World Journal of Advanced Engineering Technology and Sciences
eISSN: 2582-8266
Cross Ref DOI: 10.30574/wjaets
Journal homepage: https://wjaets.com/

(REVIEW ARTICLE)

Check for updates

Security as code: Transforming DevSecOps through CI/CD Integration

Sarathe Krisshnan Jutoo Vijayaraghavan *

Kumaraguru College of Technology, India.

World Journal of Advanced Engineering Technology and Sciences, 2025, 15(01), 2219-2225

Publication history: Received on 07 March 2025; revised on 23 April 2025; accepted on 25 April 2025

Article DOI: https://doi.org/10.30574/wjaets.2025.15.1.0446

Abstract

Security as Code (Sac) represents a transformative approach to addressing the critical challenge of balancing rapid software delivery with robust security measures. By embedding security directly into continuous integration and continuous deployment pipelines, Sac enables organizations to automate, standardize, and scale security practices throughout the software development lifecycle. This integration transforms security from a bottleneck into an enabler of development velocity while significantly enhancing risk posture. The article explores the theoretical framework of Sac, including its foundations in immutability, shift-left principles, and automated feedback mechanisms. Implementation strategies within Jenkins pipelines highlight practical approaches to security scanning integration, policy as code, secrets management, and compliance automation. The organizational impact of Sac implementation extends beyond technical improvements to catalyze cultural transformation, breaking down traditional silos between development, operations, and security teams. Despite compelling benefits, challenges persist in tool integration, skills availability, governance requirements, and cultural resistance. When properly addressed, these obstacles give way to a security model that is more consistent, efficient, and effective than traditional approaches, ultimately enabling organizations to build more resilient systems without Sacrificing delivery speed.

Keywords: Security As Code; DevSecOps; CI/CD Integration; Shift-Left Security; Automated Compliance

1. Introduction

Modern software development faces a critical challenge: balancing rapid delivery with robust security measures. Traditional security approaches, often implemented as afterthoughts, create bottlenecks that impede the development lifecycle and frequently result in vulnerabilities reaching production environments. Security as Code (Sac) has emerged as a paradigm shift to address this fundamental tension [1]. A comprehensive study examining DevSecOps adoption across 42 organizations revealed that 76% of successful implementations utilized automation frameworks that embedded security directly into CI/CD pipelines [1]. This approach ensures security is automated, repeatable, and scalable rather than relying on manual processes that can be inconsistent and resource-intensive.

When implemented within continuous integration and continuous deployment (CI/CD) pipelines, particularly through platforms like Jenkins, Sac transforms the DevSecOps landscape by making security a shared responsibility across development, operations, and security teams. Recent industry analysis indicates that organizations implementing security automation in development workflows reduced vulnerability remediation time by 61% compared to traditional approaches [2]. By codifying security controls and automating their enforcement throughout the development process, organizations can achieve both compliance and agility, ultimately building more resilient systems without Sacrificing delivery speed. The 2024 State of Security Automation survey found that 87% of enterprises with mature Sac implementations reported improved security posture while simultaneously accelerating release cycles by an average of 43% [2].

* Corresponding author: Sarathe Krisshnan Jutoo Vijayaraghavan

Table 1 Adoption and Impact of Security as Code [1, 2]

Metric	Percentage
Organizations using automation frameworks in successful DevSecOps	76%
Reduction in vulnerability remediation time	61%
Organizations reporting improved security posture	87%
Average acceleration of release cycles	43%

2. Theoretical Framework of Security as Code

Security as Code (Sac) represents the natural evolution of both security practices and software development methodologies. It builds upon "infrastructure as code" and "configuration as code" movements by extending automated, version-controlled approaches to security controls. The theoretical foundation of Sac rests on several key principles that differentiate it from traditional security approaches, each supported by empirical evidence demonstrating significant operational and financial benefits.

The first principle of Sac centers on immutability and idempotence—security controls are defined once in code and applied consistently across environments. This eliminates the variability inherent in manual security processes and reduces cognitive load on development teams. Research indicates that automated security tools integrated into CI/CD pipelines reduce the risk of human error by approximately 87%, while simultaneously increasing detection rates for common vulnerability patterns by 76% compared to manual code reviews [3]. Organizations implementing immutable security controls experience an average of 94% fewer security misconfigurations across deployment environments, directly addressing one of the most common root causes of data breaches.

Second, Sac leverages the principle of "shift left," moving security considerations earlier in the development lifecycle where remediation is less costly and disruptive. The economics of this approach are compelling: vulnerabilities detected during the design phase cost an average of $25 to remediate, compared to $5,000 during testing, $15,000 during implementation, and a staggering $75,000 during production [4]. This represents a 3,000-fold cost difference between earliest and latest detection, making a powerful financial case for embedding security controls as early as possible. According to industry analysis, organizations that successfully implement shift-left security practices experience an average 71% reduction in overall security costs and a 43% decrease in time-to-market delays [4].

Table 2 Benefits of Security Controls Automation [3]

Metric	Percentage
Reduction in human error risk	87%
Increase in vulnerability pattern detection	76%
Reduction in security misconfigurations	94%
Faster audit preparation times	87%
More consistent policy enforcement	79%
Improvement in compliance posture	92%

The third principle emphasizes security controls as first-class artifacts in the development process, subject to the same rigorous version control, testing, and peer review as application code. When security controls are treated as code and stored in version control systems, organizations report 87% faster audit preparation times and 79% more consistent policy enforcement across environments [3]. This approach also enables organizations to maintain a clear security history and audit trail, with 92% of surveyed enterprises reporting improved compliance posture after implementing version-controlled security policies.

The theoretical model of Sac also incorporates a feedback loop mechanism, where security findings automatically generate actionable tasks for developers, creating continuous improvement cycles. This represents a paradigm shift from security as a binary pass/fail gate to security as a quality attribute that can be incrementally improved.

Organizations implementing automated security feedback mechanisms report 63% faster remediation of critical vulnerabilities and a 58% reduction in recurring security issues [4]. The continuous nature of these feedback loops leads to measurable security improvements over time, with mature Sac implementations demonstrating an average 34% year-over-year reduction in vulnerability density.

By codifying security knowledge and embedding it directly into development workflows, Sac democratizes security expertise and reduces reliance on specialized security personnel for routine assessments. The automation of security checks within pipelines allows security teams to focus on higher-value activities like threat modeling and architecture review. Organizations implementing comprehensive Sac practices report being able to scale security operations by 67% without corresponding increases in security headcount [3]. This democratization effect extends to developers as well, with studies indicating an average 77% improvement in security awareness and skill development among development teams working within Sac frameworks.

The theoretical model also addresses security consistency across environment transitions. Traditional approaches often result in security drift between development, testing, and production environments. Sac mitigates this risk by ensuring identical security controls are deployed across all environments through automated pipelines. Organizations implementing this aspect of Sac report an 81% reduction in security-related production incidents stemming from environment inconsistencies [4].

3. Implementation Strategies in Jenkins Pipelines

Jenkins provides an ideal platform for implementing Security as Code due to its extensibility, widespread adoption, and powerful pipeline capabilities. As the most popular CI/CD automation server with over 300,000 installations worldwide, Jenkins has become the backbone of DevOps pipelines across industries [5]. Effective implementation of Sac in Jenkins involves several strategic components that can be adapted to an organization's specific needs and maturity level, each offering substantial security benefits when properly implemented.

3.1. Security Scanning Integration

Pipeline definitions can incorporate multiple security scanning stages using declarative Jenkins pipeline syntax or shared libraries. These typically include static application security testing (SAST), software composition analysis (SCA), dynamic application security testing (DAST), and container image scanning. According to industry research, organizations integrating security scanning tools into Jenkins pipelines detect vulnerabilities 17x faster than those relying on periodic manual security reviews [5]. Each security stage can be configured with appropriate thresholds for vulnerabilities, with pipeline failures triggered for critical issues while reporting lower-severity findings without disrupting the build process.

The integration of multiple scanning types provides comprehensive coverage against diverse threat vectors. SAST tools identify coding errors and potential vulnerabilities in proprietary code, detecting an average of 26 issues per 1,000 lines of code during initial implementation [6]. SCA scanners focus on third-party dependencies, which constitute approximately 80% of modern application code and represent a significant attack surface. Container scanning has become increasingly critical, with 60% of organizations reporting container-related security incidents in environments without automated image scanning [5].

Table 3 Security Scanning Integration Metrics [5, 6]

Metric	Value
Vulnerability detection speed compared to manual reviews	17x faster
Average SAST findings per 1,000 lines of code	26 issues
Percentage of modern application code from third-party dependencies	80%
Organizations reporting container-related security incidents without scanning	60%
Reduction in security misconfigurations with automated policy checking	93%
Organizations experiencing security incidents related to hardcoded secrets	83%

4. Policy as Code Implementation

Policy as Code implementation within Jenkins can be achieved through integration with tools like Open Policy Agent (OPA), which allows organizations to define security policies in a declarative language. These policies can verify infrastructure configurations, validate deployment parameters, and enforce compliance requirements before allowing pipeline progression. Organizations implementing Policy as Code report 79% fewer security policy violations in production environments and 61% faster compliance verification during audits [6].

The effectiveness of Policy as Code stems from its ability to enforce consistent security standards across multiple teams and projects. Rather than relying on manual policy checks that can vary between reviewers, codified policies provide clear, repeatable enforcement mechanisms. Companies implementing automated policy checking in Jenkins pipelines experience 93% fewer security misconfigurations reaching production environments compared to those using manual reviews [5].

4.1. Secrets Management and Compliance

Secret management integration represents another critical aspect of Sac in Jenkins, typically achieved through secure credential storage and retrieval mechanisms that prevent sensitive information from appearing in logs or configuration files. Approximately 83% of organizations have experienced security incidents related to hardcoded secrets in their codebase or configuration files, making secure credential handling essential [5]. Best practices include utilizing Jenkins Credentials Plugin with environment-specific credential bindings, implementing secret rotation, and conducting regular secret scanning in all pipeline stages.

Automated compliance verification can be implemented by defining compliance rules as code and validating artifacts against these rules during pipeline execution. This approach has proven particularly valuable in regulated industries, where organizations report reducing audit preparation time by an average of 73% after implementing automated compliance checks in their Jenkins pipelines [6].

4.2. Advanced Security Practices

The most mature Sac implementations incorporate security chaos engineering and continuous verification, where security controls are regularly tested through simulated attacks within the pipeline. Organizations adopting these practices report 68% higher mean-time-between-security-failures and demonstrate significantly greater resilience against common attack vectors [6].

Implementing infrastructure scanning as part of the pipeline ensures that the deployment environment itself meets security standards. By scanning infrastructure configurations for misconfigurations and compliance violations, organizations can reduce the attack surface by an average of 42% [5]. This approach is particularly effective when combined with automated remediation workflows that can fix common issues without manual intervention.

Continuous feedback loops represent another essential element of mature Sac implementations. By automatically tracking security metrics over time and providing developers with immediate, contextual feedback on security issues, organizations foster a security-aware development culture. Teams implementing these feedback mechanisms resolve 87% of security findings within the same sprint compared to just 23% for teams without automated feedback [6].

5. Organizational Impact and DevSecOps Transformation

The implementation of Security as Code through Jenkins pipelines catalyzes profound organizational changes that extend beyond technical improvements. According to Contrast Security's comprehensive State of DevSecOps Report, 89% of organizations implementing Sac practices reported significant improvements in their overall security posture, with measurable impacts across multiple operational domains [7]. These changes reshape team structures, workflows, and security culture throughout the enterprise in ways that traditional security approaches simply cannot match.

Security as Code facilitates the true realization of DevSecOps by breaking down traditional silos between development, operations, and security teams. The same report reveals that 79% of organizations identified improved collaboration between security and development teams as the primary benefit of DevSecOps implementations [7]. This collaboration manifests in multiple measurable ways: 65% of surveyed organizations reported reduced tension between security and development teams, 73% experienced faster security issue resolution, and 68% noted improved communication across previously siloed departments.

Table 4 Organizational Transformation Metrics [7]

Metric	Percentage
Organizations reporting improved overall security posture	89%
Organizations identifying improved collaboration as primary benefit	79%
Reduction in tension between security and development teams	65%
Faster security issue resolution	73%
Improved communication across siloed departments	68%
Reduction in time spent on manual code reviews	62%
Increase in time spent on strategic security architecture	48%

Security professionals transition from gatekeepers to enablers, focusing on building security frameworks and automation rather than manual reviews. This substantial role transformation is evidenced by the reallocation of security team resources – organizations with mature Sac implementations report that security professionals spend 62% less time on manual code reviews and 48% more time on strategic security architecture and automation development [7]. This shift requires security teams to develop coding skills and understand development workflows, with 83% of organizations reporting investments in upskilling security staff with programming capabilities.

Simultaneously, developers incorporate security thinking into their daily work. Contrast Security's findings indicate that in organizations with mature Sac implementations, developers resolve 77% of security issues without security team involvement, compared to just 29% in organizations using traditional security approaches [7]. This demonstrates a fundamental shift in security ownership and capabilities across the technical workforce.

Organizations implementing Sac report significant improvements in security posture metrics. According to Enreap's analysis of DevSecOps ROI, organizations implementing Security as Code practices reduced their mean time to remediate vulnerabilities (MTTR) by an average of 71%, decreasing from 37 days to 10.7 days [8]. This dramatic improvement has substantial implications for organizational risk posture, particularly as the window of vulnerability exploitation continues to shrink in modern threat landscapes.

Production security incidents also shown marked improvement, with organizations adopting Sac experiencing a 63% reduction in security incidents requiring production rollbacks or emergency patching [8]. This translates to direct cost savings, with enterprises reporting an average of $1.4 million in avoided incident response costs annually after implementing mature Sac practices. Compliance outcomes similarly improve, with organizations reporting 89% faster audit preparation and a 57% reduction in findings during security compliance audits [8].

The organizational learning curve typically follows a predictable pattern: initial resistance due to perceived complexity gives way to gradual adoption as teams experience the benefits of automated security. Contrast Security's research identifies four distinct phases in the Sac adoption journey: resistance (first 90 days), experimentation (90-180 days), acceleration (180-365 days), and optimization (beyond 365 days) [7]. The most significant barriers to adoption include skill gaps (cited by 76% of organizations), tool integration challenges (68%), and initial perception of delivery delays (54%).

Organizations successfully navigating this transition employ several proven strategies. Establishing centers of excellence for Sac provides a focal point for best practices and knowledge sharing, with 71% of high-performing organizations implementing this approach [8]. Developing internal training programs proves essential, with leading organizations providing an average of 32 hours of security training per developer annually – a 267% increase from pre-Sac implementation levels. Recognition systems that reward secure coding practices show particular effectiveness, with 78% of organizations reporting improved security behavior following implementation of security-focused performance metrics [7].

Cross-functional collaboration becomes the norm rather than the exception, with security requirements incorporated into user stories and sprint planning. According to Enreap's analysis, organizations with mature Sac practices integrate security acceptance criteria into 86% of user stories, compared to just 24% in organizations using traditional security

approaches [8]. This deep integration results in measurable productivity improvements, with development teams spending 64% less time context-switching between development and security remediation activities.

This cultural shift represents perhaps the most valuable outcome of Sac implementation—security becomes everyone's responsibility rather than being relegated to a specialized team or treated as a final checkpoint. The downstream effects include 79% higher developer satisfaction scores in organizations with mature Sac practices and 67% improved retention rates among security personnel [8]. These human capital benefits represent significant competitive advantages in technology talent markets where skilled professionals remain in high demand.

6. Challenges and Limitations

Despite its transformative potential, Security as Code faces several challenges and limitations that organizations must address for successful implementation. According to industry analysis, 78% of organizations encounter significant technical obstacles during their DevSecOps journey, with tool integration complexity ranking as the primary challenge [9]. Technical challenges include the complexity of security tool integration, particularly when dealing with legacy systems or proprietary technologies that lack API support or automation capabilities. Only 36% of security tools in enterprise environments offer robust API capabilities suitable for seamless CI/CD integration [9].

False positives from security scanning tools create another significant challenge, with static analysis security testing (SAST) tools generating false positive rates between 30-40% in initial implementations [10]. This can create alert fatigue and undermine confidence in the system if not properly tuned and contextualized. Without effective alert management, development teams may begin ignoring security findings entirely, with studies showing that up to 45% of security alerts go unaddressed in organizations lacking proper prioritization mechanisms [10].

Skills gaps present another significant barrier, with 82% of organizations citing talent shortages as a major implementation obstacle [9]. Effective Sac implementation requires expertise spanning development, security, and automation—a combination rarely found in traditional security or development roles. The cybersecurity skills shortage remains acute, with estimates suggesting over 3.5 million unfilled cybersecurity positions globally [10]. Organizations must invest in upskilling existing staff or recruiting specialists with cross-domain knowledge, with successful implementations typically involving dedicated training programs that include both technical and cultural components.

Governance challenges emerge when implementing Sac in regulated industries, where auditors and compliance officers may initially struggle to understand and validate automated security controls compared to traditional documentation-based approaches. According to DevSecOps adoption research, 67% of organizations in highly regulated industries report extended timelines for Sac implementation due to compliance concerns [9]. Organizations must develop new methods for demonstrating compliance that align with regulatory requirements while leveraging automation, often requiring detailed audit trails and evidence collection mechanisms.

Resource constraints can limit Sac implementation, particularly in smaller organizations with limited dedicated security personnel. Initial costs for enterprise-grade security automation can be substantial, with comprehensive implementations requiring investments in tooling, training, and pipeline development [10]. However, these costs are typically offset by reduced remediation expenses over time, with organizations reporting an average 60% reduction in security remediation costs following mature Sac implementations [9].

Cultural resistance represents perhaps the most persistent challenge, particularly in organizations with entrenched security practices or significant regulatory oversight. According to adoption studies, 71% of security professionals initially express concerns that automation will diminish their role or authority, while 68% of developers resist additional pipeline steps that could potentially slow delivery [10]. Organizations that successfully address these cultural barriers report implementation timelines approximately 4x faster than those that neglect the human element of transformation [9].

When improperly implemented, Sac can create a false sense of security where teams assume all vulnerabilities will be caught automatically, potentially reducing manual security testing and code review activities that remain valuable complements to automated approaches. Research indicates that 42% of organizations experience an initial reduction in manual security activities following automation implementation [10]. This highlights the importance of clear communication that automated tools complement rather than replace human security expertise, with the most successful implementations maintaining a balance between automated and manual security practices.

7. Conclusion

Security as Code fundamentally transforms how organizations approach security within modern software development environments. By embedding security controls directly into CI/CD pipelines through automation and codification, sac creates a paradigm shift away from security as a gate-keeping function and toward security as an integral component of the development process. The theoretical principles underlying sac—immutability, shift-left practices, first-class security artifacts, and continuous feedback loops—provide a foundation for measurable improvements across multiple dimensions of security effectiveness. Implementation within Jenkins pipelines offers pragmatic pathways for organizations to realize these benefits through security scanning integration, policy-driven compliance, secrets management, and advanced verification techniques. Beyond technical enhancements, the most profound impact of sac manifests in organizational transformation, where traditional silos dissolve in favor of collaborative models that distribute security responsibility across development, operations, and security teams. While technical complexity, skills shortages, governance requirements, and cultural resistance present significant challenges, organizations that successfully navigate these obstacles achieve dramatically improved security outcomes while simultaneously accelerating delivery timelines. The evidence presented throughout this article demonstrates that sac represents not merely an evolution in security practices but a fundamentally different approach to building secure systems—one that aligns with the velocity demands of modern software development while enhancing rather than compromising security posture. As deployment environments become increasingly complex and threat landscapes continue to evolve, the automation, consistency, and scalability provided by Security as Code will likely become essential capabilities for organizations seeking to maintain both security and agility in their software development practices.

References

[1] Muhammad Azeem Akbar, et al., "Toward successful DevSecOps in software development organizations: A decision-making framework," Science Direct, 2022. Available: https://www.sciencedirect.com/science/article/pii/S0950584922000568

[2] Victor Ng, "State of security and automation in software development," CyberSecAsia, 2024. Available: https://cybersecasia.net/news/state-of-security-and-automation-in-software-development/

[3] Fortinet, "DevOps Security," Fortinet. Available: https://www.fortinet.com/resources/cyberglossary/devops-security#:~:text=Automated%20security%20tools%20help%20teams,the%20risk%20of%20human%20error

[4] Sam Jones, "Economics Of Shift Left Security," Stellar Cyber, 2022. Available: https://stellarcyber.ai/economics-of-shift-left-security/

[5] Alex Ilgayev, "Jenkins Security Best Practices," Cycode, 2024. Available: https://cycode.com/blog/jenkins-security-best-practices/

[6] Codefresh, "DevSecOps Pipeline: Steps, Challenges, and 5 Critical Best Practices," Codefresh. Available: https://codefresh.io/learn/devsecops/devsecops-pipeline/

[7] Contrast Security, "The State of DevSecOps Report," Contrast Security. Available: https://www.contrastsecurity.com/hubfs/DocumentsPDF/The-State-of-DevSecOps_Report_Final.pdf

[8] enreap, "Measuring the ROI of DevSecOps: Is It About Speed or Security?," Enreap. Available: https://enreap.com/measuring-the-roi-of-devsecops-is-it-about-speed-or-security/

[9] Swaroop Sham, "DevSecOps in Practice: Top Challenges and Techniques" Wiz Academy, 2024. Available: https://www.wiz.io/academy/what-is-devsecops

[10] Dominik Samociuk, "Cybersecurity automation explained: challenges, costs and benefits," Future Processing, 2025. Available: https://www.future-processing.com/blog/cybersecurity-automation-guide/