International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2020): 7.86

Automated Policy Enforcement in DevSecOps: OPA, Kyverno, and Tekton Chains for Supply Chain Integrity

Sandhya Guduru

Masters in Information Systems Security, Software Engineer - Technical Lead

Abstract: ***As software development increasingly relies on fast, iterative delivery through CI/CD pipelines, ensuring robust security across the pipeline has become a critical challenge. The rise of DevSecOps promotes embedding security earlier in the development lifecycle. Still, traditional approaches often fall short in handling the complexity of cloud-native environments and securing the software supply chain. This paper explores the intersection of DevSecOps practices and Policy-as-Code tooling within Kubernetes-native pipelines to address key issues such as insecure configurations, weak policy enforcement, and limited build transparency. It highlights the role of tools like Open Policy Agent (OPA), Gatekeeper, Kyverno, Tekton Pipelines, and in-toto in automating compliance, enforcing policies, and validating build integrity. By identifying core problem areas—including misconfigurations, dependency risks, and lack of verifiable artifact metadata—the paper proposes a framework for improving CI/CD security posture. The proposed approach aims to strengthen trust, ensure artifact provenance, and enable scalable, secure software delivery in cloud-native environments.***

Keywords: Open Policy Agent (OPA), DevSecOps, Kyverno, Tekton Pipelines, in-toto, software supply chain security, artifact verification, admission control, Gatekeeper

1. Introduction

As organizations embrace continuous integration and continuous deployment (CI/CD) to accelerate software delivery, the need for robust security measures integrated into every development lifecycle phase has become more critical than ever. This shift has given rise to DevSecOps, a practice that embeds security into development and operations workflows. By automating security policies, testing, and compliance checks, DevSecOps ensures security is prioritized without slowing down delivery, making it essential for modern software teams.

In Kubernetes-native environments, automated policy enforcement is crucial in maintaining security and integrity. While Kubernetes offers significant flexibility through features like custom resource definitions (CRDs), admission controllers, and dynamic workloads, this flexibility can also create security risks. Without proper policy enforcement, teams may inadvertently deploy insecure configurations or allow vulnerable workloads into production. Automated policy enforcement tools address these risks by applying predefined rules that validate resource definitions and prevent unsafe deployments from reaching production clusters.

Recent software supply chain attacks, such as the SolarWinds breach, have highlighted severe vulnerabilities in the software development process. These incidents have shown that even small gaps in CI/CD workflows can be exploited by malicious actors to compromise artifacts, inject backdoors, or steal credentials. Securing the software supply chain now requires comprehensive visibility, rigorous integrity validation, and the ability to verify the authenticity of software artifacts throughout the development and deployment process.

This paper evaluates several critical tools designed to automate policy enforcement and enhance supply chain security. Open Policy Agent (OPA) and its Kubernetes-native extension, Gatekeeper, offer a flexible framework for defining and enforcing security policies using the Rego language. Kyverno, a Kubernetes-native policy engine, simplifies the adoption of policy-as-code practices by allowing policies to be written in YAML. Tekton Chains, an extension of Tekton Pipelines, secures artifact signing and integrates with the in-toto framework to capture Build Metadata and ensure artifact integrity. These tools provide the foundation for achieving supply chain integrity, transparency, and compliance in modern CI/CD environments.

2. Literature Review

The convergence of DevSecOps principles, Policy-as-Code frameworks, and software supply chain security tooling represents a growing focus in both academic and industry literature. This section reviews foundational concepts and tooling that support automated policy enforcement and integrity verification across modern Kubernetes-native CI/CD pipelines.

DevSecOps emerged as an evolution of the DevOps movement, aiming to embed security into every development lifecycle phase rather than treating it as a final checkpoint. Several studies and industry reports emphasize the importance of shifting security left, automating compliance checks, and codifying security controls into CI/CD pipelines [1]. Security-as-code and compliance-as-code have become essential practices to support this integration. These practices rely heavily on declarative policies that can be enforced at build time, deployment, or runtime—ensuring consistency, auditability, and repeatability across cloud-native environments.

As Kubernetes gained popularity for orchestrating containerized workloads, new approaches emerged to enforce policies at the cluster level. The Open Policy Agent (OPA) introduced a general-purpose, domain-agnostic policy engine designed to support Policy-as-Code across microservices and infrastructure [2]. OPA uses Rego, a declarative query language, to define and evaluate policies. Its Kubernetes-native extension, Gatekeeper, enables admission control by integrating directly with the Kubernetes API server.

OPA and Gatekeeper have become popular for validating custom resource definitions (CRDs), enforcing label standards, restricting insecure configurations, and ensuring compliance with organizational policies [3]. However, the complexity of writing Rego and debugging policy logic posed adoption challenges, especially for teams unfamiliar with domain-specific languages [4].

To address the need for Kubernetes-native, YAML-centric policy enforcement, Kyverno emerged as a CNCF sandbox project. Unlike OPA, Kyverno was designed specifically for Kubernetes, allowing policies to be defined using familiar YAML syntax and native Kubernetes patterns [5]. This approach lowered the barrier to entry for platform teams and developers, enabling broader adoption of Policy-as-Code practices. Kyverno supports admission control, mutation, and validation policies and integrates seamlessly into GitOps workflows and CI/CD pipelines. Its focus on native resource handling, schema validation, and conditional logic based on resource fields made it particularly suitable for enforcing secure defaults in cloud-native environments.

Parallel to the evolution of policy enforcement, attention began shifting toward securing the integrity of the software supply chain. The SolarWinds breach and rising concerns around dependency confusion attacks highlighted systemic weaknesses in artifact provenance, dependency trust, and build process verification. To counter these threats, the in-toto framework introduced a model for supply chain integrity based on cryptographic attestations [6]. in-toto captures metadata at each step of the software lifecycle, allowing consumers to verify that authorized parties built, tested, and signed a package or image.

The Tekton project advanced this idea under the CD Foundation, which introduced Tekton Pipelines for defining cloud-native CI/CD workflows. Building on this, Tekton Chains provided a secure method for generating in-toto attestations and signing build artifacts automatically during pipeline execution [7]. Tekton Chains leverages in-toto to record metadata and links it to build steps while using signature frameworks like cosign and Rekor to ensure transparency and auditability.

Figure 1 illustrates the flow of a typical Tekton CI/CD pipeline integrating Chains and in-toto for attestation capture. pipeline

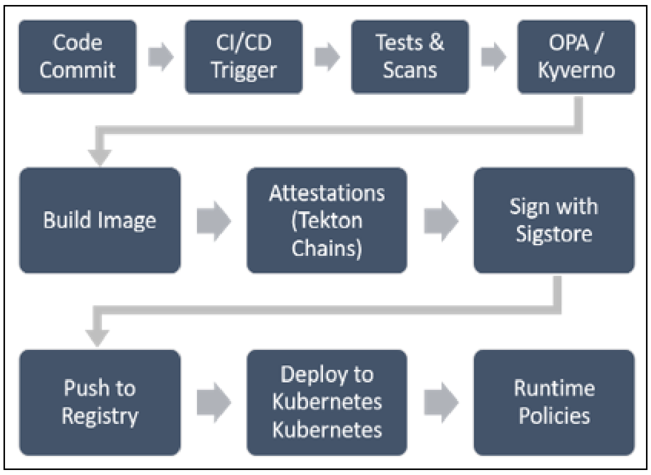

Figure 1: End-to-end flow of a Tekton-based CI/CD

As a concept, supply chain security evolved to include secure artifact delivery and verification and continuous monitoring and tracking of dependencies throughout the software development lifecycle. Researchers and security professionals pointed out that traditional approaches to software security often overlooked the interconnectedness of build pipelines and third-party dependencies, which created significant vulnerabilities [8]. The introduction of attestation-based frameworks, such as in-toto and Tekton Chains, sought to address this by capturing metadata from every phase of the build and deployment process and linking it to cryptographic signatures to ensure that software artifacts came from trusted sources.

In line with these developments, the importance of adopting zero-trust principles for software supply chain security was underscored in several works. Zero-trust architecture, which assumes that no actor—inside or outside the network—should be trusted by default, is particularly effective in mitigating threats from compromised supply chains [9]. The widespread adoption of zero-trust models has prompted the development of security automation tools that enforce policies for authentication, authorization, and verification in CI/CD environments. These tools focus on minimizing trust at every stage of the software delivery process, which is crucial in preventing attacks like dependency confusion, software tampering, and privilege escalation.

At the same time, the need for tamper-evident logging and audit trails became more pronounced in light of high-profile attacks such as the SolarWinds breach. These incidents highlighted the lack of transparent and immutable logs, which could have provided earlier detection of unauthorized changes and attacks [10].

3. Problem Statement

Adopting continuous integration and deployment (CI/CD) pipelines alongside Kubernetes-native environments has transformed how organizations develop, test, and deploy software. While these advancements enable faster delivery and innovation, they also introduce various security challenges. Despite the rise of DevSecOps, which emphasizes integrating security into the development pipeline, several critical issues remain unresolved. This section identifies key problems that must be addressed to strengthen the security of modern software supply chains and improve policy enforcement in cloud-native environments.

3.1 Lack of Comprehensive Supply Chain Visibility

One of the most pressing concerns in modern software development is the lack of visibility into the software supply chain. As CI/CD pipelines grow in complexity, integrating numerous external libraries, dependencies, and components, organizations often struggle to track the origin and integrity of every artifact within the pipeline. This lack of visibility increases the risk of attacks like dependency confusion, where attackers inject malicious dependencies into the pipeline, exploiting the complexity of third-party packages.

Without robust tracking, it becomes difficult to verify the authenticity of each artifact, leaving the system vulnerable to the introduction of compromised code. To mitigate these risks, organizations must adopt solutions that ensure full visibility into the supply chain. They offer mechanisms to trace the provenance and integrity of every software artifact from its source to production deployment.

3.2 Inconsistent Policy Enforcement in Kubernetes

With its flexibility and scalability, Kubernetes has become the de facto platform for orchestrating containerized workloads. However, its open and customizable nature introduces significant security risks. While tools such as Open Policy Agent (OPA) and Kyverno offer mechanisms for policy enforcement, organizations face challenges in consistently integrating and applying these tools across all stages of the development pipeline.

Kubernetes environments can become a target for misconfigurations, such as insecure resource definitions or improper access controls, if policies are not consistently enforced. This lack of automated, comprehensive policy enforcement makes it more difficult for teams to ensure that only secure and compliant configurations are deployed to the production environment. There is a pressing need for a more streamlined, Kubernetes-native policy enforcement approach that seamlessly integrates into CI/CD workflows, ensuring continuous validation and reducing human error.

3.3 Insufficient Integrity Verification of Artifacts

A critical vulnerability in modern CI/CD workflows is the insufficient verification of software artifacts. Container images, binaries, and other deployment artifacts are typically generated and passed through multiple stages within the pipeline, often without rigorous integrity checks. Without tools that can provide cryptographic attestations or tamper-evident mechanisms, it becomes impossible to verify that the deployed artifacts have not been altered or compromised during the build process.

Malicious actors can exploit this gap by inserting backdoors or malicious code into artifacts, leading to potentially catastrophic security breaches. As software becomes more complex and the scale of CI/CD pipelines increases, automated methods for ensuring the integrity of software artifacts are essential. These tools must be able to authenticate and verify the authenticity of each artifact to maintain trust in the pipeline's outputs and prevent the deployment of compromised software.

3.4 Challenges in Adopting Zero-Trust Security Models

Zero-trust security has gained significant attention as a way to protect the software supply chain from internal and external threats. Zero-trust assumes that no entity, inside or outside the network, should be trusted by default. This paradigm is particularly relevant to CI/CD environments, where malicious actors can compromise any pipeline stage. However, implementing zero-trust principles across a distributed, cloud-native environment such as Kubernetes is challenging.

Organizations often struggle to ensure that every actor in the CI/CD pipeline is continuously authenticated and that the integrity of each artifact is validated. Moreover, enforcing zero-trust policies requires sophisticated tools for authentication, authorization, and auditing, which many organizations lack. Achieving a true zero-trust model in modern software delivery pipelines necessitates comprehensive automation, continuous verification of identities, and robust policy enforcement mechanisms to prevent unauthorized access and maintain the security of the entire pipeline.

4. Proposed Solutions

To address the security challenges identified in the previous section, organizations must implement a comprehensive set of solutions that enhance visibility, enforce policies, validate the integrity of software artifacts, and adopt zero-trust principles throughout their CI/CD pipelines. The proposed solutions aim to secure modern software supply chains by integrating automated security controls, improving artifact verification processes, and strengthening policy enforcement. Below are the proposed solutions that directly address the four key problems identified in the problem statement.

4.1 Improving Supply Chain Visibility with Traceability

A lack of visibility into the software supply chain is one of the primary risks in modern development pipelines. To mitigate this issue, organizations must adopt traceability frameworks that enable them to track the full lifecycle of every software artifact, from source code to deployment. Adopting tools such as in-toto can significantly improve supply chain visibility by enabling cryptographic attestations at every pipeline stage. This solution ensures that metadata about the build process—such as the parties involved, testing stages, and artifact signing—is securely recorded and associated with the artifacts.

By integrating Tekton Chains with in-toto, organizations can capture metadata at each build step, providing an immutable record of artifact provenance. This enables teams to verify that software artifacts come from trusted sources and have not been tampered with during the build process. Additionally, integrating metadata capture into the CI/CD pipeline ensures that every artifact is traceable throughout its lifecycle. This solution reduces the risk of dependency confusion attacks and ensures the deployed software is authentic and secure.

To further enhance visibility, organizations should also consider implementing tools that track third-party dependencies and monitor any changes or updates to external packages. Using tools such as Grype or Trivy, which scan dependencies for vulnerabilities, can provide additional insight into the software supply chain, allowing teams to proactively identify and mitigate risks from external components.

4.2 Strengthening Policy Enforcement in Kubernetes

Kubernetes' flexibility and scalability make it a powerful platform for deploying containerized workloads, but without robust policy enforcement, its openness also introduces significant security risks. Automated, consistent policy enforcement in Kubernetes environments is critical to prevent misconfigurations, insecure resource definitions, and unvalidated deployments.
Organizations should adopt policy engines such as Open Policy Agent (OPA) and Kyverno to address this issue to enforce security best practices and compliance requirements. OPA, in particular, enables the creation of custom policies using the declarative Rego language, allowing security teams to define rules that govern the security posture of Kubernetes environments. For example, OPA can be used to ensure that only authorized containers are allowed to run, prevent the use of insecure image registries, and enforce compliance with security configurations such as network policies and access controls.

Kyverno, as a Kubernetes-native policy engine, offers a more intuitive solution for organizations that prefer to work with YAML configuration files. Kyverno integrates seamlessly with Kubernetes workloads and allows for the creation of policies that validate, mutate, or block resources based on predefined rules. Its focus on native resource handling makes it easier for Kubernetes operators to adopt and manage policy enforcement, reducing the operational overhead.

Combining OPA or Kyverno with Gatekeeper, an extension of OPA that integrates directly with the Kubernetes API, ensures that policies are enforced from the very beginning of the CI/CD pipeline. By integrating policy enforcement early in the pipeline, teams can automatically prevent insecure or non-compliant configurations from reaching production, minimizing the risk of deployment-related vulnerabilities.

4.3 Ensuring Integrity Verification of Artifacts

The integrity of software artifacts is a fundamental concern in modern CI/CD pipelines, as compromised or tampered artifacts can introduce significant security risks. To address this challenge, organizations must implement robust artifact verification mechanisms that ensure every software component is genuine and has not been altered during the build process.

Tekton Chains, as part of the Tekton Pipelines framework, plays a key role in ensuring the integrity of artifacts by automatically generating cryptographic attestations during the build process. These attestations, which are linked to the build steps in the pipeline, allow consumers to verify that an authorized party created the artifact and that the build process was not compromised. Using cryptographic signatures and metadata, Tekton Chains integrates seamlessly with the CI/CD pipeline, enabling automated verification at each step of the delivery pipeline.

Organizations should also adopt cosign and Rekor, tools designed to sign and verify artifacts. Cosign provides secure, keyless signing and verification of container images, ensuring that artifacts are traceable and cannot be tampered with. Rekor, a transparent logging service, stores the signatures and attestations created by cosign, offering an immutable record of all verified builds and artifacts. Together, these tools provide end-to-end artifact verification, ensuring that only trusted artifacts are deployed to production.

Combining these tools with automated build pipelines, organizations can ensure that every artifact generated, whether a container image or binary, is verified for integrity before it is deployed, minimizing the risk of introducing malicious code into production environments.

4.4 Adopting Zero-Trust Principles in CI/CD Pipelines

Zero-trust security, which operates on the principle of never trusting any entity by default, is an essential paradigm for securing modern software supply chains. Implementing zero-trust principles in CI/CD environments is vital for mitigating internal and external threats, particularly those arising from compromised credentials or unauthorized access to the pipeline.

To implement zero-trust security in a Kubernetes-based CI/CD pipeline, organizations should leverage automated identity and access management (IAM) tools to enforce strong authentication and authorization at every step. HashiCorp Vault and Kubernetes RBAC (Role-Based Access Control) provide robust mechanisms for managing and controlling access to resources within the pipeline. Vault, in particular, allows organizations to securely manage secrets and identities, ensuring that only authorized users and systems can access critical resources.

Furthermore, integrating continuous verification into the CI/CD pipeline through tools like OPA or Kyverno ensures that every action, from code, commits to deployments, is verified and authenticated. Policies should be designed to enforce least-privilege access, limiting the permissions granted to users and services to the minimum necessary for performing their tasks. This minimizes the attack surface and helps prevent privilege escalation attacks.

To further enhance security, organizations should also implement audit logging and monitoring systems, providing real-time visibility into CI/CD pipeline activities. Logs should be tamper-evident and stored in immutable formats to ensure that any unauthorized changes can be traced and detected early.

5. Conclusion

The security of software supply chains has become a critical concern as organizations increasingly rely on automated CI/CD pipelines and containerized environments like

Kubernetes for rapid software delivery. The risks introduced by vulnerabilities, misconfigurations, and unauthorized access to sensitive artifacts are significant and growing, as evidenced by recent high-profile cyberattacks. As a result, organizations must implement robust, automated security controls throughout the development lifecycle to safeguard the integrity of their software.

This paper has highlighted the major security challenges modern development environments face, including limited visibility into the software supply chain, inadequate policy enforcement in Kubernetes-native environments, the need for stronger artifact verification, and the adoption of zero-trust principles. These challenges, if left unaddressed, leave organizations vulnerable to attacks that can undermine the trustworthiness and integrity of their software.

The proposed solutions—ranging from improved traceability through tools like Tekton Chains and in-toto, to stronger policy enforcement using Open Policy Agent (OPA) and Kyverno, to the integration of artifact verification mechanisms like cosign—offer a comprehensive strategy for securing the software supply chain. Adopting zero-trust security principles and integrating automated identity and access management tools ensures that no entity within the CI/CD pipeline is trusted by default, significantly reducing the potential attack surface.

By integrating these solutions into their development pipelines, organizations can enhance the security of their software delivery processes, reduce the risk of exploitation by malicious actors, and ensure compliance with security best practices. This approach mitigates the risks associated with insecure configurations and tampered artifacts and lays the foundation for building trust with customers and stakeholders by demonstrating a commitment to secure software development.

In conclusion, securing the modern software supply chain requires a multifaceted approach combining visibility, automation, policy enforcement, and trust validation at every CI/CD pipeline stage. Implementing the solutions outlined in this paper provides a roadmap for organizations to achieve a secure, efficient, and compliant software development process, ultimately ensuring the safe delivery of applications and services.

References

[1] Rangnau, T., Buijtenen, R.V., Fransen, F., & Turkmen, F. (2020). Continuous Security Testing: A Case Study on Integrating Dynamic Security Testing Tools in CI/CD Pipelines. *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)*, 145-154.

[2] Guides: Kubernetes admission control. (n.d.). Open Policy Agent. https://www.openpolicyagent.org/docs/v0.12.2/guides-kubernetes-admission-control/

[3] Jonas, K. (2019). Kubernetes Policy Enforcement with Open Policy Agent Gatekeeper. *InfoQ*. https://www.infoq.com/news/2019/09/opa-gatekeeper-kubernetes/

[4] Preuveneers, D., & Joosen, W. (2019, June). Towards multi-party policy-based access control in federations of cloud and edge microservices. In 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW) (pp. 29-38). IEEE.

[5] Ahmed, M. (2020). Integrating open policy agent (OPA) with kubernetes. Medium. https://medium.com/swlh/integrating-open-policy-agent-opa-with-kubernetes-a912bac9168d

[6] Torres-Arias, S., Afzali, H., Kuppusamy, T. K., Curtmola, R., & Cappos, J. (2019). in-toto: Providing farm-to-table guarantees for bits and bytes. In 28th USENIX Security Symposium (USENIX Security 19) (pp. 1393-1410).

[7] Aravena, R. (2019). Let Your Software Supply Chain Ride with Kubernetes {CI/CD}. https://www.usenix.org/conference/lisa19/presentation/aravena

[8] Martin, R. A. (2020, July). Visibility & control: addressing supply chain challenges to trustworthy software-enabled things. In *2020 IEEE Systems Security Symposium (SSS)* (pp. 1-4). IEEE.

[9] Imeri, A., Agoulmine, N., Feltus, C., & Khadraoui, D. (2019). Blockchain: Analysis of the New Technological Components as Opportunity to Solve the Trust Issues in Supply Chain Management. *Advances in Intelligent Systems and Computing*.

[10] Alert, C. I. S. A. (2020, December). Advanced persistent threat compromise of government agencies, critical infrastructure, and private sector organizations.