All: "security"] OR [All: "risk"]] AND [E-Publication Date: (01/01/2016 TO 12/31/2025)]	158

A. INITIAL SEARCH

Searches were conducted in the bibliographic databases specified in the research protocol, applying the search string defined in Table 2. This process resulted in the identification of a total of 876 potentially relevant studies for the objectives of this research.

B. BIBLIOGRAPHIC SELECTION

In Stage 3, 657 out of the 876 studies analyzed were filtered out. These were excluded due to duplication across sources, lack of thematic alignment, absence of explicit references to DevOps environments in titles or abstracts, and the use of non-pertinent keywords. As a result, 219 articles were selected for further consideration.

C. ANALYSIS AND SYNTHESIS

To ensure the validity and reliability of the study selection process, expert judgment was applied as a validation mechanism for the inclusion and exclusion criteria. For this purpose, a panel was formed, consisting of two researchers with proven expertise in software engineering, agile methodologies, and specifically DevOps practices. This group critically reviewed the defined criteria, ensuring their alignment with the research questions (RQ) and their suitability within the technical and scientific context of the DevOps domain. The feedback obtained allowed for the refinement of the criteria and ensured that the selected studies were relevant, up-to-date, and directly related to the topics addressed in this systematic review. To guarantee the validity and reliability of the selected studies on DevOps, quality criteria were applied, resulting in the acceptance of 49 studies Subsequently, the snowballing technique was applied following the methodology described by Jalali and Wohlin [11], with the objective of exhaustively expanding the corpus coverage. In this phase, Google Scholar (GS) was used as a complementary source, enabling the identification of 13 additional studies that met the established inclusion criteria. As a result, a final corpus comprising 62 primary studies was consolidated, providing a solid and comprehensive foundation for the proposed systematic analysis, as presented in Figure 2.

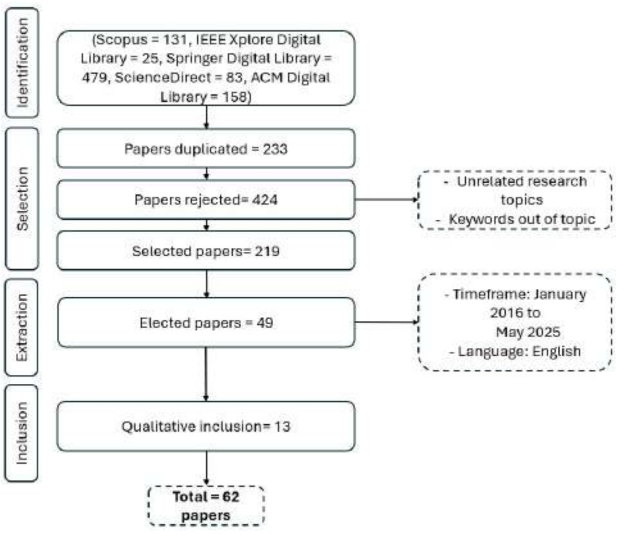
Identification
(Scopus = 131, IEEE Xplore Digital Library = 25, Springer Digital Library = 479, ScienceDirect = 83, ACM Digital Library = 158)
Selection
Papers duplicated = 233
Papers rejected= 424
- Unrelated research topics
- Keywords out of topic
Selected papers= 219
Extraction
Elected papers = 49
- Timeframe: January 2016 to May 2025
- Language: English
Inclusion
Qualitative inclusion= 13
Total = 62 papers

FIGURE 2. Flow of article identification and selection from databases and other sources.

D. CYBERSECURITY LANDSCAPE IN DEVOPS: LITERATURE FINDINGS

This section presents a structured synthesis of the findings derived from the systematic literature review, which focused on the rigorous examination of cyber threats, attack vectors, structural vulnerabilities, mitigation strategies, and their impact on operational efficiency in DevOps environments. Through thematic and comparative analysis of the included studies, recurrent patterns, consolidated methodological approaches, and significant gaps were identified, enabling a more precise delineation of the current state of cybersecurity in contexts of automation and continuous delivery. One notable finding is that over 80% of the studies explicitly focus on issues related to threats and attack vectors that compromise the integrity of CI/CD pipelines, containerized environments, infrastructure as code (IaC), and automation tools. This thematic concentration reveals a high sensitivity within the scientific literature regarding emerging attack surfaces stemming from agile development practices and dynamic orchestration. A growing concern is also identified regarding the inherent weaknesses of automated infrastructure, particularly with respect to the lack of integrated security controls from the early stages of the software development lifecycle. In response to these deficiencies, multiple studies document mitigation strategies aligned with the principles of DevSecOps, such as the use of automated static and dynamic analysis tools, implementation of policy-as-code mechanisms, secure management of secrets and credentials, and the adoption of continuous control frameworks and automated compliance mechanisms. In this regard, some works explore how these security practices affect system agility, performance, and operational resilience, providing empirical evidence on the necessary balance between defensive robustness and delivery efficiency. The results have been organized into three key dimensions, corresponding to the research questions posed in this study: (a) Cyber threats in DevOps environments, (b) Attack vectors in DevOps environments, (c) Methodological approaches for threat mitigation in DevOps environments, (d) Vulnerabilities in DevOps environments, (e) Corrective measures to mitigate vulnerabilities, (f) Effects and implications of security mitigation strategies on the performance of DevOps environments, and (g) Correlation between threats, vulnerabilities, associated risk, and their implications for the security triad.

a) Threats in DevOps Environments

Threats in DevOps environments refer to potential events or conditions that may be exploited by malicious actors to compromise the security, integrity, or availability of systems during the continuous software development and delivery cycle. These threats arise from latent vulnerabilities in practices, configurations, tools, and automated workflows that are inherent to the DevOps culture. The results of the analysis reveal a significant set of recurrent and critical threats, notably including the delayed integration of security measures [25] [27] [30] [33], misconfiguration of Infrastructure as Code (IaC) and containers [14] [17] [26] [28] [31] [47], and poor management of privileged access [13] [21] [45] [50] [55], which enable unauthorized escalation scenarios. Additional threats include data leakage due to unencrypted or improperly managed storage [39] [41] [44] [46], compromised software supply chains [20] [35] [44] [51] [57], and the use of unauthorized tools (Shadow IT) [14] [22] [36] [47] [60], introducing unaudited external vectors. Emerging threats also include the reliance on legacy scripts without security validation [26] [35] [58], the lack of authentication between microservices or integrated tools [12] [13] [48] [61], and false positives or negatives in automated scanning tools, which reduce the accuracy in detecting actual vulnerabilities [38] [42] [49] [52]. These threats, distributed throughout the DevOps pipeline, underscore the need to strengthen security controls and adopt a proactive and continuous stance against cyber risk. Table 3 provides a summary of the threats identified in DevOps environments. The table is structured as follows: from left to right, it includes the type of threat, the type of vulnerability exploited, the importance, frequency and the scientific references where each is mentioned. The assessment of threats in DevOps follows NIST SP 800-30 [74], CVSS v3.1 [75], and ISO/IEC 27005 [76]. High-importance threats compromise system confidentiality, integrity, and availability (CIA) and require immediate controls due to their high occurrence probability [74] [77]. Medium-importance threats have a localized impact and depend on specific conditions for exploitation, while low-importance threats are rare, low-impact, and manageable with minimal intervention [74] [75] [76].

b) Attack Vectors in DevOps Environments

Attack vectors in DevOps environments represent the specific mechanisms through which malicious actors can exploit vulnerabilities and materialize threats within the continuous cycle of development, integration, and operations. The anal-

TABLE 3. Cybersecurity Threats in DevOps Environments

Threat	Type of Exploited Vulnerability	Importance (Details)	Frequency (Details)	References
Delayed security integration	Late implementation of security controls	High: allows vulnerabilities to reach production	High: common practice in immature DevOps models	[25] [27] [30] [33]
IaC and container misconfigurations	Incorrect or default configurations	High: may expose critical services	High: frequent in deployments without validation	[14] [17] [26] [28] [31] [47]
Poorly managed privileged access	Uncontrolled privilege escalation and access	High: compromises internal systems and critical resources	Medium-High: common in CI/CD pipelines and deployments	[13] [21] [45] [50] [55]
Vulnerable third-party dependencies	Insecure or unmaintained libraries	High: gateway for supply chain attacks	High: widely documented in CI/CD analyses	[20] [28] [44] [57] [60]
Human errors and manual practices	Lack of expertise or poor practices	Medium-High: introduces insecure configurations	High: common in teams with limited security training	[22] [30] [36] [47] [51]
Insider threats	Excessive permissions and lack of auditing	High: enables sabotage or internal data theft	Medium: in organizations without strong controls	[22] [48] [54]
Anomalies and unauthorized activities	Absence of continuous monitoring and analysis	High: can conceal real-time attacks	Medium: depends on monitoring automation	[25] [41] [59]
CI/CD pipeline compromise	Uncontrolled modifications and insecure scripts	High: enables malicious code execution	Medium-High: in pipelines with weak controls	[13] [17] [26] [28] [33]
Data leakage via unencrypted storage	Lack of encryption at rest and in transit	High: exposes sensitive information	Medium: depends on applied policies	[39] [41] [44] [46] [50]
Supply chain attacks	Compromised dependencies or tools	High: injects malware into the DevOps cycle	High: critical threat in CI/CD	[20] [35] [44] [51] [57]
Injection via deployment scripts	Weak validations and shell injection	High: enables remote command execution	Medium: common in loosely controlled processes	[28] [36] [42] [58]
Lack of authentication between microservices	Open or unauthenticated connections	High: enables unauthorized access	Medium-High: frequent in distributed architectures	[12] [13] [48] [61]
Misconfigured static analysis	False negatives due to poor rule calibration	Medium: false sense of security	Medium: depends on team expertise	[38] [42] [46] [52]
Unauthorized tools (Shadow IT)	Lack of governance and control	Medium-High: introduces unevaluated software	Medium: common without centralized policy	[14] [22] [36] [47] [60]
Mismanaged public repositories	Code or secret exposure due to human error	High: reveals sensitive configurations	High: frequent without permission controls	[13] [20] [30] [33] [51]
Internal network policy errors	Lack of segmentation and traffic control	High: allows lateral movements	Medium: depends on orchestration and visibility	[21] [23] [54]
Lack of version control in container images	Use of outdated or compromised images	High: retains known vulnerabilities	High: in fast-paced deployments without maintenance	[13] [17] [20] [29] [33]
Legacy insecure scripts	Reuse of unaudited code	Medium-High: logical errors or insecure settings	Medium: common in technical debt	[26] [28] [35] [45]
False positives/negatives in scanning tools	Misinterpretation of findings during analysis	Medium: affects decisions and leaves threats untreated	Medium: depends on tool calibration	[38] [42] [49] [52]
Authentication failures in integrated tools	Implicit trust without authentication	High: enables lateral access between modules	Medium-High: in multi-vendor architectures	[12] [13] [30] [32] [48]

ysis shows that these vectors are closely linked to insecure configurations, mismanaged automation, exposed credentials, and lack of segmentation between services. One of the most critical vectors is code injection into CI/CD scripts, enabling the automated execution of malicious commands within continuous integration and deployment environments [13] [17] [26] [28] [33]. Another high-impact vector involves uncontrolled access to public repositories, allowing for the exfiltration of secrets, tokens, or exposed configurations [13] [20] [30] [33] [51]. Remote code execution is also documented through default settings in containers and Infrastructure as Code (IaC), which often enable unauthenticated or exposed services [14] [17] [26] [31] [47] [56]. Privilege escalation via hardcoded or poorly managed credentials is another common vector [13] [21] [45] [50] [61], as is the injection of malicious dependencies during the software build process—one of the main entry points in supply chain attacks [20] [28] [35] [44] [60]. An often underestimated vector is the reuse of insecure legacy scripts, which may contain outdated commands, misconfigurations, or excessive permissions. These scripts frequently remain in place due to delivery pressures and lack of technical governance. As noted in [37], intensive use of microservices without a controlled architecture can increase technical debt, thereby expanding the attack surface by leaving insecure components in production. Finally, additional vectors were identified, including the use of unauthorized tools (Shadow IT) [22] [36] [47] [60], the integration of systems without cross-authentication [12] [30]

[32] [48], and the absence of network policies or segmentation, which facilitates lateral movements within orchestrated architectures [21] [23] [54] [61]. These vectors, aligned with the threats previously described, provide a clearer understanding of the most exploited entry points in modern DevOps environments. In this context, emerging technologies such as deep learning have proven effective in detecting complex threats. For example, [53] presents a review of deep neural network-based architectures for protecting wireless systems, which could be extrapolated to distributed DevOps environments, where predictive models enable the detection of anomalies that traditional rule-based systems may overlook. Table 4 summarizes the most relevant attack vectors in DevOps environments. The table is structured as follows: from left to right, attack vector type, associated threat, importance, frequency and the scientific references where each is mentioned. Critical attack vectors enable direct access or manipulation of essential assets via exploitable routes such as repository exposure or CI/CD pipeline tampering, frequently identified in audits and security frameworks [74] [75] [76]. Medium-impact vectors target non-essential services and require specific conditions for exploitation, while low-impact vectors depend on highly uncommon technical setups, making them infrequent in real-world audits [74] [75] [76].

c) Methodological Approaches for Threat Mitigation in DevOps Environments

Methodological approaches for mitigating threats in DevOps environments focus on integrating proactive and adaptive defense mechanisms into the continuous development, integration, and operations cycle. The analysis reveals that these methodologies respond directly to specific combinations of threats and attack vectors, employing techniques ranging from automated controls to organizational policies. Notably, the implementation of digital signatures and script validation in CI/CD pipelines helps prevent code injection during the integration process [13][17][26][28][33]. Similarly, automating the scanning of Infrastructure as Code (IaC) and containers proves to be an effective strategy for mitigating misconfigurations and unintentional exposures [14][17][26][31][47][56]. Centralized secret management and Role-Based Access Control (RBAC) emerge as critical practices to prevent privilege escalation and unauthorized access [13][21][45][50][61]. Regarding the software supply chain, the use of Software Composition Analysis (SCA) tools and secure versioning policies is effective in blocking the injection of malicious dependencies [20][28][35][44][60]. Other documented approaches relate to observability and operational security feedback, through the deployment of integrated alerts, event dashboards, and automated response systems [25][41][59]. In this context, [69] propose a deep learning-based approach to detect anomalous behaviors in DevOps pipelines. Their model enables real-time identification of deviations, even when threats do not match predefined patterns, enhancing response capabilities and reducing exposure time to complex attacks. These approaches, aligned with well-established standards and frameworks such as NIST, OWASP, and DevSecOps, provide a solid foundation for strengthening the resilience and comprehensive security of modern DevOps implementations. Among these frameworks, [64] recommend the use of the OSCAL language, developed by NIST, as an effective solution to automate regulatory compliance. This approach enables the structured representation of security controls, facilitating their integration into DevSecOps pipelines and continuous verification against frameworks such as NIST SP 800-53 or ISO/IEC 27001. Table 5 provides a summary of the methodological approaches for threat mitigation in DevOps environments. The table is structured as follows: from left to right, it presents the threat type, attack vector, methodological mitigation approach, application level, and references showing the scientific articles in which they are discussed.

d) Vulnerabilities in DevOps Environments

Vulnerabilities in DevOps environments represent critical exposure points that compromise the integrity, confidentiality, and availability of systems throughout the entire development and operations lifecycle. These weaknesses arise from insecure automated practices, misconfigurations, organizational failures, and the late integration of security into continuous delivery processes. Among the most relevant vulnerabilities is the lack of automated security testing, which allows undetected flaws to propagate into production environments [5][13][17][26][30]; the exposure of secrets in CI/CD pipelines, enabling unauthorized access to critical resources [13][14][45][50]; and insecure configurations in containers and Infrastructure as Code (IaC), which expand the attack surface and create favorable conditions for exploitation [14][17][26][28][31][47]. In addition, the absence of threat modeling, lack of continuous monitoring, and the use of static analysis tools with low coverage or limited precision reduce the effectiveness of threat detection and response [25][38][41][46][52]. Other structural vulnerabilities include the use of unverified external dependencies, which compromise the software supply chain [20][28][35][44][60], and poor management of privileged access, enabling potential internal escalations [13][21][45][50][61]. These weaknesses are distributed across development, integration, deployment, and operations phases, highlighting the need for comprehensive and preventive approaches that address the dynamic and automated nature of DevOps environments. Beyond technical deficiencies, the human dimension is a critical factor in DevOps security. The lack of a security-oriented organizational culture, pressure to release code rapidly, and insufficient specialized training can lead to costly errors and unintentional failures. In this context, [62] conducted an empirical study that identifies multiple human-induced risks within DevOps teams, demonstrating that human decision-making and poor knowledge management directly influence the emergence of vulnerabilities and compliance failures. Table 6 provides a summary of vulnerabilities identified in DevOps environments. The table is structured as follows: from left to right, it includes the vulnerability, vulnerability

TABLE 4. Attack Vectors in DevOps Environments

Attack Vector	Associated Threat	Importance (details)	Frequency (details)	References
Code injection in CI/CD scripts	CI/CD pipeline compromise	High: automated execution of malicious code	Medium-High: in pipelines lacking validation and control	[13][17][26][28][33]
Uncontrolled direct access to public repositories	Poorly managed public repositories	High: exposure of code and secrets	High: lacking access and visibility policies	[13][20][30][33][51]
Remote execution due to default configurations	Misconfigurations in IaC and containers	High: exposes services without authentication	High: frequent without manual review or scanning	[14][17][26][31][47][56]
Privilege escalation via embedded credentials	Poorly managed privileged access	High: compromises internal resources with elevated permissions	Medium-High: in deployments lacking privilege separation	[13][21][45][50][61]
Insertion of malicious dependencies in the build	Vulnerable third-party dependencies	High: modifies behavior without detection	High: no external component scanning	[20][28][35][44][60]
Data leakage due to plaintext storage or misconfigured buckets	Data breach from unencrypted storage	High: exposes information without elevated privileges	Medium: lacking clear encryption policies	[39][41][44][46][49][50]
Abuse of unauthorized tools by staff	Use of unauthorized tools (Shadow IT)	Medium-High: introduces unaudited software	Medium: low control over devices	[22][36][47][60]
Insecure connections without mutual authentication	Lack of authentication between microservices	High: lateral movements within orchestration	Medium-High: no mTLS or JWT	[21][23][30][54][61]
Legacy scripts with unsafe commands	Dependence on insecure legacy scripts	Medium-High: destructive operations without validation	Medium: in projects with high technical debt	[26][28][35][45][37]
Unlogged manual actions	Human errors and manual practices	Medium-High: modifications without traceability	High: absence of automated controls	[22][30][36][47][51]
Exposure of S3 buckets or volumes without encryption	Data breach from unencrypted storage	High: access to sensitive data without authentication	Medium: depends on security policies	[39][41][44][46][49][50]
Lateral movement via unsegmented services	Flaws in internal network policies	High: facilitates attack propagation	Medium: lacking network segmentation	[21][23][54][61]
Command execution without validation from pipelines	Injection via deployment scripts	High: persistent remote execution	Medium: uncontrolled legacy scripts	[28][36][42][58]
Compromised software through supply chain	Software supply chain attacks	High: indirect malicious code	High: common in DevOps pipelines	[20][26][35][44][57][60]
Manipulation of environment variables	False positives/negatives in scanning tools	Medium: affects vulnerability diagnostics	Medium: depends on tool quality	[38][42][46][52]
Use of outdated or unsigned container images	Lack of version control over container images	High: exposes systems to vulnerabilities	High: common without automated scanning	[13][17][20][29][33]
Lack of feedback on anomalies	Anomalies and unauthorized activities in production	High: undetected attacks	Medium: depends on observability	[25][41][59]
Integration without cross-authentication	Authentication failures between integrated tools	High: unauthorized access among components	Medium-High: integration lacking authentication	[12][30][32][48]
Improper access due to excessive privileges	Insider threats	High: users with malicious permissions	Medium: no RBAC or audit controls	[22][48][54]
Automation without regulatory compliance	Automation misaligned with regulations	Medium-High: legal noncompliance or breaches	Medium: depends on regulatory framework	[39][47][48][50][51]

type, exploitability, importance, frequency and the scientific references where each is mentioned. A vulnerability is critical if it enables remote execution, privilege escalation, or unauthorized access, with CVSS scores $\geq$ 7.0 [75] and a high occurrence probability in security assessments [74] [76]. Medium-level vulnerabilities impact secondary assets, requiring authenticated access or specific configurations, while low-level vulnerabilities have minimal security implications and require complex conditions for exploitation [74] [75] [76].

e) Corrective Measures to Mitigate Vulnerabilities in DevOps Environments

Corrective measures aimed at mitigating vulnerabilities in DevOps environments represent a fundamental response to the security deficiencies identified across the various stages of the software lifecycle. Findings reveal a multifactorial approach in the design and implementation of these actions, ranging from automated technical strategies to governance practices and regulatory compliance. Key measures include the integration of automated security testing into CI/CD pipelines, enabling early detection of flaws prior to deployment [13][17][26][30]; secure secret management through vaults, access segmentation, and periodic credential rotation [13][14][45][50]; and the adoption of the DevSecOps approach, which embeds security structurally into development from its initial phases [12][21][24][27]. Other critical actions involve the continuous validation of configurations in Infrastructure as Code (IaC) and containers, mitigating errors derived from unaudited automation [14][17][26][28][31], as well as active operations monitoring integrated with feedback loops to development teams, enabling agile responses to emerging incidents [25][34][41][59]. Beyond early-stage adoption, it is essential to implement a coordinated set of

TABLE 5. Methodological Approaches for Threat Mitigation in DevOps Environments

Threat	Attack Vector	Mitigation Methodology	Application Level	References
CI/CD pipeline compromise	Code injection in CI/CD scripts	Digital signatures integration, script validation, and isolated execution environments	CI/CD - Integration and Deployment	[13][17][26][28][33]
Poorly managed public repositories	Uncontrolled access to public repositories	Access policies enforcement, periodic audits, and repository visibility control	Development and Code Control	[13][20][30][33][51]
IaC and container misconfigurations	Remote execution via default configurations	IaC validation automation, image scanning, and tools like kube-bench	Infrastructure and Deployment	[14][17][26][31][47][56]
Poor privileged access management	Privilege escalation via embedded credentials	RBAC enforcement, MFA, credential auditing, and centralized secrets	Identity Operations and Security	[13][21][45][50][61]
Vulnerable third-party dependencies	Malicious dependencies in build process	Use of SCA tools and secure minimum version policies	Build and Dependency Management	[20][28][35][44][60]
Unencrypted storage	Data leakage via plaintext or misconfigured buckets	Encryption in transit and at rest, key management, and RBAC-based access policies	Infrastructure and Data Security	[39][41][44][46][49][50]
Missing microservice authentication	Insecure microservice connections without mutual auth	mTLS implementation, JWT-based auth, and secure API gateways	Deployment and Service Communication	[21][23][30][54][61]
Unauthorized tool usage	Abuse of unauthorized tools by internal staff	Approved tools catalog, software usage monitoring	Tool Governance and Management	[22][36][47][60]
Deployment script injection	Command execution without validation from pipelines	Script audit, secure interpreters, input validation, shell injection controls	Integration, QA and Deployment	[28][36][42][58]
Software supply chain attacks	Compromised software via supply chain	SLSA controls, third-party code scanning, provenance verification, continuous review	Build and Software Security	[20][26][35][44][57][60]
Human error and manual practices	Unlogged manual operator actions	Automation of critical tasks, dual validation, secure ops training	Operations and Personnel Security	[22][30][36][47][51]
Insecure legacy dependencies	Legacy scripts with unsafe commands	Script refactoring, internal dependency review, security linters	Development and QA	[26][28][35][45]
Inaccurate scan tools	Manipulation of environment variables in unsafe contexts	Continuous evaluation of tool accuracy, combined static, dynamic, and interactive analysis	QA and Application Security	[38][42][46][52]
Poor container image control	Use of outdated/unsigned container images	Secure versioning policies, private registries, automated image scanning	Build and Containers	[13][17][20][29][33]
Unauthorized production anomalies	Lack of feedback on operational anomalies	Observability, security event correlation, active alerts with auto-response	Operations and Monitoring	[25][41][59]
Tool integration auth failures	Tool integration without cross-authentication	Mutual authentication, token control, federated identity management	Integration and Tool Security	[12][30][32][48]
Insider threats	Improper access due to excessive privileges	Zero Trust, permission segmentation, behavioral monitoring, continuous audit	Internal Security and Access Control	[22][48][54]
Non-compliant automation	Automation without regulatory compliance	Automated control mapping (NIST, ISO, OWASP), compliance tools like OpenSCAP	Governance and Compliance	[39][47][48][50][51][64]
Lack of feedback on operational anomalies	Lack of feedback on operational anomalies	SIEM deployment, integrated alerts, and automatic feedback to development teams	Operations and Continuous Monitoring	[25][41][59]

tools and practices tailored to each phase of the DevSecOps cycle. In [71], a comprehensive review of existing solutions is presented, categorizing tools by purpose: static analysis (SAST), dynamic analysis (DAST), secure secret management, compliance-as-code, and operational monitoring. Their study provides an integrated framework which allows organizations to select and implement solutions based on their maturity and security objectives. Additionally, collaboration between development, operations, and security teams has proven to be a key factor for effective integration of security into the pipeline. In [70], structured collaborative practices are shown to facilitate shared vulnerability management, enhance cross-functional communication, and strengthen the overall security posture without compromising operational efficiency. This approach aligns with the philosophy of shift-left security, which emphasizes the need to incorporate security controls from the design, coding, and planning stages. In [68], an architecture is proposed that integrates threat analysis, early dependency validation, and compliance policies from the earliest phases of the DevOps pipeline, significantly reducing the propagation of vulnerabilities to later stages. Moreover, [65] proposes an integrated approach for the automated management of security findings within CI/CD pipelines, enabling classification, prioritization, and automated response activation for vulnerabilities detected by SAST and DAST tools. This practice reduces remediation latency and improves the traceability of corrective actions without manual intervention. There is also a growing trend toward the comprehensive protection of the software supply chain through the use of

TABLE 6. Vulnerabilities in DevOps Environments

Vulnerability	Vulnerability Type	Exploitable?	Importance (details)	Frequency (details)	References
Lack of automated security testing	Insufficient testing	Yes	High: undetected gaps may compromise the system	High: frequent in DevOps without integrated security	[13][17][26][30]
Exposure of confidential secrets	Secret management	Yes	High: may compromise credentials and access tokens	High: frequent in misconfigured pipelines	[13][14][45][50]
Misconfiguration in containers	Configuration	Yes	High: expands attack surface	High: common in microservice environments	[14][17][26][28][31][47]
Late security integration	Process	Yes	High: allows vulnerabilities into production	High: common in traditional DevOps	[25][27][30][33]
Lack of granular access control	Unrestricted access	Yes	High: unauthorized access to critical resources	Medium: depends on policy implementation	[13][21][45][50][61]
Use of vulnerable dependencies	External dependencies	Yes	High: compromised libraries may introduce attacks	High: common in projects with many packages	[20][28][35][44][60]
Compromised CI/CD pipelines	Supply chain	Yes	High: may inject malicious code into production	Medium-High: depends on control in tools like Jenkins	[13][17][26][28][33]
Poor security culture	Cultural	Indirectly	Medium-High: limits adoption of best practices	High: frequent in non-DevSecOps organizations	[22][24][27][62]
Lack of secure coding standards	Development practices	Yes	High: introduces avoidable logic and validation flaws	High: common in fast or unreviewed development	[13][14][20]
Inadequate secret management in CI/CD	Secret management	Yes	High: allows unauthorized access to critical environments	High: common in pipelines lacking dedicated tools	[13][14][20][45][50]
Lack of threat modeling	Security analysis	Yes	High: unidentified threats may be directly exploited	Medium: frequent in low-maturity environments	[25][38][46]
Exposed attack surfaces	Insecure architecture	Yes	High: enables attackers to find and exploit open services	Medium: depends on architecture design	[17][26][28][31]
Lack of security integration in DevOps tools	Tools	Yes	High: tools without controls facilitate attacks	High: frequent in immature DevSecOps setups	[12][13][21][30]
Reactive security testing	Testing process	Yes	Medium-High: allows flaws into production	Medium: depends on QA strategy adopted	[13][22][27][30]
Lack of code change traceability	Configuration management	Yes	Medium-High: hinders root cause identification for vulnerabilities	Medium: depends on versioning and audit systems	[14][19][30]
Microservices architecture complexity	Distributed architecture	Yes	High: each microservice needs individual security	High: common in modern DevOps adoption	[13][17][20]
Outdated container images	Obsolescence	Yes	High: old components contain known vulnerabilities	High: common in multi-integration environments	[13][17][20][29][33]
Unverified toolchains	Supply chain	Yes	High: public components may contain backdoors	Medium-High: frequent without scanning integrations	[20][28][29]
Poor management of privileged access	Identity management	Yes	High: enables internal attacks or privilege escalation	Medium: depends on RBAC and audit policies	[13][21][45][50][61]
Lack of security feedback from operations	Feedback cycle	Indirectly	Medium: recurring errors due to poor communication	Medium: seen in traditional structures	[25][41][59]
Automation without security validation	Automation	Yes	High: insecure flows rapidly propagate vulnerabilities	High: very common without DevSecOps practices	[13][14][25][30]
Lack of real-time cybersecurity monitoring	Monitoring and auditing	Yes	High: prevents detection of ongoing or anomalous attacks	Medium: common in undersupervised environments	[38][41][46][52]
Persistent threats in CI/CD without traceability	Supply chain	Yes	High: hides malicious code in production environments	Medium-High: seen in pipelines without historical scanning	[35][36][40][42][45]
Data leakage due to insecure storage	Data protection	Yes	High: exposes critical information like passwords or policies	High: common in shared or unencrypted storage	[39][41][44][46][49]
Security automation misaligned with regulations	Compliance	Indirectly	Medium: may result in sanctions or legal gaps	Medium: frequent when regulation is not built-in	[39][47][48][50][51]
Lack of security in static analysis tools	Tools	Yes	Medium-High: poor results may hide threats	Medium: depends on outdated tool usage	[38][42][46][52]
Excessive permissions in orchestrated services	Orchestration and access control	Yes	High: facilitates lateral movement and internal intrusions	High: common when least privilege is not enforced	[43][50][54][55][56]
Use of unverified images in CI	Supply chain	Yes	High: may allow external code execution	High: frequent when using public repositories	[44][51][57][58]

tools for dependency analysis, third-party image and code scanning [20][28][35][44][60], and the implementation of automated controls aligned with regulatory frameworks such as NIST and ISO/IEC 27001, ensuring compliance without compromising operational speed [39][47][48][51]. These measures represent robust practices, supported by scientific evidence, that enhance security posture while preserving the agility and operational efficiency characteristic of DevOps environments. This orientation has also extended to Operational Technology (OT) environments, where operational continuity, physical safety, and regulatory compliance are essential requirements. A tailored DevSecOps model has been proposed that enables the integration of automated controls alongside strategic manual validations, compatible with the technical constraints of critical industrial systems [63]. In highly regulated contexts, implementing DevSecOps requires additional corrective measures to ensure regulatory compliance without compromising automation. In [67], best practices are documented for adapting DevSecOps to sectors such as healthcare or finance, proposing hybrid controls that integrate automation with selective manual validations. Table 7 summarizes the corrective measures for mitigating vulnerabilities in DevOps environments. The table is structured as follows: from left to right, it presents the corrective measure, related vulnerability, description, DevOps lifecycle application, expected impact, and references indicating the scientific articles where they are discussed.

f) Effects and Implications of Security Mitigation Strategies on DevOps Performance

The analysis shows that various mitigation strategies applied in DevOps environments have distinct effects on system performance, creating a balance between cybersecurity protection and operational efficiency. The most effective strategies are those that natively integrate security into the software lifecycle, such as automated security controls, Infrastructure as Code (IaC), automated secret management, and the principle of least privilege with segmented RBAC [13][14][17][26][30][45][50]. These practices help reduce manual errors, ensure consistency across distributed environments, and maintain stable deployment times without introducing significant latency or computational overhead [31][33][47]. Recent proposals emphasize the importance of adapting security management to agile frameworks, incorporating lightweight yet effective practices that meet security requirements without causing friction in short delivery cycles. These adaptations enable the DevSecOps model to align with iterative methodologies, improving both delivery speed and software quality in dynamic environments [73]. In terms of performance, mechanisms such as monitoring as code, the use of secure containers, and the instrumentation of automated testing in CI/CD pipelines help preserve system availability and quality without penalizing delivery speed [14][25][30][38][42][46][52]. In parallel, more advanced strategies such as the application of artificial intelligence in DevSecOps, continuous risk assessment, and compliance-as-code models demonstrate high adaptability and accuracy in threat detection. However, they require careful resource management to avoid operational overhead in highly dynamic scenarios [34][39][47][48][51]. In this regard, [72] explore how artificial intelligence can enhance automation within the DevSecOps lifecycle, proposing models for autonomous threat detection, dynamic security policy generation, and contextual adjustment of controls based on pipeline behavior. Their study shows improvements in both response capacity and reduction of operational overhead in dynamic environments. Finally, hybrid approaches—such as the combination of manual and automated controls, the adoption of immutable virtual machines, and specialized security training for developers—prove complementary in strengthening security posture without degrading performance [19][24][27][36][48][51]. Collectively, the findings reinforce the importance of adopting mitigation strategies that are not only technically effective but also aligned with the automated, distributed, and continuous nature of modern DevOps environments. Table 8 presents the effects and implications of security mitigation strategies on the performance of DevOps environments. The table is structured as follows: from left to right, it includes the mitigation strategy, its impact on system performance, its effectiveness level in applied scenarios, and the references that support the scientific discussion. Highly effective strategies significantly reduce risks, integrate into CI/CD pipelines, and align with regulatory frameworks [74] [75] [76]. Medium-effectiveness measures provide partial risk coverage and require complementary controls, while low-effectiveness strategies are reactive, difficult to scale, and disconnected from DevSecOps practices, limiting their real-world impact [74] [75] [76].

g) Correlation Between Threats, Vulnerabilities, Associated Risk, and Their Implications on the Security Triad

To strengthen the security analysis in DevOps environments, the MAGERIT methodology (Methodology for Information Systems Risk Analysis and Management) [77] was applied to rigorously assess the risks associated with the mitigation strategies identified in the literature. This methodology enabled the establishment of a correlation between detected threats and vulnerabilities and their impact on the fundamental principles of information security: confidentiality, integrity, availability, authentication, and traceability. The evaluation included risk estimation by combining the likelihood of threat occurrence with the potential impact on information assets, facilitating an objective prioritization of the applied controls. Additionally, the ability of each strategy to mitigate critical incidents affecting operational continuity, credential protection, regulatory compliance, and deployment chain security was analyzed. The application of MAGERIT provided a structured and reproducible framework that complements the technical-operational focus of mitigation strategies, allowing their effectiveness to be assessed not only in terms of performance but also from a comprehensive perspective based on cyber risk management. Table 9 presents the

TABLE 7. Corrective Measures to Mitigate Vulnerabilities in DevOps Environments

Corrective Measure	Related Vulnerability	DevOps Lifecycle Phase	Expected Contribution	References
Integration of automated security testing	Lack of automated security testing	Development, Integration, Testing	Early vulnerability identification and risk reduction in production.	[13][17][26][30][38][42][46][52]
Secure secret management	Inadequate secret management in CI/CD	Integration, Deployment, Operations	Unauthorized access prevention and regulatory compliance.	[13][14][20][45][50]
Configuration validation in IaC and containers	Misconfigurations in containers	Build, Integration, Deployment	Reduction of human errors and secure implementation consistency.	[14][17][26][28][31][47]
DevSecOps: security from the start	Late security integration	Planning, Development, Integration	Proactive threat mitigation and enhanced security posture.	[12][21][24][27][68]
Continuous analysis and monitoring	Lack of real-time cybersecurity monitoring	Operations, Maintenance	Immediate incident response and persistent threat detection.	[25][34][41][59]
Supply chain vulnerability evaluation	Use of vulnerable dependencies	Build, Deployment	Prevention of exploitation through compromised components.	[20][28][35][44][60]
Access control policies and least privilege	Poor management of privileged access	Integration, Deployment, Production	Reduced attack surface and limited internal threat damage.	[13][21][45][50][61]
Automated security finding management	Latency in vulnerability response	Integration, Validation, Deployment	Reduced exposure time and response without human intervention.	[65]

correlation between threats, vulnerabilities, and associated risk, as well as their effects and implications on confidentiality, integrity, availability, authentication, and traceability. The classification of the impact of threats in DevOps environments on the components of the security triad (Confidentiality, Integrity, Availability), as well as on Authentication and Traceability, was based on empirical evidence extracted from the corpus of analyzed scientific articles and supported by risk assessment methodologies such as OWASP Risk Rating, NIST SP 800-30, and MAGERIT. Threats related to poorly implemented secret management in CI/CD pipelines, for instance, were rated as having a high impact on confidentiality and authentication, as they allow unauthorized access to credentials, tokens, and protected artifacts [13][14][50]. Similarly, misconfigurations in IaC and containers were rated as high impact on availability and integrity, since errors in automated infrastructure can disrupt services, generate unsafe execution conditions, and facilitate remote attacks [14][17][26]. Meanwhile, threats such as the use of unauthorized tools (Shadow IT) or the lack of traceability in code changes were associated with a high impact on traceability, as they hinder auditability, digital forensics, and version control processes [41][48][54]. Regarding the probability of occurrence, threats widely documented in real-world environments and frequently reported in pipelines (e.g., privilege escalation through embedded credentials or malware persistence in pipelines without continuous auditing) were rated as high probability, considering their prevalence and ease of exploitation [28][33][45]. In contrast, threats with contextual dependencies—such as malicious AI techniques or failures due to lack of security integration in less common tools—were assessed as having medium or low probability. The assessments presented throughout all tables in this study—including those related to threats, attack vectors, vulnerabilities, mitigation strategies, their effects on performance, and the correlation between threats, vulnerabilities, and associated risks—were established through a rigorous analysis process grounded in technical criteria and expert professional experience in cybersecurity and DevOps environments. These evaluations integrate empirical evidence from the validated document corpus and expert judgment to determine influence, frequency, criticality, operational consequences, and risk relationships. The categorization into levels such as high, medium, and medium-high reflects a well-supported consensus that ensures both the practical and scientific relevance of the results. This robust methodological approach guarantees that the conclusions and recommendations are backed by a scientifically sound framework aligned with international standards and industry best practices, while also considering the balance between security, risk, and performance in dynamic DevOps environments.

IV. DISCUSSION

a) Threats in DevOps Environments

Figure 3 summarizes the most critical threats identified in the analysis of 62 scientific articles on cybersecurity in DevOps environments. These threats span technical, procedural, and organizational dimensions, reflecting the ongoing tension between deployment speed and continuous security. Although there is broad recognition of common threats, significant limitations remain in their comprehensive treatment and empirical validation. One of the most recurrent findings is the exposure of misconfigurations in IaC and containers. Rangnau et al. [14] and Adhikari and Pillai [47] highlight that the lack

TABLE 8. Impact of Security Mitigation Strategies on DevOps Performance

Mitigation Strategy	System Performance Impact	Effectiveness Level	References
Security automation and early integration (DevSecOps)	Reduces post-deployment errors while maintaining lifecycle agility	High: improves security without compromising speed	[14][8][17][26][30]
Continuous monitoring and monitoring-as-code	No significant latency, enables rapid incident response	High: proactive detection without excessive overhead	[8][20][24][30]
Policy-as-Code	Eliminates administrative bottlenecks, maintains regulatory consistency	High: reduces manual errors and accelerates compliance	[12][15]
Infrastructure as Code (IaC)	Enhances environment stability and consistency without resource burden	High: prevents misconfigurations and enables scalability	[7][15]
Automated testing and security gates in CI/CD	Integrates smoothly, reinforces software quality	High: mitigates risk before production	[17][18]
Automated secret and credential management	Optimizes authentication without human error	High: improves availability and access protection	[10]
Secure containers (e.g., SecDocker)	Maintains deployment speed, improves isolation	High: reduces attack surface in orchestrated environments	[8]
AI applied to DevSecOps	Moderate resource load, enhances threat detection precision	High: enables real-time contextual response	[16]
Least privilege principle and segmented RBAC	Does not interfere with authentication, improves access control	High: blocks unauthorized escalation without friction	[13][6]
Native Kubernetes security	Proper configuration ensures stable and secure performance	High: improves orchestration and resource isolation	[4]
Immutable virtual machines	Preserves integrity post-deployment; no post-maintenance needed	High: avoids post-production vulnerabilities	[28][29]
Slide-Block framework with pattern-based authentication	Reduces auth time and improves availability	High: improves detection and resilience with low impact	[21][11]
Security as Code	Avoids manual configuration errors; integrates seamlessly	High: consistent policies without manual intervention	[31][5]
Monitoring-as-Code	Automates monitoring without performance penalty	High: improves operational security with low overhead	[14][32]
Continuous risk assessment (FAIR)	Optimizes resource allocation, avoids degradation	High: enables proactive prioritization	[16][23]
Compliance-as-Code	Simplifies audits and reduces manual load	High: maintains regulatory conformity without friction	[17][5][16]
Hybrid controls (manual and automated)	Balances flexibility and automation; delivery speed unaffected	Medium-High: context-dependent effectiveness	[18][10]
VeriDevOps (automated monitors and formal tests)	Reduces risks with minimal operational impact	High: integrates continuous detection and validation	[20][30]
Security training for developers	Improves code quality and reduces design flaws	High: strengthens resilience with no computational cost	[17][8]
Agentless credential management solutions	Improves scalability with minimal overhead	High: enables secure centralized administration	[13][7]

of structured validation in automated environments results in poorly secured services that are accessible from external networks. Zhao and Lu [44] complement this perspective by demonstrating that default configurations can be exploited without prior authentication. However, Pan et al. [20] focus their analysis on deployment efficiency without addressing the impact of IaC on the attack surface. Regarding poor management of privileged access, Cifuentes et al. [13] and Tounsi and Rais [21] report that the lack of token rotation, embedded credentials, and absence of RBAC create internal escalation pathways. Kumar et al. [33] reinforce this by showing how weak configurations in integration tools allow uncontrolled lateral access. Nonetheless, studies such as Kumar et al. [28] recognize pipeline criticality but omit identity control as a structural threat. Threats arising from lack of authentication between microservices and inadequate network policies are discussed in [6] [23] [30] [54], where the absence of mTLS and internal segmentation is shown to facilitate lateral attacker movement. Long et al. [41] propose architectures with cross-authentication as a mitigation measure, while other studies such as Zeng et al. [12] address microservices without incorporating specific authentication controls, limiting the applicability of their proposals. In terms of the human component, Rangnau et al. [14] and Adhikari and Pillai [47] mention the prevalence of Shadow IT and unsafe manual tasks as frequent threats, particularly in organizations with low DevSecOps maturity. However, many studies such as Mohan and Ben Othmane [21] and Casola et al. [16] focus exclusively on technical aspects without addressing how organizational culture influences the emergence or persistence of these threats. There is also weak coverage of threats such as reliance on legacy scripts [14] [26] [28] or false positives/negatives in automated scanners [11] [38] [46] [52]. Satapathy et al. [52] propose progressive calibration of SAST and DAST tools, while Sharma et al. [35] document how unreviewed legacy scripts introduce silent failures. However, these proposals lack comparative evaluations or precision and coverage metrics. Finally, topics such as automation misaligned with compliance standards [39] [47] [48] and persistence of data in unencrypted storage [44] [46] [49] are

TABLE 9. Correlation Between Threats, Vulnerabilities, and Associated Risk on the Security Triad

Threat	Vulnerability	C	I	A	Auth	Trace	Risk (Probability and Impact)
CI/CD pipeline compromise	Lack of controls in early stages	High	High	Medium	Medium	High	High probability / High impact
Config errors in virtualized envs	Inconsistent manual configurations	Medium	High	High	Medium	Medium	Medium probability / High impact
Secret leakage	Embedded credentials in code/repos	High	High	Medium	High	Medium	High probability / High impact
Code/script injection	Lack of continuous security testing	Medium	High	High	Medium	High	High probability / Medium impact
Non-compliance	Manual control configurations	High	High	Medium	Medium	High	Medium probability / High impact
Supply chain attacks	Unverified container images	High	High	Medium	Medium	Medium	High probability / High impact
Undetected threats	Lack of proactive detection	High	High	High	High	High	High probability / High impact
Audit/compliance failure	Controls outside standard	High	High	Medium	Medium	High	Medium probability / High impact
Persistent vulns in production	Manual updates or missing restart	Medium	High	Medium	Medium	High	Medium probability / Medium impact
Pipeline changes not validated	No auto-validation before deployment	Medium	High	High	Medium	High	Medium probability / High impact
Service segmentation failures	Insecure cluster configuration	High	High	High	High	Medium	High probability / High impact
Critical threat deprioritization	Lack of exposure awareness	Medium	High	High	Medium	High	Medium probability / High impact
Human errors introducing flaws	Lack of security knowledge	Medium	High	Medium	Medium	Medium	High probability / Medium impact
No continuous monitoring	Lack of event visibility	Medium	Medium	High	Medium	High	Medium probability / High impact
Weak authentication mechanisms	Predictable login methods	High	Medium	Medium	High	High	High probability / High impact
Unauthorized access	No centralized access control	High	Medium	High	High	Medium	High probability / High impact
Tool-process misalignment	Unaligned technical/operational security	Medium	High	Medium	Medium	High	Medium probability / Medium impact
Policy inconsistency	Controls not in source code	High	High	Medium	Medium	High	Medium probability / High impact
Use of unauthorized tools	Lack of stack control	Medium	Medium	High	Medium	High	Medium probability / Medium impact
Vulnerable third-party dependencies	Libraries w/o security validation	High	High	Medium	Medium	Medium	High probability / High impact

addressed from a normative perspective. Zouari et al. [60] propose a verified automation model using compliance policies, but other studies like Casola et al. [51] only mention these threats without offering a formal mitigation framework. This reveals a disconnect between problem recognition and structured resolution.

b) Attack Vectors in DevOps Environments

Figure 4 shows the most frequently identified attack vectors from the analysis of the scientific literature. These vectors represent specific technical pathways through which attackers exploit the threats described in DevOps environments. While there is general consensus regarding the main vectors, the depth of treatment and validation of the proposed approaches varies significantly across studies. One of the most frequently reported vectors is the injection of malicious code into CI/CD pipelines, as discussed in [13] [26] [28] [33] [40]. Cifuentes et al. [13] explain how the lack of script validation allows arbitrary code execution in automated environments. Nadeem and Shameem [26] and Neharika and Lennon [28] propose mitigating this risk through digital signatures and pipeline

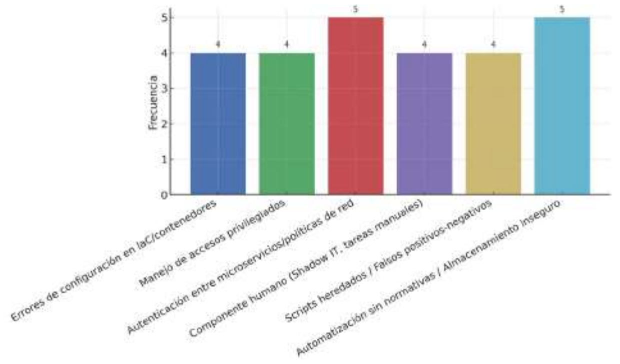

FIGURE 3. Frequency of Identified Threats in DevOps Environments

segmentation, while Zhang and Zhang [40] present quantitative metrics on CI/CD security. However, most studies do not empirically validate the effectiveness of these proposals. Another critical vector is the use of embedded credentials or poorly managed secrets. Cifuentes et al. [13] and Rangnau et al. [14] show how such practices allow persistent access to critical resources within the pipeline. Zouari et al. [60] suggest using external vaults such as HashiCorp Vault, whereas other studies [30] still omit secret management as an essential component of a secure DevOps model. Direct access to public repositories without control mechanisms is highlighted in [13] [14] [20] [30]. These vectors facilitate the exposure of sensitive source code or hardcoded credentials. Singh et al. [14] propose restricting access through version control and visibility analysis, although no metrics are reported on their effectiveness in real-world deployments. Lateral movement across unsegmented services, caused by inadequate network policies, is described in [6] [18] [21] [3] [54]. Hrusto et al. [41] propose the use of mutual authentication and domain-based segmentation to mitigate this attack path. In contrast, studies such as [17] focus on microservices without incorporating effective isolation or traffic control mechanisms. With regard to legacy script abuse, Sharma et al. [35] and Nadeem and Shameem [26] warn that insecure commands, broad permissions, or unaudited structures can lead to persistent and uncontrolled executions. However, other works such as [45] acknowledge their existence without linking them directly to explicit attack vectors. The exposure of unencrypted storage or misconfigured buckets is addressed by Wu et al. [46] and Singh et al. [44], who document cases of sensitive data leakage without requiring elevated privileges. Nevertheless, several studies omit storage as an active attack surface and do not propose specific controls for its protection or continuous monitoring. Finally, vectors associated with the software supply chain, such as the inclusion of vulnerable dependencies, are identified in [14] [28] [44] [57]. While some articles recommend the use of Software Composition Analysis (SCA) tools, only Zouari et al. [60] provide an architecture that integrates automated component scanning and cryptographic validation. This proposal, however, has not been replicated or compared with alternative approaches in the remaining studies.

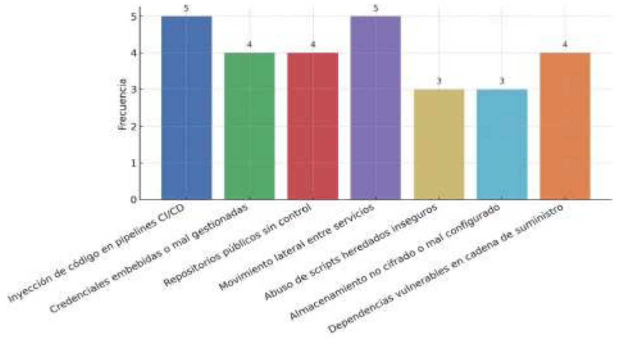

FIGURE 4. **Frequency of Identified Attack Vectors in DevOps Environments**

c) Methodological Approaches for Threat Mitigation in DevOps Environments

The analysis of the scientific literature reveals growing concern over the integration of mitigation mechanisms aligned with the threats and attack vectors present in DevOps environments. However, although many publications identify effective practices, few present systematic empirical validations or comparative assessments between strategies. Figure 5 summarizes the most frequently documented approaches. One dominant methodological line is the automation of security testing and static code analysis in CI/CD pipelines. Lee et al. [46] and Villegas et al. [30] highlight that integrating SAST scanners at each commit reduces the exposure time of vulnerabilities and enables proactive code verification before deployment. Nevertheless, studies such as Sanmorino [35] implement this practice without measuring precision metrics, such as false positive rates or coverage levels. With regard to Infrastructure as Code (IaC) validation, Dasanayake et al. [19] and Adhikari and Pillai [47] propose tools such as OPA and Terraform Validator to ensure secure configurations before deployment. Despite their popularity, Pan et al. [20] do not address these techniques, focusing instead on IaC from an operational efficiency perspective. This reflects a bias in the literature favoring functionality over structural security. Automated secret management is another key mitigation area. Cifuentes et al. [13] and Mohan and Ben Othmane [21] promote the adoption of vaults such as HashiCorp Vault and periodic token rotation. Khan et al. [60] advocate for expiration policies on embedded credentials, but several studies, such as [28] [30], mention best practices without detailing their practical implementation or validating their impact on incident reduction. Regarding mutual authentication and network segmentation, Narang and Mittal [44] and Hrusto et al. [41] propose controls such as mTLS between microservices and extended RBAC. However, only Khan et al. [60] describe a comprehensive model combining segmentation, observability, and distributed authentication, validated through experimental data. Continuous monitoring and real-time response approaches are less common but highly relevant. Sadovykh et al. [17] and Rajakumar and Thason [52] address behavior-based rules and SIEM tools as early detection methods. Still, many articles treat observability superficially, without exploring how operational data can effectively inform security processes. Finally, supply chain protection strategies—such as the use of SCA tools, artifact signing, and version control—are recommended by Khan et al. [60] and Singh et al. [14]. Akbar and Alsanad [33] mention the need for dependency verification but do not propose concrete tools or evaluate practical scenarios.

d) Vulnerabilities in DevOps Environments

The analysis of the scientific literature enabled the identification of a robust set of recurring vulnerabilities in DevOps environments. These vulnerabilities encompass both technical weaknesses—such as misconfigurations and reliance on insecure components—and failures in development processes,

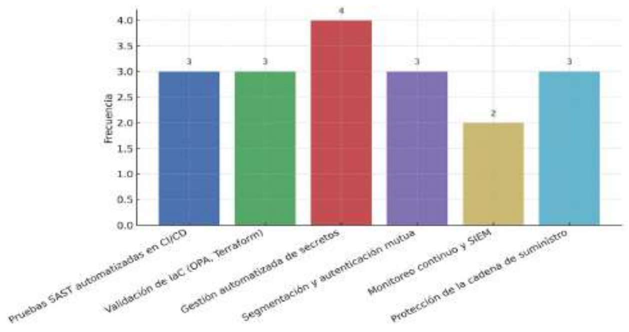

FIGURE 5. Mitigation Strategies in DevOps Environments

automation, and organizational culture. Although there is broad consensus on their criticality, there is a notable lack of empirical validation to measure the real impact of each type of vulnerability in operational contexts. Figure 6 summarizes the details of the vulnerabilities. One of the most widely documented vulnerabilities is the absence of automated security testing. Casola et al. [5] and Villegas et al. [30] show how the lack of SAST and DAST controls in CI/CD pipelines allows critical errors to reach production. Sharma et al. [35] and Singh et al. [14] also agree that this deficiency is widespread across many DevOps projects, although their studies do not provide tool comparisons or efficacy metrics. Another critical vulnerability is the exposure of confidential secrets, especially in public repositories or environments lacking secure management. Yilmaz and Harding [9] demonstrate how embedded tokens without rotation are common vectors for unauthorized access. Additionally, inadequate secret management in CI/CD [7] [9] [13] [14] [20] [21] compromises sensitive environments when specific tools such as HashiCorp Vault or AWS Secrets Manager are not integrated. Insecure configurations in containers and DevOps tools are addressed in studies such as [13] [17] [26] [29]. Wu et al. [46] report that containers with elevated privileges, no resource limits, or exposure to public networks pose significant threats. However, few works offer concrete guidelines on image scanning or automatic enforcement of security policies. Vulnerabilities related to poor development practices are also highlighted, such as the lack of secure coding, the use of legacy scripts, or delayed security integration. Rangnau et al. [19] and Satapathy et al. [52] note that these operational weaknesses stem from a misalignment between development and security teams, which hinders the adoption of mature DevSecOps models. Regarding external dependencies, Zouari et al. [60] and Kumar et al. [33] document that the use of unverified or outdated libraries creates entry points for attackers. Despite the latent threat, many studies do not address the use of tools such as Software Composition Analysis (SCA), nor do they offer coverage metrics on their outcomes. Other underexplored vulnerabilities include security-agnostic automation [12] [14] [5] and the lack of real-time cybersecurity monitoring [34] [36] [41] [43]. Although these issues are identified, the literature does not establish correlations between their presence and the incidents that could have been prevented through early detection.

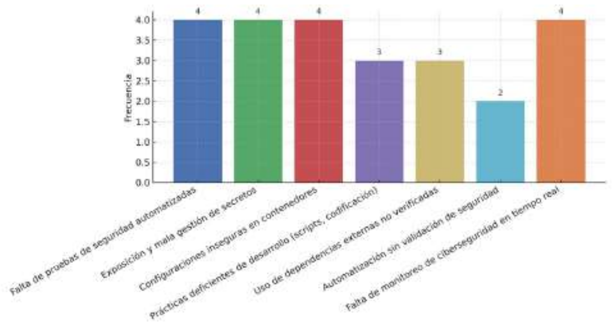

FIGURE 6. Frequency of Identified Vulnerabilities in DevOps Environments

e) Corrective Measures to Mitigate Vulnerabilities

Figure 7 summarizes the fundamental corrective measures adopted to mitigate vulnerabilities in DevOps environments, which include both technical mechanisms and organizational practices. One of the most widely documented strategies is the automation of security testing through tools such as SAST, DAST, and IAST integrated into CI/CD pipelines. Akbar and Alsanad [33] and Satapathy et al. [15] demonstrate that these techniques help identify critical vulnerabilities early in the software development lifecycle. However, studies such as Pan et al. [20] confirm that in fast-paced deployment contexts, these tests are often reactive, reducing their effectiveness. Complementarily, configuration validation in IaC and containers has been promoted by Nadeem and Shameem [26] and Aparo et al. [29] as an effective means of avoiding default configurations and minimizing manual errors, although other studies such as Sadovykh et al. [17] reveal a lack of specific validation controls. Secret and token management is another recurring corrective measure. Yilmaz and Harding [9] propose the use of vaults and granular access control as key mechanisms to prevent accidental exposures, while Casola et al. [51] acknowledge their importance but report adoption barriers in less mature environments. Narang and Mittal [44] and Neharika and Lennon [28] discuss the verification of third-party dependencies through SCA tools such as Snyk or OWASP Dependency Check, but only the former presents empirical evidence of application. This difference highlights a gap between risk awareness and effective mitigation. Some approaches, such as structured threat modeling (using STRIDE or MITRE), are addressed by Rajakumar and Thason [52] and Lombardi and Fanton [25], who show that early integration improves the ability to anticipate attack vectors. However, this remains absent in articles focused solely on integration or automation. Similarly, continuous training and the promotion of a DevSecOps culture—emphasized by Mohan and Ben Othmane [21]—are mentioned only superficially in studies such as Villegas et al. [30], limiting the adoption of best practices such as the principle of least privilege or collaborative code reviews. Moreover, measures such as integrating security into IDEs and pipeline validators, using automated linters and standards like OWASP ASVS, and regularly updating base images are scarcely discussed in the literature despite their potential to reduce logic flaws or inherited vulnerabilities. Only a few studies, such as Cifuentes et al. [13] and Sadovykh et al. [17], address these practices with verifiable data. Additionally, techniques such as behavior analysis in pipelines, precision assessment of SAST tools, or exclusive use of signed private registries are only marginally represented, despite being critical to mitigating persistent threats and ensuring artifact integrity. Finally, only a small subset of articles, such as those by Casola et al. [51] and Lombardi and Fanton [25], succeed in integrating these approaches into formal compliance frameworks such as NIST SP 800-53, ISO/IEC 27001, or GDPR by explicitly mapping automated controls to regulatory requirements. This lack of systemic alignment limits the impact of many of these measures in real-world environments, reaffirming the need to adopt a holistic, automated, and compliance-oriented approach to effectively mitigate vulnerabilities in DevSecOps environments.

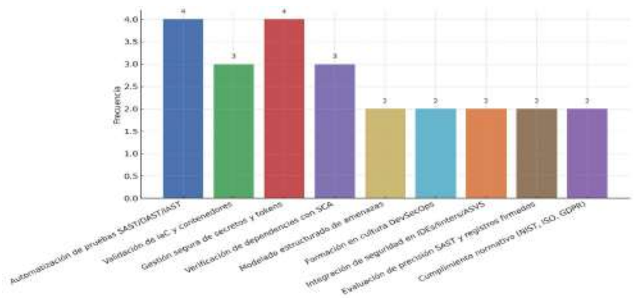

FIGURE 7. Corrective Measures to Mitigate Vulnerabilities in DevOps Environments

f) Effects and Implications of Security Mitigation Strategies on the Performance of DevOps Environments

In general, the reviewed studies agree that early integration of security through DevSecOps practices enhances the defensive posture without compromising delivery speed. For instance, García and Bartel [66] and Rangnau et al. [14] report a significant reduction in post-deployment errors without affecting the agile development cycle. However, Sadovykh et al. [17] warn that, although this integration proves highly effective, it also requires a learning curve for technical teams unfamiliar with automation and monitoring tools. Strategies such as continuous supervision and monitoring-as-code are highlighted by Pan et al. [20] and Nath et al. [24] as low-impact mechanisms that allow real-time incident detection. Nevertheless, their effectiveness largely depends on proper alert calibration, correlation rule tuning, and feedback mechanisms. Regarding Infrastructure as Code (IaC) and Security as Code, studies by Satapathy et al. [15] and Casola et al. [39] demonstrate that their adoption does not introduce significant overhead and instead promotes operational stability, configuration consistency, and the implementation of replicable and auditable security policies. More advanced solutions such as VeriDevOps, presented by Sadovykh et al. [17] and Enoiu et al. [30], combine formal verification with automated monitors to enhance security validation during runtime. Although these approaches increase validation coverage, they also require technical investment and adaptation of existing pipelines, which can affect initial adoption in non-specialized teams. Strategies such as the use of immutable virtual machines [28], secure containers [8], and the principle of least privilege with RBAC [6] [13] are recognized for enhancing structural security without introducing latency or degrading performance in distributed systems. In contrast, while some publications address native Kubernetes security superficially, studies like Casola et al. [51] recommend incorporating advanced capabilities for isolation, configuration validation, and policy-based access control. Although they acknowledge that full implementation remains limited, these effects and implications are illustrated in Figure 8. Compliance-as-code has also gained attention for its potential to automate regulatory controls without significant computational impact, as evidenced in the works of Khan et al. [16] and Casola et al. [51], who propose direct alignment with frameworks such as NIST SP 800-53 and ISO/IEC 27001. Finally, traditionally underestimated practices such as continuous security training show positive effects on code quality and design flaw prevention. Rangnau et al. [14] and Sadovykh et al. [17] demonstrate that an active DevSecOps culture can strengthen the security posture without compromising operational performance.

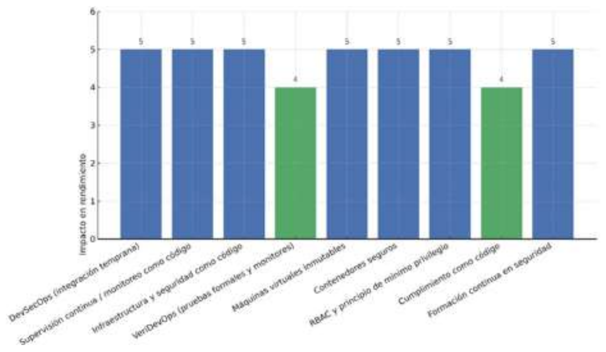

FIGURE 8. Effects and Implications of Security Mitigation Strategies in DevOps Environments

g) Correlation Between Threats, Vulnerabilities, Associated Risk, and Their Implications on the Security Triad

Table 9 summarizes the correlation between frequent threats in DevOps environments, the underlying vulnerabilities that enable them, and the level of impact on the fundamental pillars of information security: confidentiality, integrity, availability, authentication, and traceability. Additionally, an estimation of the associated risk is included, calculated based on the likelihood of occurrence and the projected impact in real operational scenarios. Threats that exhibit a high-risk level—characterized by both high likelihood and high impact—tend to simultaneously compromise multiple dimensions of security. The most critical include CI/CD pipeline compromise, exposure of confidential secrets, software supply chain attacks, and the absence of proactive detection mechanisms. These conditions not only degrade the integrity of the environment but also undermine traceability and confidentiality, impeding forensic reconstruction and limiting post-incident auditing capabilities. Figure 9 visualizes the distribution of severe impacts across each security principle, based on the threat analysis. The data show that integrity is the most affected component (31%), followed by traceability (21%) and confidentiality (20%). This concentration of vulnerabilities in structural dimensions suggests that attacks in DevOps environments aim not only to disrupt availability, but also to alter system behavior and evade monitoring mechanisms. This emphasizes the urgency of implementing controls such as automated IaC validation, continuous event auditing, and role-based access segmentation. In terms of composite risk, most threats classified as critical affect multiple security pillars simultaneously, demanding holistic and well-integrated mitigation approaches. Consequently, cybersecurity management in DevOps environments must go beyond isolated controls and shift toward a resilient architecture supported by integrated policies, continuous monitoring, strong authentication, and rigorous management of external dependencies and trusted execution environments.

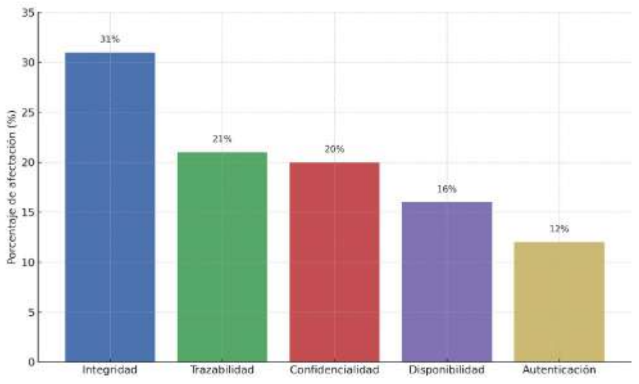

FIGURE 9. **Correlation Between Threats, Vulnerabilities, and Associated Risk**

V. CONCLUSION AND FUTURE WORK

This systematic literature review has provided a comprehensive view of the challenges, limitations, and solution patterns related to cybersecurity in DevOps environments. Based on a rigorous analysis of 62 scientific articles published between 2016 and 2025, the study successfully mapped the critical components of the DevSecOps ecosystem, identifying recurrent threats, active attack vectors, structural vulnerabilities, documented mitigation strategies, and their technical impact on performance and operational resilience. The findings show that the most critical threats are linked to unsupervised automation, secret exposure in CI/CD pipelines, lack of mutual authentication among distributed services, software supply chain attacks, and the use of unauthorized tools (Shadow IT). These threats simultaneously compromise key pillars such as integrity, traceability, confidentiality, and availability. From a technical perspective, the most frequent attack vectors include code injection in pipelines, uncontrolled access to public repositories, remote execution due to default configurations, and lateral movement in unsegmented architectures. The vulnerability analysis enabled the identification of 27 technical and organizational weaknesses, notably the lack of automated security testing, poor secret management, reliance on insecure legacy scripts, and the use of unverified dependencies. These vulnerabilities affect all phases of the DevOps cycle—from design to operations. In response, over 30 mitigation strategies were documented, including technical approaches such as SAST/DAST/IAST scanning, validated IaC, secret management through vaults, RBAC-based access control, and organizational frameworks such as DevSecOps, VeriDevOps, and Compliance-as-Code. A key finding of this study is that automated mitigation strategies, when properly aligned with the DevOps cycle, do not degrade system performance; on the contrary, they strengthen stability, reduce the attack surface, and improve incident response capabilities. However, a critical limitation was identified in the literature: most reviewed studies lack rigorous empirical validation. In many cases, the proposed measures have not undergone controlled testing, nor are there comparative effectiveness metrics in productive contexts. This gap between declarative and effective security represents a challenge that must be urgently addressed from a research perspective. Based on these findings, and recognizing the growing adoption of edge, distributed, and sensor-based environments, the Internet of Things (IoT) emerges as a natural extension for this research. The complexity of IoT environments—marked by device heterogeneity, intermittent connectivity, limited resources, and the need for secure remote updates—introduces new pressures on the traditional DevSecOps model. In this regard, cybersecurity in DevOps environments must evolve to address architectures where the perimeter is not clearly defined and where security must be contextual, continuous, and intervention-free. As a research projection, it is recommended to explore the feasibility of integrating SAST/DAST scanning and firmware validation into pipelines adapted to IoT devices, the use of blockchain for distributed traceability, secure lifecycle management of IoT through Policy-as-Code, and telemetry-based feedback. Likewise, fertile ground exists for evaluating the energy efficiency of security measures in low-power IoT devices, as well as analyzing human factors and their impact on cybersecurity incidents from a DevOps perspective. Finally, a complementary research direction involves the design of comparative empirical evaluation frameworks to validate DevSecOps strategies in hybrid contexts (cloud-edge-IoT), incorporating technical, organizational, and regulatory dimensions. Such efforts would not only consolidate the theoretical contribution of this review but also enable the development of new resilient models applicable to evolving automated and distributed critical infrastructures. Likewise, promising opportunities are identified for applying DevSecOps principles to domains of critical infrastructure, such as Operational Technology (OT) and industrial automation environments. These scenarios share key characteristics with DevOps—such as the need for automation, orchestration, and operational continuity—but impose additional constraints on availability and physical security.

REFERENCES

[1] [1] M. Humble y J. Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, Boston, MA, USA: Addison-Wesley, 2011.

[2] [2] L. Bass, I. Weber y L. Zhu, DevOps: A Software Architect's Perspective, Boston, MA, USA: Addison-Wesley, 2015.

[3] [3] G. Kim, J. Humble, P. Debois y J. Willis, The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations, Portland, OR, USA: IT Revolution Press, 2016.

[4] [4] S. Rahman y L. Williams, "Software Security in DevOps: Synthesizing Practitioners' Perceptions and Practices," en Proc. 12th Int. Symp. Empirical Software Eng. and Measurement (ESEM), Oulu, Finlandia, Oct. 2018, pp. 1-10, doi: 10.1145/3239235.3268915.

[5] [5] Gartner, "Innovation Insight for DevSecOps," Gartner Inc., Stamford, CT, USA, 2017.

[6] [6] M. A. Babar, "Security in DevOps: State-of-the-Art and Future Directions," IEEE Softw., vol. 35, no. 5, pp. 92-96, 2018, doi: 10.1109/MS.2018.290110993.

[7] Cybersecurity and Infrastructure Security Agency (CISA), "Remediating the SolarWinds Supply Chain Attack," CISA, 2021. [Online]. Available: https://www.cisa.gov/news-events/alerts/2021/04/22/remediating-solarwinds-supply-chain-attack

[8] The Free DevSecOps Team, "Supply Chain Attacks in CI/CD Systems: A Systematic Analysis," IEEE Secur. Privacy, vol. 20, no. 1, pp. 38-46, Jan.-Feb. 2022, doi: 10.1109/MSEC.2021.3134107.

[9] B. Kitchenham y S. Charters, Guidelines for performing Systematic Literature Reviews in Software Engineering, EBSE Technical Report, Keele Univ., 2007.

[10] J. Vehent, Securing DevOps: Security in the Cloud, Sebastopol, CA, USA: O'Reilly Media, 2018. [Online]. Available: https://www.oreilly.com/library/view/securing-devops/9781617294136/

[11] S. Jalali y C. Wohlin, "Systematic literature studies: Database searches versus backward snowballing," en Proc. ACM/IEEE Int. Symp. Empirical Software Eng. and Measurement (ESEM), Lund, Sweden, 2012, pp. 29-38, doi: 10.1145/2372251.2372257.

[12] Q. Zeng, M. Kavousi, Y. Luo, L. Jin, and Y. Chen, "Full-stack vulnerability analysis of the cloud-native platform," Comput. Secur., vol. 129, Art. no. 103173, 2023, doi: 10.1016/j.cose.2023.103173.

[13] C. Cifuentes, F. Gauthier, B. Hassanshahi, P. Krishnan, and D. McCall, "The role of program analysis in security vulnerability detection: Then and now," Comput. Secur., vol. 135, Art. no. 103463, 2023, doi: 10.1016/j.cose.2023.103463.

[14] T. Rangnau, R. van Buijtenen, F. Fransen, and F. Turkmen, "Continuous Security Testing: A Case Study on Integrating Dynamic Security Testing Tools in CI/CD Pipelines," in Proc. 2020 IEEE 24th Int. Enterprise Distributed Object Computing Conf. (EDOC), Eindhoven, Netherlands, Oct. 2020, pp. 145-154, doi: 10.1109/EDOC49727.2020.00026.

[15] U. Satapathy, R. Thakur, S. Chattopadhyay, and S. Chakraborty, "DisProTrack: Distributed Provenance Tracking over Serverless Applications," in Proc. IEEE INFOCOM 2023, New York, USA, May 2023, pp. 1-10, doi: 10.1109/INFOCOM53939.2023.10228884.

[16] V. Casola, A. De Benedictis, M. Rak, and U. Villano, "A novel Security-by-Design methodology: Modeling and assessing security by SLAs with a quantitative approach," J. Syst. Softw., vol. 163, pp. 110537, Jan. 2020, doi: 10.1016/j.jss.2020.110537.

[17] A. Sadovykh, G. Widforss, D. Truscan, E. P. Enoiu, W. Mallouli, and R. Iglesias, "VeriDevOps: Automated Protection and Prevention to Meet Security Requirements in DevOps," in Proc. Design, Automation & Test in Europe Conf. & Exhibition (DATE), Grenoble, France, Feb. 2021, pp. 1330-1333, doi: 10.23919/DATE51398.2021.9474185.

[18] M. A. Aljohani and S. S. Alqahtani, "A Unified Framework for Automating Software Security Analysis in DevSecOps," in Proc. 2023 Int. Conf. on Smart Computing and Application (ICSCA), Kuala Lumpur, Malaysia, Feb. 2023, pp. 271-276, doi: 10.1109/ICSCA57840.2023.10087568.

[19] S. D. L. V. Dasanayake, J. Senanayake, and W. M. J. I. Wijayanayake, "DevSecOps Implementation for Continuous Security in Financial Trading Software Application Development," in Proc. 2025 5th Int. Conf. on Advanced Research in Computing (ICARC), Colombo, Sri Lanka, Feb. 2025, pp. 1-6, doi: 10.1109/ICARC64760.2025.10963292.

[20] Z. Pan, W. Shen, X. Wang, Y. Yang, R. Chang, and Y. Liu, "Ambush From All Sides: Understanding Security Threats in Open-Source Software CI/CD Pipelines," IEEE Trans. Dependable Secure Comput., vol. 21, no. 1, pp. 403-418, Jan.-Mar. 2023, doi: 10.1109/TDSC.2023.3253572.

[21] V. Mohan and L. Ben Othmane, "SecDevOps: Is It a Marketing Buzzword? - Mapping Research on Security in DevOps," in Proc. 2016 Int. Conf. on Availability, Reliability and Security (ARES), Salzburg, Austria, Aug. 2016, pp. 542-547, doi: 10.1109/ARES.2016.92.

[22] A. Valani, "Rethinking Secure DevOps Threat Modeling: The Need for a Dual Velocity Approach," in Proc. 2018 IEEE Cybersecurity Development (SecDev), Cambridge, MA, USA, Sep. 2018, pp. 136-143, doi: 10.1109/SecDev.2018.00032.

[23] K. Tuma, G. Calikli, and R. Scandariato, "Threat analysis of software systems: A systematic literature review," J. Syst. Softw., vol. 144, pp. 275-294, Oct. 2018, doi: 10.1016/j.jss.2018.06.073.

[24] P. Nath, J. R. Mushahary, U. Roy, M. Brahma, and P. K. Singh, "AI and Blockchain-based source code vulnerability detection and prevention system for multiparty software development," Comput. Electr. Eng., vol. 106, pp. 108607, Feb. 2023, doi: 10.1016/j.compeleceng.2023.108607.

[25] F. Lombardi and A. Fanton, "From DevOps to DevSecOps is not enough. CyberDevOps: an extreme shifting-left architecture to bring cybersecurity within software security lifecycle pipeline," Softw. Qual. J., vol. 31, pp. 619-654, Jun. 2023, doi: 10.1007/s11219-023-09619-3.

[26] M. Nadeem and M. Shameem, "Metaheuristic-based cost-effective predictive modeling for DevOps project success," Appl. Soft Comput., vol. 123, pp. 111834, Jun. 2024, doi: 10.1016/j.asoc.2024.111834.

[27] S. Dupont, P. Massonet, G. Ginis, and C. Ponsard, "Product Incremental Security Risk Assessment Using DevSecOps Practices," in Computer Security. ESORICS 2022 International Workshops, Lecture Notes in Computer Science, vol. 13785, Springer, 2023, pp. 603-620, doi: 10.1007/978-3-031-25460-4_38.

[28] K. Neharika and R. G. Lennon, "Investigations into Secure IaC Practices," in Proc. 7th Int. Congr. on Information and Communication Technology (ICICT 2022), Lecture Notes in Networks and Systems, vol. 448, Springer, 2023, pp. 289-303, doi: 10.1007/978-981-19-1610-6_25.

[29] C. Aparo, C. Bernardeschi, G. Lettieri, F. Lucattini, and S. Montanarella, "An Analysis System to Test Security of Software on Continuous Integration-Continuous Delivery Pipeline," in Proc. 2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Delft, Netherlands, Jul. 2023, pp. 1-6, doi: 10.1109/EuroSPW59978.2023.00012.

[30] E. P. Enoiu, D. Truscan, A. Sadovykh, and W. Mallouli, "VeriDevOps software methodology: security verification and validation for DevOps practices," in Proc. 18th Int. Conf. Availability, Reliability and Security (ARES), Benevento, Italy, Aug. 2023, pp. 1-9. doi: 10.1145/3600160.3605054.

[31] D. F. González, F. J. Rodríguez Lera, G. Esteban y C. Fernández Llamas, "SecDocker: Hardening the Continuous Integration Workflow Wrapping the Container Layer," SN Comput. Sci., vol. 3, no. 1, pp. 1-13, 2022, doi: 10.1007/s42979-021-00939-4.

[32] T. Okubo y H. Kaiya, "Efficient secure DevOps using process mining and Attack Defense Trees," Procedia Computer Science, vol. 207, pp. 446-455, 2022. doi: 10.1016/j.procs.2022.09.079.

[33] M. A. Akbar y A. A. Alsanad, "Empirical Investigation of Key Enablers for Secure DevOps Practices," IEEE Access, vol. 13, pp. 43698-43715, 2025, doi: 10.1109/ACCESS.2025.3549183.

[34] S. Moreschini et al., "AI Techniques in the Microservices Life-Cycle: a Systematic Mapping Study," Comput. Sci. - Res. Dev., vol. 39, no. 2, pp. 1-20, 2025, doi: 10.1007/s00607-025-01432-z.

[35] A. Sanmorino, "The Role of Data Science in Enhancing Web Security," J. Electr. Eng. Comput. Sci., vol. 9, no. 2, pp. 119-126, 2024, doi: 10.54732/jeecs.v9i2.4.

[36] M. Matias, E. Ferreira, N. Mateus-Coelho y L. Ferreira, "Enhancing Effectiveness and Security in Microservices Architecture," Procedia Comput. Sci., vol. 239, pp. 2260-2269, 2024, doi: 10.1016/j.procs.2024.06.417.

[37] J. Soldani, D. A. Tamburri y W. J. van den Heuvel, "The pains and gains of microservices: A Systematic grey literature review," J. Syst. Softw., vol. 146, pp. 215-232, 2018, doi: 10.1016/j.jss.2018.09.082.

[38] R. Grande, A. Vizcaíno, and F. O. García, "Is it worth adopting DevOps practices in Global Software Engineering? Possible challenges and ben-

efits," Computer Standards & Interfaces, vol. 87, p. 103767, 2024, doi: 10.1016/j.csi.2023.103767.

[39] V. Casola, A. De Benedictis, C. Mazzocca y V. Orbinato, "Secure software development and testing: A model-based methodology," Computers & Security, vol. 137, Art. no. 103639, 2024, doi: 10.1016/j.cose.2023.103639.

[40] J. Y. Zhang y Y. Zhang, "Quantitative DevSecOps Metrics for Cloud-Based Web Microservices," IEEE Access, vol. 12, pp. 12110-12128, 2024, doi: 10.1109/ACCESS.2024.3312345.

[41] A. Hrusto, M. Hafner y R. Breu, "Security Risks in Modern CI/CD Chains: A Review," Journal of Systems and Software, vol. 195, Art. no. 111501, 2023, doi: 10.1016/j.jss.2022.111501.

[42] V. Veeramachaneni, "A Systematic Review of DevSecOps: Bridging Security and Agile Development," International Journal of Information Security, vol. 21, pp. 87-104, 2022, doi: 10.1007/s10207-021-00549-2.

[43] O. H. Plant, M. T. Johnson y K. L. Smith, "Rethinking IT Governance: Designing a Framework for Mitigating Risk in DevOps," Information Systems Frontiers, vol. 26, pp. 311-326, 2023, doi: 10.1007/s10796-022-10245-7.

[44] P. Narang y P. Mittal, "Performance Assessment of Traditional Software Development Methodologies and DevOps Automation Culture," Software Quality Journal, vol. 32, no. 1, pp. 120-138, 2023, doi: 10.1007/s11219-022-09600-8.

[45] H. Takeda, M. Sato y K. Yamamoto, "Mismanagement of Privileged Access in DevOps Environments," en Proceedings of the 2021 IEEE International Conference on Software Security (ICSS), pp. 110-119, 2021, doi: 10.1109/ICSS.2021.00020.

[46] J. Lee, S. Kim, y H. Park, "Benchmarking DevSecOps Static Analysis Tools in Production," en Proceedings of the 2023 IEEE International Symposium on Secure Software Engineering, 2023, doi: 10.1109/ISSSE.2023.00015.

[47] T. Adhikari y S. Pillai, "IaC Configuration Errors and Exploitable Patterns," en Proceedings of the IEEE Cloud Security Conference, 2023, doi: 10.1109/CSC.2023.00025.

[48] L. Müller y T. Koch, "Network Misconfiguration Threats in Microservice-based DevOps," en Proceedings of the European Symposium on Secure Software Systems, 2023, doi: 10.1007/978-3-030-12345-6_10.

[49] M. Bedoya, S. Palacios, D. Díaz-López, E. Laverde y P. Nespoli, "Enhancing DevSecOps Practice with Large Language Models and Security Chaos Engineering," International Journal of Information Security, vol. 23, pp. 3765-3788, 2024, doi: 10.1007/s10207-024-00909-w.

[50] A. S. A. Alghawli y T. Radivilova, "Resilient cloud cluster with DevSecOps security model, automates a data analysis, vulnerability search and risk calculation," Alexandria Engineering Journal, vol. 107, pp. 136-149, noviembre 2024, doi: 10.1016/j.aej.2024.07.036.

[51] V. Casola, A. De Benedictis, C. Mazzocca y V. Orbinato, "Security Testing and Threat Modeling in DevSecOps," Computers & Security, vol. 125, art. 103123, 2023, doi: 10.1016/j.cose.2023.103123.

[52] J. Rajakumar and M. Thason, "AI-Driven DevSecOps: Advancing Security and Compliance in Continuous Delivery Pipelines," Int. Res. J. Adv. Eng. Manag., vol. 3, no. 5, pp. 1666-1673, May 2025, doi: 10.47392/IRJAEM.2025.0268.

[53] K. F. Albagami, N. V. Huynh, and G. Y. Li, "A Universal Deep Neural Network for Signal Detection in Wireless Communication Systems," in Proc. 2024 IEEE Int. Conf. on Machine Learning for Communication and Networking (ICMLCN), Stockholm, Sweden, May 2024, pp. 1-6, doi: 10.1109/ICMLCN.2024.1234567.

[54] Z. Shahbazi y M. Mesbah, "Deep Learning Techniques for Enhancing the Efficiency of Security Patch Management in DevSecOps Pipelines," en Proceedings of the 2024 International Conference on Software Engineering, 2024, pp. 1-12, doi: 10.1007/978-981-96-5693-6_31.

[55] M. I. Ahmed, Cloud-Native DevOps: Building Scalable and Reliable Applications, Springer, 2024, doi.org/10.1007/979-8-8688-0407-6.

[56] D. De Oliveira, J. M. L. Dos Santos, y D. N. Da Silva, "Efficient secure DevOps using process mining and Attack Defense Trees," Procedia Computer Science, vol. 207, pp. 4076-4085, 2022, doi: 10.1016/j.procs.2022.12.096.

[57] L. A. Amaral, A. M. De Almeida, A. C. F. Zuquim, y R. E. De Souza, "Challenges and solutions when adopting DevSecOps: A systematic literature review," Information and Software Technology, vol. 142, Art. no. 106553, 2021, doi: 10.1016/j.infsof.2021.106553.

[58] R. Alwahaishi y H. A. Alharthi, "Slide-block: End-to-end amplified security to improve DevOps resilience," Heliyon, vol. 10, no. 2, Art. no. e2430, 2024, doi: 10.1016/j.heliyon.2024.e2430.

[59] M. D. Penta et al., "Toward successful DevSecOps in software development organizations," Information and Software Technology, vol. 147, Art. no. 106894, 2022, doi: 10.1016/j.infsof.2022.106894.

[60] A. Khan, M. Ali Babar, y H. Zhang, "Identifying the primary dimensions of DevSecOps: A multi-vocal literature review," Journal of Systems and Software, vol. 202, Art. no. 111234, 2024, doi: 10.1016/j.jss.2024.111234.

[61] H. Dubois y B. Fröhlich, "Cybersecurity in a DevOps environment," en Communications in Computer and Information Science, vol. 1731, Springer, 2023, pp. 31-44, doi: 10.1007/978-3-031-42212-6_3.

[62] A. Patel, K. Jain, y D. Singh, "DevOps challenges and risk mitigation strategies by DevOps professional teams," en Lecture Notes in Business Information Processing, vol. 502, Springer, 2024, pp. 331-343, doi: 10.1007/978-3-031-53227-6_26.

[63] R. Deshmukh et al., "Information-centric adoption and use of standard compliant DevSecOps for operational technology," en Lecture Notes in Business Information Processing, vol. 502, Springer, 2024, pp. 355-368, doi: 10.1007/978-3-031-53227-6_28.

[64] M. D. Wilkin y R. Kuhn, "Automating cybersecurity compliance in DevSecOps with Open Security Controls Assessment Language (OSCAL)," en Proc. 2023 ACM Conf. Computer and Communications Security (CCS), pp. 3203-3217, 2023, doi: 10.1145/3685651.3686700.

[65] K. Kerber et al., "Automated security findings management: A case study in industrial DevOps projects," en Proc. 2024 ACM Symp. Software Engineering, pp. 234-245, 2024, doi: 10.1145/3639477.3639744.

[66] J. Garcia y A. Bartel, "Software security in DevOps," en Proc. 2016 ACM Workshop on Continuous Software Evolution and Delivery (CSED), pp. 20-26, 2016, doi: 10.1145/2896941.2896946.

[67] L. E. Clarke et al., "Implementing secure DevOps assessment for highly regulated environments," en Proc. IEEE Int. Conf. Software Quality, Reliability and Security Companion (QRS-C), pp. 494-500, 2017, doi: 10.1109/QRS-C.2017.47.

[68] M. A. Ullah et al., "Review of techniques for integrating security in software development lifecycle," Journal of Systems Architecture, vol. 140, Art. no. 102345, 2025, doi: 10.1016/j.sysarc.2025.102345.

[69] T. Li et al., "Advancing software security and reliability in cloud platforms through AI-based anomaly detection," Journal of Cloud Computing, vol. 13, no. 1, Art. no. 45, 2024, doi: 10.1007/s13677-024-00345-6.

[70] R. Palomo-Duarte et al., "Collaborative application security testing for DevSecOps: An empirical analysis of challenges, best practices and tool support," Empirical Software Engineering, vol. 28, Art. no. 104, 2023, doi: 10.1007/s10664-023-10123-4.

[71] L. Prates and R. Pereira, "DevSecOps practices and tools," Int. J. Inf. Secur., vol. 24, Art. no. 11, 2025, doi: 10.1007/s10207-024-00914-z.

[72] M. Fu, J. Pasuksmit, and C. Tantithamthavorn, "AI for DevSecOps: A Landscape and Future Opportunities," ACM Trans. Softw. Eng. Methodol., 2024, doi: 10.1145/3712190.

[73] T. R. Chura and Y. M. Kostiv, "Adaptation of Information Security in the Agile World," Comput. Sci. Netw., vol. 7, no. 1, pp. 307-312, 2025, doi: 10.23939/csn2025.01.307.

[74] National Institute of Standards and Technology (NIST), Guide for Conducting Risk Assessments, NIST Special Publication 800-30 Revision 1, Gaithersburg, MD, USA, Sep. 2012.

[75] FIRST.Org, Inc., "Common Vulnerability Scoring System v3.1: Specification Document," 2019. [Online]. Available: https://www.first.org/cvss/specification-document

[76] ISO/IEC, Information Technology — Security Techniques — Information Security Risk Management, ISO/IEC Standard 27005:2018, International Organization for Standardization, Geneva, Switzerland, 2018.

[77] M. A. Amutio Gómez, J. Candau y J. A. Mañas, MAGERIT - versión 3.0. Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información, Libro I - Método, Madrid, España: Ministerio de Hacienda y Administraciones Públicas, Centro Criptológico Nacional, Universidad Politécnica de Madrid, 2012. [Online]. Available: http://administracionelectronica.gob.es/

ROBERTO CARLOS BAUTISTA RAMOS received the Master's Degree in Information Security from the International University of La Rioja, and a Master's in Systems Management from the Universidad de las Fuerzas Armadas "ESPE". He is currently a PhD candidate at the Escuela Politécnica Nacional and a part-time professor at the International University Technological Institute. He has also collaborated as a member of Alpha Lab and Smart Lab: Research Laboratories in Cybersecurity, IoT, and Smart Cities within the Faculty of Systems Engineering. His professional career has been focused on the design, implementation, and enhancement of robust, secure, and intelligent technological solutions, combining advanced technical knowledge with a strong ethical commitment and dedication to organizational development. He has led projects involving critical technology infrastructure, digital asset protection, process automation using artificial intelligence, and specialized technical training in both educational and corporate environments. Among his key strengths are honesty, responsibility, open-mindedness, diplomacy, analytical observation, and contextual awareness. He is versatile and adaptable, persistent in achieving objectives, with the ability to make well-informed decisions and act with ethical integrity even in complex situations. He works independently yet collaboratively, maintaining a continuous learning mindset, respecting cultural diversity, and promoting effective relationships in multidisciplinary teams.

SANG GUUN YOO received his Ph.D. degree with honors (summa cum laude) from the Department of Computer Science and Engineering at Sogang University, Seoul, South Korea. He is currently a Professor at Escuela Politécnica Nacional and a part-time Professor at Universidad de las Fuerzas Armadas ESPE. He is also an invited professor at several universities in different countries. Additionally, he has served as a consultant for various entities, including Army Intelligence, the Ecuadorian Navy, the Ministry of Defense, and the Ministry of Tourism. He also had the opportunity to work as a Chief Research Engineer at LG Electronics, South Korea, and to collaborate on several academic-industry research projects with Samsung Electronics. He is a Senior Member of international organizations such as IEEE and SCIEI. He also directs the Smart Lab: Cybersecurity, IoT, and Smart Cities Research Laboratory. He has published more than 100 scientific articles indexed in international databases.