International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 7.101

Evaluating the Effectiveness of the Shift-Left Strategy for Reducing Information Security Risks in the Software Life Cycle

Romm Nikita

Principal DevOps Engineer, Tel Aviv, Israel
Email: *nikitaromm[at]gmail.com*

Abstract: ***The article examines the effectiveness of the shift-left strategy for managing information security risks across the software life cycle in the context of DevSecOps practices. The study systematises recent publications on security integration into CI/CD pipelines, security-as-code and automation of SAST, DAST, SCA and IaC-scanning within continuous delivery. The work describes how contemporary approaches distribute security controls across life-cycle phases, analyses their influence on vulnerability detection timing, reduction of post-release defects and compliance with regulatory requirements. Special attention is paid to architectural models that treat security checks as programmable, version-controlled artefacts embedded into CI/CD toolchains. The goal of the article is to build an analytical framework for evaluating shift-left efficiency in real corporate environments. The framework relies on comparative analysis of recent studies and on the author s earlier monograph devoted to automation and protection of DevOps processes in corporate CI/CD chains. The article will be useful for researchers and practitioners designing secure SDLCs, corporate CISOs and DevSecOps engineers who plan to strengthen early-phase security controls without sacrificing delivery speed.***

Keywords: Shift-left security, DevSecOps, CI/CD pipeline, software life cycle, information security risk, security as code, automated security testing, SAST/DAST, cloud-native applications, secure SDLC.

1. Introduction

The evolution of DevOps has led to dense automation of software delivery chains, where deployment frequency and rapid feedback dominate engineering priorities. In such environments traditional perimeter-oriented security, concentrated near production release, fails to detect vulnerabilities early enough and leaves organisations exposed to defects discovered only after deployment. Recent DevSecOps literature describes a transition to early integration of security checks into development workflows, often denoted as shift-left security, where vulnerability detection and configuration control move into planning, coding and build phases of the software life cycle.

Within this trend security is treated as programmable infrastructure: policies, scanning configurations and compliance rules reside in version control and are executed automatically as part of CI/CD pipelines. Such interpretation enables continuous security testing and reduces manual audit overhead, but raises questions about the real extent of risk reduction achieved compared with more conservative, audit-centric models. Systematic reviews of DevSecOps underline the lack of unified evaluation criteria and heterogeneous reporting of outcomes, which complicates quantitative assessment of shift-left practices in industrial settings.

Earlier research consolidated in the monograph on automation and securing DevOps processes in corporate CI/CD environments fixed a methodological baseline: classification of DevOps toolchains, identification of typical bottlenecks in release stability and discussion of integrated security controls for corporate pipelines. On this foundation the present article concentrates specifically on the efficiency of shift-left strategy as a risk-management instrument throughout the software life cycle.

The goal of the study is to construct an analytical model for evaluating the influence of shift-left DevSecOps practices on information security risk in software projects. To reach this goal three research tasks are formulated:

1) Systematise interpretations of shift-left security in current DevSecOps publications and map them to SDLC phases.
2) Compare proposed technical and organisational mechanisms (toolchains, automation patterns, metrics) in terms of their stated effect on vulnerability discovery, remediation effort and residual risk.
3) Synthesise an integrated evaluation framework aligned with corporate software development conditions, suitable for subsequent empirical validation on real CI/CD pipelines.

Novelty lies in combining recent heterogeneous DevSecOps studies into a single risk-oriented evaluation model, explicitly bound to shift-left decisions along the life cycle rather than to generic security maturity levels.

2. Materials and Methods

The study relies on ten scientific publications from 2020–2025 that examine DevSecOps and shift-left security in different segments of the software life cycle. D. R. Chittibala [1] discusses integration of security practices into the DevOps pipeline and describes Security-as-Code and automated controls in CI/CD. R. Dachepally [2] focuses on shift-left security through automated scanning tools (SAST, DAST, SCA and IaC scanners) and their influence on vulnerability detection timing. L. Gudala, S. G. R. Bojja, V. R. R. Alluri and T. Ahmad [3] introduce a cloud-native security framework unifying continuous security testing, container security and automated compliance checks. R. Manchana [4]

analyses DevSecOps for cloud-native cybersecurity, contrasting early security with continuous protection "to the right". C. Mavani and H. K. Mistry [5] evaluate DevSecOps practices in securing cloud-native application development within the DevOps life cycle. K. I. Mohammed, B. Shanmugam and J. El-Den [6] present a systematic literature review on DevSecOps and application security, synthesising themes of automation, tool orchestration and cultural transformation. Y. Ramaswamy [7] describes a practical DevSecOps implementation with automated security scanning and reports reduction of post-deployment vulnerabilities. Raskhan [8] outlines security integration into the DevOps pipeline with emphasis on threat modelling, IaC and continuous security testing. A. K. Reddy, V. R. R. Alluri, S. Thota, C. S. Ravi and V. S. M. Bonam [9] study DevSecOps for cloud-native applications, detailing Kubernetes, Docker and Terraform controls with shift-left orientation. K. K. Voruganti [10] examines security-by-design practice with a DevSecOps shift-left approach, highlighting framework-level recommendations for early integration of security into SDLC. Based on this evidence, the present study refines the contribution of the Secured DevOps methodology proposed in the monograph [11].

The research uses qualitative comparative analysis of publications, structured along SDLC phases and classes of security controls (static and dynamic application security testing, software composition analysis, secrets management, IaC validation, policy-as-code). Narrative synthesis is applied to extract reported outcomes relevant to information security risk: reduction of post-release vulnerabilities, decrease in remediation effort, improvement of compliance and observability. Elements of conceptual modelling support construction of an integrated shift-left DevSecOps framework, combining architectural patterns, tooling and organisational practices. The earlier monographic work on automation and securing DevOps processes in corporate CI/CD chains is treated as methodological background for aligning the framework with real corporate environments rather than purely academic case studies.

3. Results

The comparative reading of sources reveals a common baseline: shift-left security is interpreted as systematic relocation of security controls from late testing and pre-release audit stages towards earlier phases of the software life cycle, with automation and Security-as-Code ensuring repeatability and integration into CI/CD pipelines. Chittibala [1] and Dachepally [2] emphasise embedding security checks directly into build and integration steps through SAST, DAST, SCA and IaC scanning, making each pipeline execution function as a security checkpoint with automated policy enforcement. Reddy and co-authors [9] and Gudala et al. [3] extend this view to cloud-native environments, where container image scanning, Kubernetes configuration checks and Terraform-based IaC validation support continuous enforcement of baseline configurations already in development and staging stages.

At the architectural level the publications converge on a layered DevSecOps structure, where code repositories, CI servers, artifact registries and deployment platforms are augmented with security integrations. Ramaswamy [7] describes a reference pipeline including automated static and dynamic analysis, secrets detection, runtime policy enforcement and IaC compliance validation, producing measurable reduction of post-deployment vulnerabilities and improved traceability of security decisions. Mavani and Mistry [5] detail similar chains in cloud-native scenarios, stressing that automation of security tasks, from container scanning to continuous monitoring, becomes the only practical way to keep up with release frequency. In both cases security checks accompany each artifact from commit to deployment, decreasing the probability that exploitable defects survive into production.

Mohammed and co-authors [6] aggregate results of earlier studies in a systematic review and identify several recurrent themes: lack of standardised DevSecOps frameworks, fragmented toolchains and organisational resistance to integrating security into developer workflows. These observations correlate with analytical works of Gudala et al. [3] and Voruganti [10], who underline that shift-left cannot be reduced to plugging scanners into CI/CD; organisations need structured patterns for distributing responsibilities between development, operations and security, orchestrating tools and defining metrics for success. In the selected publications shift-left strategy emerges as a set of coordinated decisions: where to place tests and quality gates, how to codify policies and which feedback loops to create for developers.

A synthetic view of these decisions is presented in Figure 1. The figure summarises an integrated shift-left DevSecOps model that spans the entire software life cycle. Planning and design stages incorporate threat modelling, definition of security acceptance criteria and initial compliance requirements drawn from [8; 9; 10]. In coding and build phases SAST, SCA and secrets-detection tools from [1; 2; 7] are run on every commit, with security policies expressed as code in pipeline configuration. Testing and pre-deployment stages reuse DAST, container scanning and IaC validation described in [3–5; 9]. Deployment and operation stages rely on continuous monitoring, logging and policy-enforcement mechanisms; while these controls are "to the right", their configuration and activation are prepared early using Security-as-Code patterns.

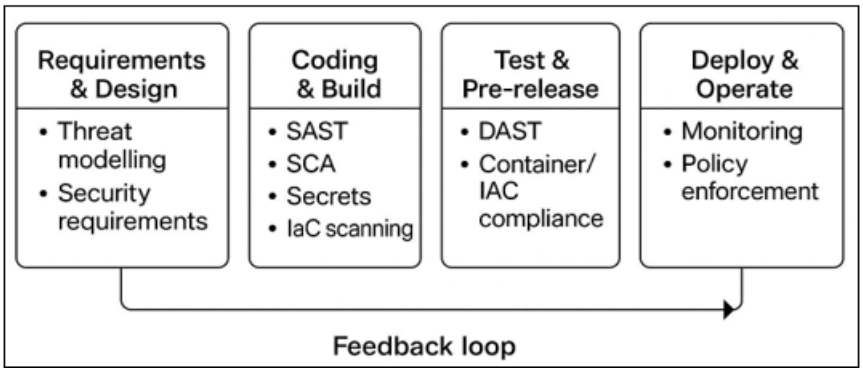

Figure 1: Integrated shift-left DevSecOps model for reducing information security risks across the software life cycle (compiled by the author based on his own research)

The model in Figure 1 highlights several channels through which shift-left influences information security risk. First, the number of life-cycle phases without explicit security controls shrinks: all sources [1–10] converge on the idea that design, coding and integration cannot remain "blind spots" handled only by downstream testing. Second, the latency between defect introduction and detection shortens because automated checks run continuously and synchronously with developer activities. For example, Dachepally [2] and Ramaswamy [7] report that pipeline-embedded scanning identifies vulnerabilities before merge or during pre-deployment tests, with subsequent decreases in post-release incidents. Third, the cost of remediation falls due to early context availability: Chittibala [1] and Mavani & Mistry [5] underline that developers fix issues faster when feedback arrives while code is fresh and the relevant context is still present in working memory.

From a risk-management perspective the reviewed publications implicitly structure the threat surface along artefact types: source code, third-party components, infrastructure definitions, configuration and runtime environment. Shift-left strategies address these surfaces unevenly. Papers [1; 2; 7; 8] concentrate mainly on source code, using SAST and coding guidelines, whereas works [3–5; 9] place greater emphasis on cloud-native assets: containers, Kubernetes manifests and IaC scripts. Mohammed et al. [6] note that comprehensive DevSecOps deployments require alignment of both directions; partial implementation leaves residual blind zones, such as unmanaged secrets, insufficient logging or unscanned infrastructure templates. The integrated model derived here suggests that genuine shift-left effectiveness depends on coverage of all artefact types, not merely on integrating a narrow selection of automated scanners early in the process.

Another dimension of analysis concerns metrics used to assess DevSecOps outcomes. Only part of the examined publications reports explicit indicators. Ramaswamy [7] describes reduced post-deployment vulnerabilities and improved mean time to remediate (MTTR) in a simulated agile team, while Mohammed et al. [6] call for more rigorous quantitative validation of DevSecOps benefits and highlight the scarcity of comparable empirical data. Mavani and Mistry [5] and Raskhan [8] discuss improvement of security posture, compliance adherence and time-to-market in qualitative terms. Across sources there is recurrent mention of key indicators such as vulnerability discovery rate, coverage of automated tests, MTTR and deployment frequency, but values and measurement procedures differ. The analytical framework proposed in the present article treats these metrics as a minimal set required for evaluation of shift-left efficiency and organises them along three groups: detection metrics (where in the life cycle defects are found), remediation metrics (how fast and at what cost they are mitigated) and exposure metrics (how many vulnerabilities reach production and how long they remain exploitable).

Finally, the synthesis of [1–10] and of the earlier monograph [11] shows that corporate environments introduce constraints absent from small-scale experimental settings. Legacy systems, heterogeneous toolchains, regulatory obligations and limited security staff complicate implementation of sophisticated DevSecOps architectures. Shift-left in such conditions demands gradual adoption: first codifying existing manual checks as pipeline steps, then introducing automated scanning and only later moving towards unified policy-as-code and centralised risk dashboards. The evaluation framework formulated in this study enables ranking of potential measures by projected impact on exposure metrics rather than by technological novelty, which adapts shift-left strategy to real organisational limitations.

4. Discussion

The analysis of publications indicates convergence on several structural characteristics of mature shift-left DevSecOps deployments, yet interpretation of effectiveness varies. A considerable share of works focuses on conceptual explanation of DevSecOps principles and on descriptive case studies rather than on formal risk metrics. Table 1 groups the sources according to the type of research and dominant focus related to shift-left security.

Table 1: Research focus of reviewed studies on shift-left DevSecOps [1–10]

Reference	Type of study	Primary focus related to shift-left security
[1]	Conceptual and qualitative analysis	Integrating Security-as-Code and automated checks into CI/CD for earlier vulnerability detection
[2]	Practice-oriented conceptual study	Use of automated scanning tools to move vulnerability management to early SDLC phases
[3]	Cloud-native framework description	Continuous security testing and automated compliance in Kubernetes and Docker pipelines
[4]	Conceptual paper with literature view	Combination of "shift left" and "secure right" strategies in cloud-native cybersecurity
[5]	Applied analytical study	Impact of DevSecOps practices on security of cloud-native application development
[6]	Systematic literature review	Thematic synthesis of DevSecOps automation, tool orchestration and cultural transformation
[7]	Practice-driven implementation report	Empirical evaluation of DevSecOps pipeline with security automation and MTTR reduction
[8]	Conceptual survey	Security integration into DevOps with emphasis on IaC, threat modelling and continuous testing
[9]	Applied conceptual and case-based work	DevSecOps for cloud-native applications with shift-left security and tool integration
[10]	Framework-oriented conceptual paper	Security-by-design guidelines using a shift-left DevSecOps approach across SDLC

The distribution in Table 1 illustrates that only one source [7] presents explicit empirical evaluation with concrete operational metrics such as MTTR and observed reduction of post-deployment vulnerabilities, while another subset [5; 8; 9] describes applied experiences in general qualitative terms. Conceptual and framework-oriented publications [1–4; 6; 10] focus on patterns, toolsets and organisational recommendations, which are valuable for design of shift-left strategies but do not in themselves confirm risk-reduction effect. This imbalance supports the conclusion of Mohammed et al. [6] about the need for more systematic empirical validation of DevSecOps practices.

To connect shift-left practices with information security risk throughout the life cycle, the articles implicitly map security controls to SDLC stages. Table 2 aggregates this mapping for the reviewed sources, concentrating on whether a publication explicitly addresses early-phase security activities (requirements and design), development-phase controls (coding and build) and pre-release or operational controls.

Table 2: Coverage of SDLC phases by shift-left security measures in reviewed publications [1–10]

Reference	Requirements & design (threat modelling, security requirements)	Coding & build (SAST, SCA, secrets, IaC)	Test / pre-release (DAST, container scanning, compliance)	Operations (monitoring, incident response)
[1]	Indirectly (security requirements via policies)	Explicit focus	Explicit, through pipeline-integrated tests	Brief mention
[2]	Not central	Strong emphasis on scanning tools	Strong emphasis	Limited
[3]	Implicit in framework	Significant focus on container/IaC	Significant focus on continuous testing	Discussed as part of cloud monitoring
[4]	Strong focus on early security strategy	Coverage of automation practices	Coverage of "secure right" continuous protection	Detailed discussion
[5]	Limited	Strong emphasis	Strong emphasis	Included
[6]	Analytical synthesis of all phases	Synthesised	Synthesised	Synthesised
[7]	Limited but present via design guidelines	Detailed implementation	Detailed implementation	Quantitatively evaluated
[8]	Explicit role of threat modelling	Emphasis on IaC and automated tests	Present	Present
[9]	Covered via policy and governance	Emphasis on DevSecOps tooling	Emphasis on Kubernetes and Docker security	Covered as part of cloud-native security
[10]	Central, through security-by-design	Recommended controls	Recommended controls	Touched in framework

The mapping in Table 2 shows that only a minority of works place strong emphasis on requirements and design phases [4; 6; 8; 10], even though threat modelling and early definition of security criteria are decisive for preventing architectural vulnerabilities. The majority of publications concentrate on coding, build, test and pre-release stages, where tools and automation are more mature and easier to integrate. This imbalance partially explains residual risk observed in industrial practice: if shift-left focuses almost exclusively on artefacts that are easy to scan, design-level flaws and insecure business logic may still reach production despite heavy automation in CI/CD. From the standpoint of the evaluation framework developed in this article, truly effective shift-left strategy must link tool-based controls in later phases with earlier analytical activities, ensuring that tests are derived from explicit threat models and requirements rather than scattered heuristics.

Another issue emerging from the discussion is organisational readiness for DevSecOps. Works [1; 6; 8; 9] underline the need for cultural change: developers assume responsibility for basic security hygiene, security engineers provide reusable policies and pipelines, operations teams ensure that monitoring and logging infrastructure reflects new patterns of change. In absence of such redistribution of responsibilities, shift-left tools risk becoming yet another layer of friction, producing long lists of findings without ownership. This risk is particularly high in large corporate environments described

in the earlier monograph, where existing change-management procedures, segregation-of-duties requirements and legacy tooling restrict the flexibility to adapt pipeline configurations rapidly [11]. The evaluation framework therefore includes organisational indicators: presence of cross-functional DevSecOps teams, degree of shared visibility into security metrics and formal integration of security tasks into backlog planning.

A further theme concerns alignment between shift-left and "secure-right" strategies. Manchana [4] explicitly frames DevSecOps as a continuous loop: early security is complemented by runtime protection, observability and incident response. Mohammed et al. [6] and Ramaswamy [7] stress that without telemetry and feedback from production environments, early-phase assumptions remain unverified. For risk management this means that the effectiveness of shift-left cannot be seen only through reduced number of vulnerabilities discovered late; feedback from operational incidents should continuously recalibrate threat models and test suites. The proposed integrated model therefore treats runtime events as measurements feeding back into early stages: detected intrusion patterns lead to new secure coding rules, modified SAST configurations and updated playbooks for threat modelling.

The contribution of the current study, compared with referenced works and the prior monograph [11], lies in explicitly risk-oriented formulation of shift-left evaluation criteria. Instead of viewing DevSecOps adoption as binary ("implemented / not implemented"), the article proposes to measure distribution of vulnerability detection across life-cycle phases, relative remediation effort and residual exposure in production. On top of that, organisational and process indicators supplement technical metrics and enable informed decisions about which shift-left measures bring tangible risk reduction in concrete corporate environments rather than merely aligning with generic best-practice lists.

5. Conclusion

The conducted analytical study shows that shift-left strategy within DevSecOps delivers reduction of information security risks only when security controls cover all phases of the software life cycle, from requirements and design to operation, and when they are executed as part of automated CI/CD pipelines. Publications collectively confirm that early integration of SAST, SCA, secrets-management and IaC validation leads to earlier detection of vulnerabilities and lower remediation costs, especially when feedback is provided directly to developers through pipeline gates and Security-as-Code mechanisms. At the same time, the literature reveals systematic under-representation of design-level activities and insufficient empirical quantification of risk reduction.

The evaluation framework developed in the article links shift-left practices to three groups of indicators: where and how vulnerabilities are discovered, how rapidly and economically they are remediated and how much residual exposure remains in production. Application of this framework in corporate environments described in the earlier monograph enables gradual, risk-prioritised introduction of DevSecOps controls, with clear justification for each new check or tool in terms of its contribution to reduced attack surface and improved resilience of software services. For researchers the framework supplies a structure for future empirical studies; for practitioners it provides a basis for aligning DevSecOps investments with measurable security outcomes rather than with generic industry maturity benchmarks.

References

[1] Chittibala, D. R. (2023). DevSecOps: Integrating security into the DevOps pipeline. International Journal of Science and Research (IJSR), 12(12), 2074–2078. (https://doi.org/10.21275/SR24304171058)

[2] Dachepally, R. (2025). DevSecOps: Shifting security left with automated scanning tools. IJAIDR, 16(1). (https://doi.org/10.5281/zenodo.14993296)

[3] Gudala, L. (2020). Bridging Dev, Sec, and Ops: A cloud-native security framework. International Journal of Intelligent Systems and Applications in Engineering, 8(4), 297–308. (https://ijisae.org/index.php/IJISAE/article/view/7296)

[4] Manchana, R. (2024). DevSecOps in cloud native cybersecurity: Shifting left for early security, securing right with continuous protection. International Journal of Science and Research (IJSR), 13, 1374–1382. (https://doi.org/10.21275/SR24822104530)

[5] Mavani, C., & Mistry, H. (2022). Securing the DevOps lifecycle: Evaluating the impact of DevSecOps practices on cloud-native application development, 4, 353–365. (https://d1wqtxts1xzle7.cloudfront.net/125213405/SECURING_THE_DEVOPS_LIFECYCLE_2022_37-libre.pdf?1762022119=&response-content-disposition=attachment%3B+filename%3DSECURING_THE_DEVOPS_LIFECYCLE_EVALUATING.pdf&Expires=1764141821&Signature=KVCsKDlsswl~d4dUfLnzpBqrEtcUSCIIjXGRHV6gSzvk7O-e4nyDmmmBH1H2HJ75W7vJrRAQfRDNkMec4U6EbdVp6CgRoy1HII2~qE2kXHpJy5wx6pzGCllK7X~Yh83umlZX9IewMrWSNeG7Zu5IcLXuL63P88fextpdv~Z5uS-yvE3RpImEDyohD9OSC5M1vC~q3qFX7Rs5U0J1~6Z-5mI9Wnfy65BnAyqFOv07LxiCynmE2eGqhYZ8W7-S~Pd8z05bqR-3-3v5MOzBglyfDIHqb9gNmsmaHdWkmC6QgC4h-bRdaoGl7tQv3U5hHj9uOGi2eKBMUXjrXlErHsnTGQ__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA)

[6] Mohammed, K. I., Shanmugam, B., & El-Den, J. (2025). Evolution of DevSecOps and its influence on application security: A systematic literature review. Technologies, 13(12), 548. (https://doi.org/10.3390/technologies13120548)

[7] Ramaswamy, Y. (2023). DevSecOps in practice: Embedding security automation into agile software delivery pipelines. Journal of Computational Analysis and Applications, 31, 1372–1381. (https://doi.org/10.48047/jocaaa.2023.31.04.27)

[8] Raskhan. (2025). Security in DevOps (DevSecOps): Integrating security into the development pipeline. International Journal of Advanced Research in Computer Science & Technology (IJARCST), 8(3),

12140–12144. (https://doi.org/10.15662/IJARCST.2025.0803001)

[9] Reddy, A. K., Alluri, V. R. R., Thota, S., Ravi, C. S., & Bonam, V. S. M. (2021). DevSecOps: Integrating security into the DevOps pipeline for cloud-native applications. Journal of Artificial Intelligence Research and Applications, 1(2), 89–114. (https://jairajournal.org/index.php/publication/article/view/41)

[10] Voruganti, K. K. (2021). Implementing security by design practice with DevSecOps shift left approach. Journal of Technological Innovations, 2(1). (https://doi.org/10.93153/esjrgj76)

[11] Romm, N. (2025). Methodology for automating and securing DevOps processes: CI/CD optimization and increasing release stability in the corporate environment [Paperback]. (English ed.).