© NOV 2025 | IRE Journals | Volume 9 Issue 5 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I5-1712035

Devsecops In Practice: How Integrating Security into Ci/Cd Pipelines Changes the Way We Engineer Software

UDOKPORO JAMACHI BERNARD
De Montfort University

Abstract- ***The Continuous Integration (CI) and Continuous Deployment (CD), which was rapidly ratified by the software engineering development industry, turned into a fast-paced process, causing new insecurity threat to be generated. This paper therefore, explains how integrating security into CI/CD pipelines changes the way we engineer software through Development Security Operations (DevSecOps). Integrating security into CI/CD pipelines via DevSecOps fundamentally transforms software engineering by shifting security from a late-stage bottleneck to an intrinsic, automated part of the entire development lifecycle, promoting early vulnerability detection, reducing costs and risks, fostering a collaborative culture, and ultimately enabling faster delivery of inherently more secure software. This "shifting left" approach uses automation and tools like Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) to embed security checks, policy enforcement, and monitoring directly into developer workflows, ensuring security is a shared, continuous responsibility rather than a separate, disruptive activity.***

I. INTRODUCTION

In today's fast-paced software development landscape, organizations are under increasing pressure to deliver high-quality products quickly and efficiently. The traditional Software Development Life Cycle (SDLC), which separates development, testing, and deployment, often leads to slow release cycles, high maintenance costs, and limited visibility across the development process.

Emergence of DevOps - a cultural and technological shift that integrates software development (Dev) and operations (Ops). By automating repetitive tasks and ensuring scalable infrastructure, DevOps accelerates software delivery through Continuous Integration and Continuous Delivery (CI/CD) pipelines.

However, the speed of modern CI/CD pipelines introduces security risks. Misconfigurations and vulnerabilities can propagate rapidly across systems, leading to severe breaches. This highlights an urgent need: security must be an integral part of the development process rather than an afterthought. This is where DevSecOps comes in - embedding security into DevOps workflows to ensure that applications remain resilient without compromising agility.

DevSecOps DevSecOps stands for development, security, and operations. This approach integrates security practices within the DevOps process. DevSecOps aims to make security an integral part of the software development lifecycle, from initial design to production. Instead of treating security as a separate phase, it is woven into every part of the process, ensuring fast, secure, and high-quality software delivery.

In DevSecOps, teams collaborate closely, sharing responsibilities for security, which helps detect and address vulnerabilities early. By incorporating security early in the development process, organizations can reduce the risk of late-stage security issues, leading to more stable and secure products.

II. MEANING OF DEVSECOPS

DevSecOps is the practice of integrating security testing at every stage of the software development process. It includes tools and processes that encourage collaboration between developers, security specialists, and operation teams to build software that is both efficient and secure. DevSecOps brings cultural transformation that makes security a shared responsibility for everyone who is building the software.

III. WHAT DOES DEVSECOPS STAND FOR?

DevSecOps stands for development, security, and operations. It is an extension of the DevOps practice. Each term defines different roles and responsibilities

of software teams when they are building software applications.

Development: Development is the process of planning, coding, building, and testing the application.

Security: Security means introducing security earlier in the software development cycle. For example, programmers ensure that the code is free of security vulnerabilities, and security practitioners test the software further before the company releases it.

Operations: The operations team releases, monitors, and fixes any issues that arise from the software.

IV. BENEFITS OF DEVSECOPS

Adopting a DevSecOps approach requires careful change management, particularly for organizations with deep-rooted legacy processes and systems. While the initial transition may present challenges, the long-term benefits are substantial, transforming security from a reactive measure into a proactive, integral part of the development process.

Proactive Security:
Traditionally, much of cyber security has been a largely reactive function, addressing vulnerabilities and risks only after they have been discovered. However, the modern threat landscape demands a proactive approach, demonstrated, in part, by the rise of Continuous Threat Exposure Management (CTEM).

DevSecOps is a proactive software security strategy that embodies this shift. Implementing security practices throughout the SDLC not only helps organizations reduce the number and severity of potential threats, but also alleviates the budgetary strain associated with late-stage remediation. Further, a DevSecOps approach encourages development, operations and security teams to collaborate on key objectives, leading to overall organizational alignment, and greater work satisfaction.

Faster MTTR:
Identifying issues earlier in the cycle allows teams to prevent vulnerabilities from becoming deeply embedded in the codebase, making remediation easier and more efficient because fewer dependencies exist. The collaborative nature of DevSecOps rensures that all teams are aligned on remediation plans and disaster recovery strategies, further speeding up the remediation process.

Improved Compliance:
By integrating security into each stage of the development process, DevSecOps facilitates continuous compliance. Organizations can regularly test code and enforce version control and comprehensive documentation, making compliance an ongoing, integral part of the development cycle, rather than an after-the-fact checkbox exercise.

Speedy, Secure Software Delivery:
Although the initial investment in DevSecOps may seem daunting – encompassing costs related to new tools, training, and the integration of automated security practices – remediation prior to production minimizes the risk of costly breaches or security incidents in the long run. A secure development process enables organizations to deliver software efficiently, scale their operations effectively, and ultimately save time and money.

Scalable Security:
Security has historically been an isolated function, separate from the development process. DevSecOps breaks down the silos, reducing bottlenecks and fostering a more organized and collaborative environment. By democratizing security responsibilities, DevSecOps creates a security-first culture where security is understood as a critical component of broader business goals. When code is developed securely from the outset, the overall security and trustworthiness of the product are enhanced over time.

Reduce time to market:
With DevSecOps, software teams can automate security tests and reduce human errors. It also prevents the security assessment from being a bottleneck in the development process.

© NOV 2025 | IRE Journals | Volume 9 Issue 5 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I5-1712035

V. DEVSECOPS IN AGILE DEVELOPMENT

Agile is a mindset that helps software teams become more efficient in building applications and responding to changes. Software teams used to build the entire system in a series of inflexible stages. With the agile framework, software teams work in a continuous circular workflow. They use agile processes to gather constant feedback and improve the applications in short, iterative development cycles.

DevSecOps Compared to Agile Development:
DevSecOps and agile are not mutually exclusive practices. Agile allows the software team to act quickly on change requests. Meanwhile, DevSecOps introduces security practices into each iterative cycle in agile development. With DevSecOps, the software team can produce safer code using agile development methods.

VI. DEVSECOPS CULTURE

The DevSecOps culture combines communication, people, technology, and process.

Communication: Companies implement DevSecOps by promoting a cultural change that starts at the top. Senior leaders explain the importance and benefits of adopting security practices to the DevOps team. Software developers and operations teams require the right tools, systems, and encouragement to adopt DevSecOps practices.

People: DevSecOps leads to a cultural transformation that involves software teams. Software developers no longer stick with conventional roles of building, testing, and deploying code. With DevSecOps, software developers and operations teams work closely with security experts to improve security throughout the development process.

Technology: Software teams use technology to perform automated security testing during development. DevOps teams use it to check the app for security flaws without compromising the delivery timeline. For example, software teams use Amazon Inspector to automate continual vulnerability management at scale.

Process: DevSecOps changes the conventional process of building software. With DevSecOps, software teams perform security testing and evaluation at every stage of development. Software developers check for security flaws when they are writing the code. Then a security team tests the pre-release application for security vulnerabilities. For example, they might check for the following:

- Authorization so that users can access only what they require.
- Input validation so that software functions correctly when receiving abnormal data

Then software teams fix any flaws before releasing the final application to end users.

Security testing does not end after the application goes live. The operations team continues to monitor for potential issues, make amendments, and work with the security and development teams to release updated versions of the application. For example, they might use Amazon CodeGuru Reviewer to detect security vulnerabilities, disclosed secrets, resource leaks, concurrency issues, incorrect input validation, and deviation from best practices for using AWS APIs and SDKs.

VII. COMPONENTS OF DEVSECOPS

Successful implementation of the DevSecOps practice consists of the following components:

Code Analysis: Code analysis is the process of investigating the source code of an application for vulnerabilities and ensuring that it follows security best practices.

Change Management: Software teams use change management tools to track, manage, and report on changes related to the software or requirements. This prevents inadvertent security vulnerabilities due to a software change.

Compliance Management: Software teams ensure that the software complies with regulatory requirements. For example, developers can use AWS CloudHSM to demonstrate compliance with security,

privacy, and anti-tamper regulations such as HIPAA, FedRAMP, and PCI.

Threat Modeling: DevSecOps teams investigate security issues that might arise before and after deploying the application. They fix any known issues and release an updated version of the application.

Security Training: Security training involves training software developers and operations teams with the latest security guidelines. This way, the development and operations teams can make independent security decisions when building and deploying the application.

VIII. DEVSECOPS TOOLS

A variety of application security testing tools are required for successful DevSecOps adoption. They each analyze a specific stage of the SDLC or technology layer, offering a comprehensive view of application security and enabling teams to identify and remediate vulnerabilities effectively. Some of the most common tools include:

Static Application Security Testing (SAST):
Static Application Security Testing (SAST) tools analyze source code at rest, identifying errors and vulnerabilities "from the inside" of a software application. SAST testing is commonly referred to as open- or white-box testing as it examines the code itself and does not require code execution, making it comparable to a thorough code review. SAST allows development teams to detect and correct security issues early in the SDLC, before they become embedded in the codebase.

Dynamic Application Security Testing (DAST):
Dynamic Application Security Testing (DAST) takes place later in the SDLC, focusing on the running application, and is commonly referred to as a form of closed- or black-box testing. Instead of looking at the code, it tests the application "from the outside in" to evaluate externally exposed behavior – such as responses to HTTP for web applications – to identify any vulnerabilities that may be exploited by hackers . Unlike SAST, which examines the code itself, DAST tests the application in its runtime environment, making it a crucial tool for identifying issues that only become apparent during execution. This approach ensures that security vulnerabilities exposed during runtime get addressed before the application reaches production.

Interactive Application Security Testing (IAST):
Interactive Application Security Testing (IAST) combines the capabilities of SAST and DAST by analyzing both the application's code and behavior during runtime. IAST provides real-time vulnerability detection, integrating seamlessly into the CI/CD pipeline to prevent delays. Because it analyzes how the application runs in conjunction with other security measures and configurations, IAST offers more accurate and detailed insights than SAST or DAST alone. This comprehensive approach ensures that vulnerabilities are identified and addressed throughout the development lifecycle, from build to production.

Software Composition Analysis (SCA):
Software Composition Analysis (SCA) is essential for managing the security of open-source components within an application. Many modern applications rely on open-source libraries for accelerated software development, but these components can introduce vulnerabilities if not properly managed. SCA tools scan an application's codebase to identify the use of open-source software and assess its associated risks, such as known vulnerabilities, licensing issues, and dependencies. SCA tools compare their findings with vulnerability databases to offer insights into severity and prioritization of potential security issues, helping to mitigate risks associated with third-party code.

Infrastructure as Code (IaC) Scanning:
Infrastructure as Code (IaC) describes the automated provisioning, configuring, and orchestrating of infrastructure through code, ensuring consistency and repeatability. However, IaC can introduce vulnerabilities if misconfiguration or security flaws are present in the code that defines the infrastructure. IaC scanning tools analyze these infrastructure definition files for known vulnerabilities and misconfigurations before applying a configuration to a resource. This proactive approach to infrastructure security is critical in a DevSecOps environment, where the security of both the application and its underlying infrastructure must be maintained.

© NOV 2025 | IRE Journals | Volume 9 Issue 5 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I5-1712035

Secrets Detection:
Scanning involves scanning your source code, configuration files, and environment variables for sensitive information such as API keys, database credentials, or encryption keys. This is particularly important in CI/CD pipelines, where secrets are often stored in plaintext or hardcoded, increasing the risk of exposure to unauthorized parties.

IX. THE DEVSECOPS CI/CD PIPELINE

A DevSecOps pipeline integrates security practices within the DevOps process, ensuring that applications are secure from development through deployment. It aims to automate security checks and embed security at every step of software development. This approach shifts security left, addressing vulnerabilities early when they are cheaper and easier to fix. DevSecOps uses tools to integrate with existing workflows, covering code analysis, compliance checks, and more.

The adoption of DevSecOps addresses the limitations of traditional security, which often operates as a final step. Instead, it brings a collaborative approach where developers, operations, and security teams work together. This synergy enables faster delivery without compromising security, supporting organizations that adopt agile and continuous delivery models.

A secure CI/CD pipeline ensures that security is embedded at every stage of development, reducing the attack surface and mitigating risks. Key principles include:

- Shift-Left Security: Detecting vulnerabilities early minimizes rework and strengthens overall security.
- Automated Security Scanning: Security checks and compliance policies are integrated into CI/CD workflows, ensuring consistent application of best practices.

X. IMPORTANCE OF CI/CD IN DEVSECOPS

Continuous Integration/Continuous Deployment (CI/CD) is crucial in the DevSecOps ecosystem. CI/CD automates the integration, testing, and deployment of code changes, facilitating rapid and reliable software delivery. This automation is key to maintaining a fast-paced DevOps environment without compromising security.

In a DevSecOps context, CI/CD pipelines are designed to include security checks,automated testing, and compliance monitoring throughout the development cycle. This ensures that every code change is tested for security vulnerabilities before deployment, making the final product more secure.

Integrating security into CI/CD pipelines not only minimizes the risks of security breaches but also improves the overall quality of the software. It enables teams to deliver secure and functional software rapidly, meeting the market demands and ensuring customer satisfaction.

DevSecOps aims to help development teams address security issues efficiently. It is an alternative to older software security practices that could not keep up with tighter timelines and rapid software updates. To understand the importance of DevSecOps, we will briefly review the software development process.

XI. DEVSECOPS PIPELINE PHASES AND STEPS

Source Code Management (SCM):
Source Code Management (SCM) in a DevSecOps pipeline is the initial stage where code is stored, tracked, and managed. SCM systems provide a central repository for developers to collaborate and control code versions, helping teams manage changes and maintain code integrity.

By offering real-time version control, SCM tools enable the team to revert to previous code states, track modifications, and ensure that all changes are documented. Key SCM tools in DevSecOps include Git, Bitbucket, and GitLab, which support integration with security plugins to detect issues early in the development phase.

Continuous Integration (CI):
Continuous Integration (CI) automates the integration of code changes into a shared repository several times a day. In a DevSecOps pipeline, CI is essential for quickly identifying bugs and security

vulnerabilities by running automated tests on every code change. It fosters rapid feedback to developers, enabling them to fix issues early in the lifecycle.

Common CI tools like Jenkins, GitLab CI/CD, and CircleCI are often enhanced with security scanners that identify code flaws, insecure dependencies, or configuration issues during each integration, ensuring that security checks are a standard part of the development process.

Continuous Delivery (CD):
Continuous Delivery (CD) automates the deployment process, moving code changes through multiple environments until they are ready for production. In DevSecOps, CD ensures that all code is consistently validated against security policies before it progresses.

CD tools, such as Codefresh and Octopus Deploy, streamline this workflow by deploying code in a way that incorporates security checks at each stage. This stage emphasizes reliable and repeatable deployments, enabling the team to quickly deliver secure updates to production without manual intervention.

Automated Security Testing:
Automated security testing is critical to DevSecOps, as it continuously checks code for vulnerabilities and security misconfigurations. This includes several types of testing, such as Static Application Security Testing (SAST) for source code analysis, Dynamic Application Security Testing (DAST) for runtime testing, and Software Composition Analysis (SCA) for dependency checks.

Tools like SonarQube, ZAP, and Mend integrate with CI/CD pipelines to provide fast feedback, allowing developers to address security issues without delaying delivery.

Configuration Management:
Configuration management ensures that system configurations remain consistent across environments, preventing security risks caused by misconfigured settings. In DevSecOps, tools such as Ansible, Puppet, and Chef manage and enforce configuration policies, automatically applying secure configurations and maintaining compliance standards.

By integrating with version control systems, these tools enable teams to track configuration changes, detect discrepancies, and remediate them quickly, thus minimizing the risk of configuration drift and potential security gaps.

Orchestration and Automation:
Orchestration and automation tools simplify complex workflows, managing and automating interconnected tasks within the pipeline. Automation minimizes manual interventions, reducing the chances of human error and improving the reliability of security enforcement throughout the pipeline.

In a DevSecOps context, orchestration tools like Kubernetes, Docker Swarm, and Jenkins enable teams to coordinate tasks across containers, services, and environments, ensuring that security checks and deployments follow a consistent, secure process.

Monitoring and Incident Response:
Monitoring and incident response provide visibility into application performance and potential security incidents in real-time. Effective monitoring tools, such as Prometheus, Grafana, and Splunk, alert teams to unusual activity or security breaches, enabling rapid detection and response.

By incorporating security monitoring, DevSecOps supports continuous threat detection and investigation, allowing teams to respond proactively to incidents and protect applications from emerging risks.

XII. SOFTWARE DEVELOPMENT LIFECYCLE

The software development lifecycle (SDLC) is a structured process that guides software teams to produce high-quality applications. Software teams use the SDLC to reduce costs, minimize mistakes, and ensure the software aligns with the project's objectives at all times. The software development life cycle takes software teams through these stages:

- Requirement analysis
- Planning

- Architectural design
- Software development
- Testing
- Deployment

XIII. HOW DEVSECOPS INTEGRATES SECURITY INTO CI/CD PIPELINES

Automation of Security Testing:
Security tools automatically perform tasks like Static Application Security Testing (SAST) (for code analysis), Dynamic Application Security Testing (DAST) (for runtime analysis), and Software Composition Analysis (SCA) (for dependency vulnerabilities) within the pipeline.

Policy as Code:
Security and compliance policies are encoded into the infrastructure and automation frameworks, ensuring consistent security configurations and adherence to regulatory standards like GDPR or HIPAA.

Infrastructure as Code (IaC):
Security is built into infrastructure provisioning, allowing for automated, secure configuration management and early detection of misconfigurations.

Rapid Feedback Loops:
Automated alerts, dashboards, and reports provide developers and security teams with real-time insights into vulnerabilities, allowing for quick remediation within their existing workflows.

Continuous Monitoring:
Security activities and controls are monitored continuously after deployment to identify and address new threats and vulnerabilities.
Key Changes to Software Engineering.

Security Becomes a Shared Responsibility:
Security is no longer the sole responsibility of a separate security team but a collaborative effort across development, operations, and security teams from the initial planning stages.

"Shift-Left" Security:
Security is moved to the left of the development lifecycle, meaning issues are identified and fixed much earlier, when they are cheaper and easier to resolve.

Reduced Costs and Risks:
Proactive identification and mitigation of vulnerabilities in the early stages significantly lowers the risk of costly security breaches, financial losses, and reputational damage.

Increased Velocity and Agility:
By embedding security into automated pipelines, organizations can maintain their agile development speed and continuous delivery without compromising security.

Improved Software Quality and Trust:
A holistic, integrated approach leads to more resilient, secure applications, fostering greater customer trust and ensuring compliance with industry standards.

How Integrating Security into CI/CD Pipelines Changes The Way We Engineer Software
Integrating security into CI/CD pipelines via DevSecOps changes software engineering by embedding security into every stage of the development lifecycle, shifting security "left" to find and fix vulnerabilities earlier, automating security checks, and improving collaboration between development, security, and operations teams. This proactive approach leads to more robust software, faster releases, and reduced costs associated with fixing security issues later in production.

The Key Changes to Software Engineering:

- Shifts Security Left: Security is no longer an afterthought but is integrated from the earliest stages, even within a developer's Integrated Development Environment (IDE).
- Automates Security Checks: Automated security testing and compliance checks are built into the pipeline, ensuring consistent application of security practices and continuous validation.
- Reduces Rework and Costs: By catching vulnerabilities early, teams avoid expensive and time-consuming fixes later in the development cycle.

- Improves Collaboration: DevSecOps breaks down traditional silos between development, security, and operations teams, fostering a shared responsibility for security.
- Increases Speed and Efficiency: The automation of security checks allows for faster delivery of software without compromising security, supporting rapid release cycles.
- Enhances Security Posture: By proactively identifying and addressing risks through methods like static and dynamic code analysis, dependency scanning, and container security scanning, organizations can significantly reduce their attack surface and likelihood of breaches.
- Enables Continuous Delivery: Security is integrated into the continuous delivery process, allowing security to keep pace with development rather than being a bottleneck.

XIV. CHALLENGES OF IMPLEMENTING A DEVSECOPS PIPELINE

Cultural Resistance:
Cultural resistance is a significant barrier to implementing DevSecOps, often stemming from traditional silos between development, operations, and security teams. This resistance can hinder the integration of security practices into the development process. Addressing cultural barriers requires fostering a collaborative environment where all stakeholders view security as a shared responsibility.

Promoting an organizational culture that values security within the development lifecycle is essential. Training and awareness programs can transform perceptions, showing the value of early security integration. Through leadership support and continuous education, teams can shift mindsets, encouraging collaboration and breaking down silos.

Legacy Systems:
Legacy systems present a formidable challenge in implementing DevSecOps pipelines. These systems often lack flexibility, making them difficult to integrate with modern development practices and security tools. This discrepancy requires careful planning to avoid disrupting existing operations while introducing security measures into legacy environments.

Modernizing legacy systems or creating workaround solutions can help incorporate DevSecOps practices. Teams may need to implement additional tools and processes to bridge the gap between outdated technologies and modern security standards. Addressing legacy systems requires investment, both in terms of resources and time, to ensure integration and adoption of DevSecOps methodologies.

Technical Debt:
Technical debt can hinder the adoption of DevSecOps by imposing constraints on resources and increasing development complexity. Accumulated quick fixes undermine code quality, complicating integration with security measures. Addressing technical debt is critical for the successful deployment of a DevSecOps pipeline.

Efforts to reduce technical debt require prioritizing refactoring tasks and implementing coding standards that prevent its accumulation. Regular code reviews and updates, along with automated testing tools, help in managing technical debt. Organizations must allocate time and resources to resolve outstanding issues, ensuring a cleaner, more secure codebase aligned with DevSecOps objectives.

Lack of Security Expertise:
A lack of security expertise within development and operations teams can impede DevSecOps implementation. Many teams may not have the knowledge required to embed security effectively throughout the lifecycle. Bridging this skills gap is crucial for successful integration and to realize the full benefits of a DevSecOps approach.

Training and hiring initiatives can enhance security expertise across teams, ensuring that security measures are understood and correctly applied. Organizations should invest in continuous learning opportunities, keeping teams updated on the latest security practices and tools. Building a security-focused culture supports the development of in-house expertise essential for managing and optimizing DevSecOps pipelines.

XV. CONCLUSION

The necessity, benefit and importance of Security can never be overemphasized, it is a fundamental requirement for modern software development and engineering. This paper explored how integrating security into the CI/CD pipeline using DevSecOps ensures applications remain resilient without compromising agility and. By shifting security left and automating key checks like SAST, DAST, SCA, secrets detection, and IaC/PaC, teams can detect vulnerabilities early and mitigate risks before deployment. As cyber threats grow, security can no longer be an afterthought — it must be embedded into development workflows to safeguard applications, maintain trust, and uphold software integrity in an era of rapid delivery.

The widespread adoption of agile software development workstreams and faster development cycles necessitates the integration of security into agile and DevOps processes. This evolution has given rise to DevSecOps, a practice that formalizes the incorporation of security into each stage of the agile software development life-cycle (SDLC).

Prior to agile development, when the waterfall approach to software development was dominant, security was treated as a final checkpoint; something addressed only after the product development was otherwise completed.

However, software development itself has been largely transformed into an agile and continuous process that increasingly combines development and operations (hence the term “DevOps”). This DevOps approach, combined with the evolving threat landscape and the increasing complexity of software environments, means security has become integral to every aspect of software development. This shift emphasizes the importance of embedding security throughout the SDLC, influencing collaboration, communication, and the creation of policies and procedures that span the entire process.

REFERENCES

[1] Adedamola Solanke (2022), Enterprise DevSecOps: Integrating Security into CI/CD Pipelines for Regulated Industries. February 2022. World Journal of Advanced Research and Reviews 13(02):633-648 DOI: 10.30574/wjarr.2022.13.2.0121

[2] Codefresh (2025), DevSecOps Pipeline: Steps, Challenges, and 5 Critical Best Practices. Codefresh by Octopus Deploy. 2025 Codefresh. Terms of Service.https://codefresh.io/learn/devsecops/devsecops-pipeline/

[3] Misbah Thevarmannil (2023), DevSecOps CI/CD: Enhancing Security in the Age of Continous Delivery. 16 November 202. Practical DevSecOps and Hysn Technologies Inc. registrations@practical-devsecops.com

[4] Naga Murali Krishna Koneru (2021), Integrating Security into CI/CD Pipelines: A DevSecOps Approach with SAST, DAST, and SCA Tools. October 2021. International Journal of Science and Research Archive 3(1):250-265DOI: 10.30574/ijsra.2021.3.1.0080

[5] Seemplicity (2025), DevSecOps. Yigal Alon St 94, building 2, Floor 14, Tel Aviv-Yafo, 6789139, Israel 181 Metro Drive, San Jose, CA 95110 sales@seemplicity.iopartners@seemplicity.io