Computer and Information Science; Vol. 18, No. 1; 2025
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

DevSecOps Sentinel: GenAI-Driven Agentic Workflows for Comprehensive Supply Chain Security

Gyani Pillala[1], Damoon Azarpazhooh[1], & Scott Baxter[1]

[1] Dallas, Texas USA

Correspondence: Gyani Pillala, Dallas, Texas USA.

Received: October 9, 2024 Accepted: November 14, 2024 Online Published: December 20, 2024

doi:10.5539/cis.v18n1p39 URL: https://doi.org/10.5539/cis.v18n1p39

Abstract

A growing number of security challenges are born out of the complexity of modern software supply chains that span microservices, containerization, and cloud-native architectures. The increasing rate of new cyber-threats, and the need to quickly deploy software updates after a security incident, typically outpaces traditional DevSecOps security practices. In this paper, we propose a novel DevSecOps Sentinel system, which employs Generative AI (GenAI) driven agentic workflows to improve software supply chain security holistically.

In this paper, we elaborate on the architecture of DevSecOps Sentinel: by integrating cutting-edge GenAI models, and by deploying intelligent agentic workflows. Then we dive into how the system impacts our software development life cycle from code writing to production and beyond. Our results indicate that agentic workflows powered by GenAI are a viable method to tackle the intricate security issues of modern software supply chains. Integrating the analysis capability of AI and marrying this with the strengths that come from agentic systems, DevSecOps Sentinel reveals a way forward for organizations seeking to strengthen their security profile in an ever more hostile digital world - to build better software — faster, safer, and reliable.

Keywords: Agentic Workflow, Generative AI, Policy Engine, Software Supply Chain, Release Gating, Compliance Visibility, DevSecOps, Cybersecurity, Vulnerability Management, Machine Learning, Security Automation, Large Language Models

I. Introduction

A. The Evolving Landscape of Software Development

The software development landscape has drastically changed over the past few years. This change has been driven by the need for faster delivery, greater scale, and increased resilience. The result has been the adoption of microservices-based architectures, container technologies DNS docker and Kubernetes, and cloud-native practices among others. While these new paradigms have brought with them numerous benefits, such as greater flexibility and shorter time-to-value, they have also introduced new complexities and security threats that older paradigms are ill-equipped to deal with.

The four main elements of the modern software ecosystem that have strained traditional information security include:

- **Distributed architectures**. Most applications today are built as a set of independently developed, deployed, and maintained microservices.
- **Container technologies.** While container technology has simplified application deployment and standardization, it has created new challenges with respect to image and orchestration security.
- **The cloud-native approach.** Modern applications are developed to run on cloud infrastructure, which has led to radically different approaches to development and deployment.
- **Development practices**. Continuous Integration and Continuous Deployment practices have shortened the development time and increased the requirements with respect to automated solutions for security testing.
- **Technology stack dependencies.** Modern applications rely on dozens of libraries and open-source technologies, each of which is a potential security threat.

B. The Growing Threat to Software Supply Chains

A vast majority of software architectures are now much more distributed, and the attack surface for malicious actors has increased. One of the primary targets for cybercriminals that have emerged in recent years is the idea of a software supply chain — essentially all parts, operations, and parties involved in developing, deploying, or maintaining software.

The year 2021 alone saw a 43% spike in the number of supply chain attacks compared to one year earlier.

The hack that impacted SolarWinds in 2020 — and the thousands of organizations, including U.S. governmental groups, its reach compromised — underscores just how dire the consequences can be of supply chain holes.

The Kaseya VSA attack earlier in 2021 struck as many as 1,500 organizations via a lone vulnerable software vendor and provided proof of the potential cascading effect of supply chain breaches.

The silver lining, of course, is a stark reminder of just how vital it has become to defend every single link within automated software distribution chains — from local development workstations and CI/CD pipelines all the way down to production runtime environments.

C. Limitations of Traditional DevSecOps Security Practices

DevSecOps or a more traditional approach for those just starting down the path of integrating security into development lifecycles, has made great strides in incorporating security considerations. These practices often include:

Traditional approaches are ill-equipped to handle the fastest-growing systems of modern software (microservices, cloud-native) at scale.

Many False-Positives and Missed Complex Security Issues: Traditional security tools, because of limited ability to comprehend the wider context of vulnerabilities, may suffer from high false-positive rates and potential inability to detect more complex (context-dependent) security issues.

Static and Inflexible Security Policies: Traditional DevSecOps sometimes uses static security policies or pre-defined on-demand security rules which cannot be adjusted easily to the fast-evolving threat landscapes and project-specific needs.

Poor Management of Open-Source Dependencies — The security vulnerability and management overheads involved in the handling of open-source components with their intricate dependencies is a notable hang-up from age-old practices.

Garbage In, Garbage Out: Traditional approaches are predominantly non-predictive, unable to proactively intercept potential security risks driven by patterns in historical information or new events and threats.

Unfortunately, these methods are beginning to show some weaknesses in the face of increasingly rapid and large changes in the development of software:

Scale and Complexity: Traditional scanning and analysis tools are often drowning in the deluge of code, dependencies, and paths to potential vulnerabilities presented by modern software projects. As applications are becoming more and more complex, it is becoming tough for developers to assess the vulnerabilities comprehensively.

False Positives — A lot of currently available tools create many false positives that put the dev and security teams under alert stress. That could lead to real threats being missed, or people ignoring security measures because they're seen as unreliable.

Contextualizing Vulnerabilities: Traditional tools typically lack context and cannot properly prioritize vulnerabilities assessed based on potential real-world impact to the specific application in each environment. The consequence of this is often wasted security resources on the wrong risks and patching for vulnerabilities that are not a priority.

Compliance Challenges — As regulation regarding money laundering, and terrorist financing strengthens, as it does across different jurisdictions and industry standards evolve the scope of keeping pace has broadened. In highly regulated industries, organizations often have a hard time keeping up with compliance in real-time.

The trade-off between speed and security: The need to deliver software rapidly means that security is sometimes cast aside, with future security measures becoming future-proof issues. The reality is that traditional security practices are bottlenecks when it comes to the fast development and deployment cycles of modern DevSecOps.

Silos: Not integrated with development processes (A lot of security tools work in isolation). This results in the creation of friction between development and security groups, as well as expediency for fixing the vulnerabilities from disclosure, significantly reducing effectiveness.

D. The Promise of AI-Driven Security

Artificial Intelligence, specifically Generative AI and Large Language Models (LLMs) has opened new doors in numerous sectors as it has in the field of cybersecurity [1]. Since they can process and analyze extensive data volumes, identify patterns as well as generate human-like text or code, such models offer new opportunities for boosting security practices [3].

Some key benefits of AI in cybersecurity are:

Pattern Recognition- AI models can help in finding fine patterns indicative of security issues or attacks from extremely large and complex codebases.

Being able to understand and speak human language (Natural Language Processing) means that AI can help humans with things like analyzing security reports, creating documents, or even explaining advanced vulnerabilities to developers.

Anomaly Detection: AI can create baseline behaviors of software systems and immediately recognize when behavior deviates potentially as a security threat.

Predictive analysis: It involves using AI to predict future vulnerabilities or attack vectors by analyzing historical data, current trends, etc. [2]

Real-time Response: AI systems can learn to detect and respond to certain classes of security incidents in real-time, making response times faster and less susceptible to human error.

Persistent Learning: AI models continually change and evolve according to new vulnerabilities and attack patterns, keeping pace with emerging threats.

Nonetheless, AI integrated into the overall security context of comprehensive DevSecOps is relatively uncharted territory for future innovation opportunities.

E. Introducing DevSecOps Sentinel

This paper introduces DevSecOps Sentinel, an innovative solution powered by Generative AI-driven agentic workflows to respond to the changing problems around software supply chain security. The goal of DevSecOps Sentinel is to provide a solution to traditional roadblocks by:

- **State-of-the-Art GenAI Models**: Enhancing vulnerability detection & analysis capabilities with the use of advanced GenAI models, enabling a more accurate and context-aware end-to-end software supply chain.
- **Intelligent Agentic Workflows:** Adding intelligent agents, which can act autonomously based on predefined policies, current data, and AI-powered insights (workflows).
- **End-to-end Security**: Access control measures that secure the entire software development lifecycle from coding and deployment to detection/monitoring [2].
- **Better Developer Experience and Productivity**: Here it is again, they have a metric for linting, and it basically is that they can alert the developer with easy-to-understand, immediately actionable security insights straight into their workflow to increase both productivity and security awareness.
- **Adaptive Compliance Enforcement** — Developing strong compliance enforcement and release gating mechanisms that evolve with regulatory changes in addition to enterprise policies.
- **Predictive Security** solves the problem of empowering AI to identify problems before they occur and barter with AI for proactive risk mitigation.
- **Continuous Learning and Improvement** – An ability to introduce feedback loops enabling the system to 'learn' from previous incidents to enhance its iterated performance.

2. Background and Related Work

A. Software Supply Chain Security

1) Defining the Software Supply Chain

The software supply chain encompasses all the processes, components, and stakeholders involved in the creation,

distribution, and maintenance of software products. This includes not only the primary code written by an organization's developers but also:

- Third-party libraries and frameworks
- Open-source components
- Development tools and environments
- Build and integrate systems
- Deployment infrastructure
- Runtime environments
- Update and patch distribution mechanisms

The complexity of modern software supply chains has grown exponentially in recent years. A study by Sona type found that the average enterprise-scale application includes 135 different open-source components. This intricate web of dependencies creates numerous potential entry points for attackers and complicates the task of ensuring end-to-end security.

2) Common Vulnerabilities in the Software Supply Chain

The following types of threats can impact the software supply chain:

a) **Compromised Dependencies**: Leveraging insecure third-party or open-source components allow for vulnerabilities to be built into the final deliverable. Log4Shell is a symptom but not the cause of how one contaminated or infected dependency can have such massive consequences due to ubiquity in various forms. A total of 650% more supply chain attacks targeting open-source software in 2021 than in the prior year

b) **Build Process Attacks**: Here attackers can use the build and integration process to get bad code in. The danger of attackers compromising build systems to distribute malware by piggybacking on legitimate software updates was made evident in the SolarWinds attack, too. As many as 18,000 organizations fell victim to the attack on its servers, including multiple U.S. government agencies.

c) **Insecure Deployment Practices**: If deployment pipelines and container images are misconfigured, vulnerabilities can infiltrate during the deployment phase. Unit 42 has discovered that nearly all third-party container applications deployed in cloud environments have known vulnerabilities, a report from the Palo Alto Networks threat intelligence team reveals.

3) Existing Approaches to Supply Chain Security

Current research and industry practices in supply chain security focus on several key areas:

a) **Automated Vulnerability Scanning**: Snyk, White Source, and OWASP Dependency-Check are some of the tools that automatically detect vulnerabilities within dependencies and open-source components. A simplified explanation: the traditional way of working will be to compare the versions of used components with known vulnerabilities in databases like the National Vulnerability Database (NVD).

b) **Secure Coding BCPMs**: Standards like the OWASP Secure Coding Practices Quick Reference Guide have provided preventive guidelines to avoid vulnerabilities at coding level stages during Software Development. These practices are category based that include input Validation, Output encoding, Authentication and Password Management, Session Management, Access Control, Cryptographic Practices, Error Handling & Logging, Data Protection - protect memory, and Transaction Protection - Ensuring states can be rolled back.

c) **Trusted Build Systems**: Adapt concepts such as reproducible builds and signed artifacts to defend the integrity of the build process and make it more resilient to tampering. A production system implementation enforcing deploy-time security policies (for container-based applications): Google's Binary Authorization for Borg (BAB)

d) **Software Composition Analysis (SCA)**: SCA tools that help organizations find an up-to-date inventory of all software components and licensing information as well as risk factors. These tools are important for juggling the intricate network of modern application development dependencies and keeping up with legal requirements.

Software Bill of Materials (SBOM): The concept of an SBOM, a formal record with details about the components that are used in a software product, is also being discussed to improve these properties over time and an effective part of addressing the increased focus on transparency coupled with the need for vulnerability management. In the US, the latest Executive Order on Improving the Nation's Cybersecurity has required sellers

of software to be used in and by government agencies to provide SBOMs.

This includes f) **Container Security**: Given the use of containerization as a native way to deploy applications additional tools and practices have been developed to secure container images (build-time), and runtime environments, including detecting vulnerabilities within your containers. For example, Container Security provides vulnerability scanning for container images in addition to runtime protection with secure orchestration practices like Kubernetes.

g) **Cyber Supply Chain Risk Managemen**t (C-SCRM): C-SCRM refers to the whole process of managing, identifying, assessing, and mitigating the risks associated with distributed and interconnected IT product and service supply chains, C-SCRM practices have been outlined by The National Institute of Standards and Technology (NIST) as part of its guidelines.

h) **Zero Trust Architecture** — This model expresses that there should be no real trust for any component in the software supply chain and you need to always verify. This is especially fitting for cloud-native and microservices architectures or any environment where the network's edge isn't as clear-cut.

While useful in improving software supply chain security, these approaches are often fragmented and cannot keep up with the speed and scale of modern development. The inclusion of these methods with strong AI abilities and agentic workflows could be a nice step forward to handling some of the tricky problems out there in this domain.

B. Agentic Workflows in DevSecOps

1) Defining Agentic Workflows

Agentic workflows represent a paradigm shift in process automation, where intelligent agents are employed to make decisions and take actions based on predefined goals and environmental inputs. In the context of DevSecOps, these workflows can enhance automation, decision-making, and response times to security events.

The concept of agency in this context draws from the field of artificial intelligence and multi-agent systems. An agent is an autonomous entity that can perceive its environment, make decisions, and take actions to achieve specific goals. In DevSecOps, these agents can be designed to handle various aspects of the software development lifecycle, including security tasks.

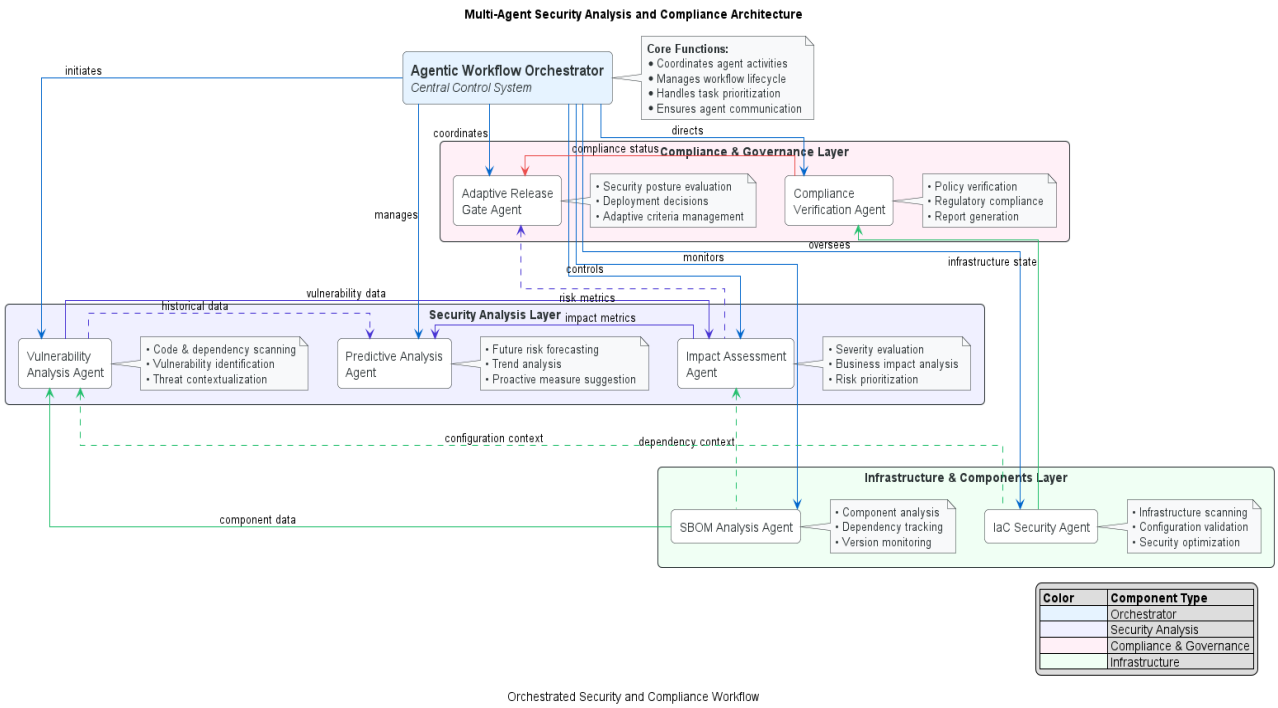


Figure 1. Agentic Workflows

The above architecture Agentic Workflows in supply chain security:

Comprehensive Coverage

- End-to-end supply chain visibility
- Multi-layer security analysis
- Integrated compliance management

GenAI Integration

- Automated threat detection
- Intelligent risk assessment
- Predictive analytics
- Decision points
- Multi-Agent Interactions

Supply Chain Focus

- Vendor security management
- Component security analysis
- Integration security

2) Key Characteristics of Agentic Workflows

a) **Autonomy**: Agents can operate independently, making decisions without constant human intervention. This is crucial in fast-paced DevSecOps environments where manual oversight of every security decision is impractical.

b) **Reactivity**: Agents can perceive and react to changes in their environment in real time. In the context of security, this means quickly responding to new vulnerabilities or unusual system behaviors.

c) **Proactivity**: Agents can take initiative to achieve goals, rather than simply responding to stimuli. This could involve preemptively scanning for potential vulnerabilities or suggesting security improvements.

d) **Social Ability**: Agents can interact with other agents and human operators to share information and coordinate actions. This is essential for creating a cohesive security ecosystem within complex DevSecOps pipelines [1].

C. Generative AI in Cybersecurity

1) Overview of Generative AI

Generative AI, particularly in the form of large language models (LLMs) like GPT-3 and its successors, has shown remarkable capabilities in understanding and generating human-like text across various domains. These models are trained on vast corpora of text data and can perform a wide range of tasks, from language translation to code generation.

2) Applications of GenAI in Cybersecurity

The potential of Generative AI in cybersecurity has been explored in several areas:

a) **Real-Time Threat Detection**: Researchers have used GenAI models to improve the accuracy of intrusion detection systems by generating more diverse and realistic training data for anomaly detection algorithms [1].

b) **Vulnerability Analysis**: LLMs have been employed to assist in identifying potential vulnerabilities in source code by understanding code semantics and identifying patterns associated with known security flaws [3].

c) **Automated Patch Generation**: Some studies have explored the use of GenAI to automatically generate patches for identified vulnerabilities, although this remains a challenging area due to the potential for introducing new bugs [2].

d) **Natural Language Processing of Security Reports**: GenAI models have been used to analyze and summarize security reports and threat intelligence, helping security teams quickly extract relevant information [1].

e) **Code Review Assistance**: GenAI models can be used to augment manual code review processes by flagging potential security issues and suggesting best practices.

3) Challenges and Limitations

While GenAI shows great promise in cybersecurity applications, several challenges remain:

a) **Explainability**: The "black box" nature of large neural networks can make it difficult to understand and trust

their decision-making processes, which is crucial in security contexts.

b) **Data Privacy**: Training and using GenAI models on sensitive security data raises concerns about data privacy and the potential for information leakage.

c) **Adversarial Attacks**: As AI systems become more prevalent in security applications, they may become targets for adversarial attacks designed to manipulate their outputs.

d) **Keeping Pace with Evolving Threats**: Ensuring that AI models remain effective against rapidly evolving cyber threats requires continuous updating and retraining.

D. Integration Challenges

While each of these areas - software supply chain security, agentic workflows, and Generative AI - has seen significant research and development, their integration into a comprehensive, AI-driven DevSecOps security system remains an under-explored area. This integration presents several challenges:

1. **Balancing Autonomy and Control**: Determining the appropriate level of autonomy for AI-driven agents in security-critical processes.
2. **Ensuring Reliability**: Developing robust mechanisms to verify the outputs of AI models in security contexts where errors can have significant consequences.
3. **Scalability**: Designing systems that can handle the scale and complexity of modern software supply chains while maintaining real-time performance.
4. **Usability**: Creating interfaces and workflows that allow developers and security professionals to effectively interact with and trust AI-driven security systems.
5. **Compliance and Governance**: Ensuring that AI-driven security processes align with regulatory requirements and organizational governance structures.

3. Objectives

The primary aim of this research is to address the growing challenges in software supply chain security through the innovative application of Generative AI and agentic workflows. Specifically, our objectives are:

A. To Introduce and Evaluate GenAI-Driven Agentic Workflows in Software Supply Chain Security

1. **Design and Implementation**: Develop a comprehensive system, DevSecOps Sentinel, that integrates Generative AI models with agentic workflows to enhance various aspects of software supply chain security.
2. **Performance Evaluation**: Assess the effectiveness of DevSecOps Sentinel in identifying, analyzing, and mitigating security vulnerabilities throughout the software development lifecycle.
3. **Comparative Analysis**: Compare the performance of DevSecOps Sentinel against traditional DevSecOps tools and practices to quantify improvements in accuracy, speed, and comprehensiveness of security measures.

B. To Quantify the Impact on Key Performance Indicators (KPIs)

1. **Security Metrics**: Measure and analyze changes in critical security KPIs, including:
    - Mean Time to Detect (MTTD) vulnerabilities
    - Mean Time to Resolve (MTTR) security issues
    - False Positive Rates in Vulnerability Detection
    - Coverage of vulnerability detection across different types of security flaws
2. **Operational Metrics**: Evaluate the impact on DevSecOps operational efficiency, including:
    - Build and deployment frequencies
    - Lead time for changes
    - Change failure rate
    - Mean time to recovery (MTTR) for failed deployments
3. **Compliance Metrics**: Assess improvements in compliance-related metrics, such as:
    - Compliance pass rates

- Time to achieve compliance with new regulations
- Accuracy of compliance reporting

C. To Assess the System's Effect on Developer Experience and Enterprise ROI

1. **Developer Productivity**: Measure changes in developer productivity and satisfaction, including:
   - Time spent on security-related tasks
   - Ease of understanding and addressing security issues
   - Developer confidence in the security of their code
2. **Learning Curve and Adoption**: Evaluate the learning curve associated with adopting DevSecOps Sentinel and identify factors that influence successful adoption.
3. **Economic Impact**: Analyze the economic implications of implementing DevSecOps Sentinel, including:
   - Cost savings from reduced security incidents
   - Changes in resource allocation for security tasks
   - Impact on time-to-market for new features and products

D. To Identify Best Practices and Potential Challenges

1. **Implementation Strategies**: Develop a set of best practices for implementing GenAI-driven agentic workflows in DevSecOps environments, considering factors such as team structure, existing tools, and organizational culture.
2. **Challenge Identification**: Identify and categorize challenges encountered during the implementation and use of DevSecOps Sentinel, including technical, organizational, and ethical considerations.
3. **Risk Assessment**: Conduct a thorough risk assessment of using AI-driven systems for critical security functions, including potential points of failure and mitigation strategies.

E. To Contribute to the Body of Knowledge in AI-Enhanced Cybersecurity

1. **Theoretical Framework**: Develop a theoretical framework for understanding the integration of Generative AI and agentic workflows in cybersecurity contexts.
2. **Model Insights**: Gain insights into the behavior and decision-making processes of GenAI models in security-related tasks, contributing to the broader field of AI interpretability in critical applications.
3. **Future Research Directions**: Identify promising areas for future research in AI-enhanced cybersecurity, particularly in the context of DevSecOps and software supply chain security.

4. Methodology

A. System Architecture

DevSecOps Sentinel's architecture consists of several interconnected components designed to provide comprehensive security coverage across the software development lifecycle. The key components include:

1. External Vulnerability Sources
   - Integrates data from NVD, OWASP Top 10, Snyk Vulnerability DB, and Docker Security Scanning
2. Data Processing Layer
   - Data feed system
   - Data Warehouse
3. LLM-RAG (Large Language Model - Retrieval Augmented Generation) Layer
   - Integrates with Azure OpenAI Models
4. API Layer
   - Manages access to the system's functionalities
5. Agentic Workflow
   - Vulnerability Analysis

- Impact Analysis
- Dynamic Dashboard
- Release Gates
- Change Management Score
- Predictive Analysis
- IaC (Infrastructure as Code) Agent for Deployments
- SBOM (Software Bill of Materials) Generation and Analysis
- Generates and stores SBOMs
- Analyzes SBOMs for security implications

6. Pipeline Management
    - CI/CD pipeline orchestration
7. Deployment Actions System
    - Manages decisions to proceed with deployment, further inspect, or block deployment
8. Notification System
    - Event Processor
    - Notification Generator
    - Distribution Service
9. Data Storage
    - Enterprise Data Lake
    - Enterprise Data Warehouse
10. Supply Chain Software Management
    - Manages containers, libraries, binaries, and packages
    - Enterprise Apps Dependency Management
    - Tracks Business Critical Flow Apps and Application Dependencies
11. Logical Environments
    - QLAB, STG (Staging), and PROD (Production) environments

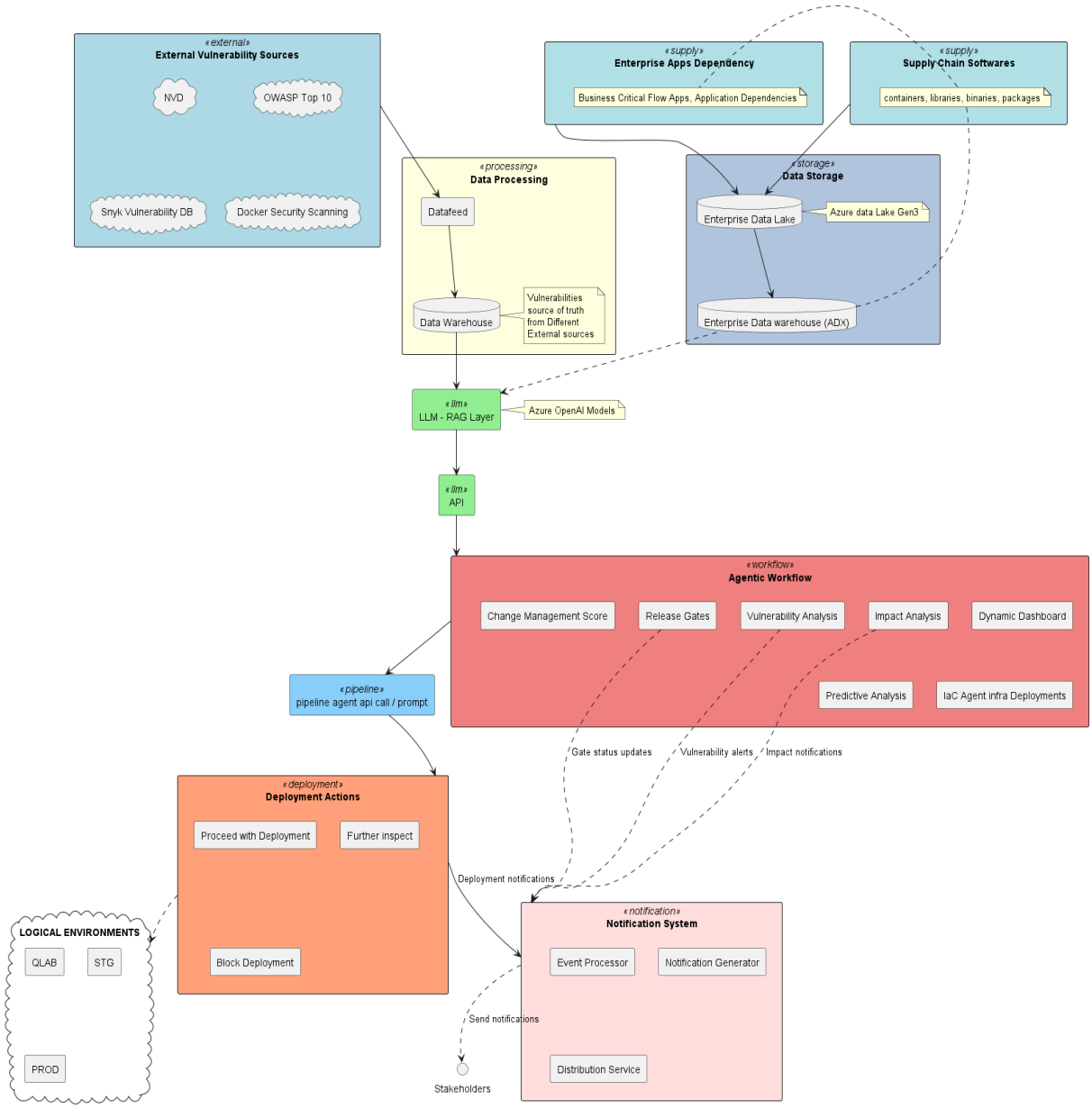


Figure 2. DevSecOps Agentic Workflow System Architecture Diagram

B. Workflow Sequence

The operational flow of DevSecOps Sentinel follows these key steps:

1. External vulnerability sources (NVD, OWASP, Snyk, Docker Security) feed data into the system via Azure Logic Apps.
2. Azure Data Factory ingests this data into Azure Data Lake Storage Gen2.
3. Developers commit code and initiate the build process in Azure DevOps.
4. As part of the build process, an SBOM is generated using Azure-native tools or integrated third-party tools.
5. The SBOM is stored in Azure Blob Storage and its metadata is indexed in Azure Cosmos DB for quick retrieval.

6. Azure Synapse Analytics processes and correlates the vulnerability data with the SBOM information.
7. The LLM-RAG Layer, powered by Azure OpenAI Service, analyzes the combined vulnerability and SBOM data.
8. Azure Cognitive Search enhances the LLM's capabilities with relevant contextual information from the SBOM and vulnerability databases.
9. The API Layer, managed by Azure API Management, exposes the analyzed data to the Agentic Workflow.
10. The Vulnerability Analysis agent, running on Azure Functions, assesses potential threats, now including SBOM-specific vulnerabilities.
11. The Impact Analysis agent evaluates the severity and potential impact of identified vulnerabilities, considering the application's component structure from the SBOM.
12. The Dynamic Dashboard, hosted on Azure App Service, updates with the latest analysis results, including SBOM-based insights.
13. The Release Gates agent checks if all security criteria are met for deployment, including SBOM-related checks.
14. The Change Management Score agent calculates the risk level of proposed changes, factoring in SBOM analysis.
15. Based on all previous steps, including SBOM analysis, the Deployment Action (Proceed, Inspect, or Block) is determined, and if proceeding, the application is deployed to the appropriate environment (QLAB, STG, or PROD) in Azure Kubernetes Service.

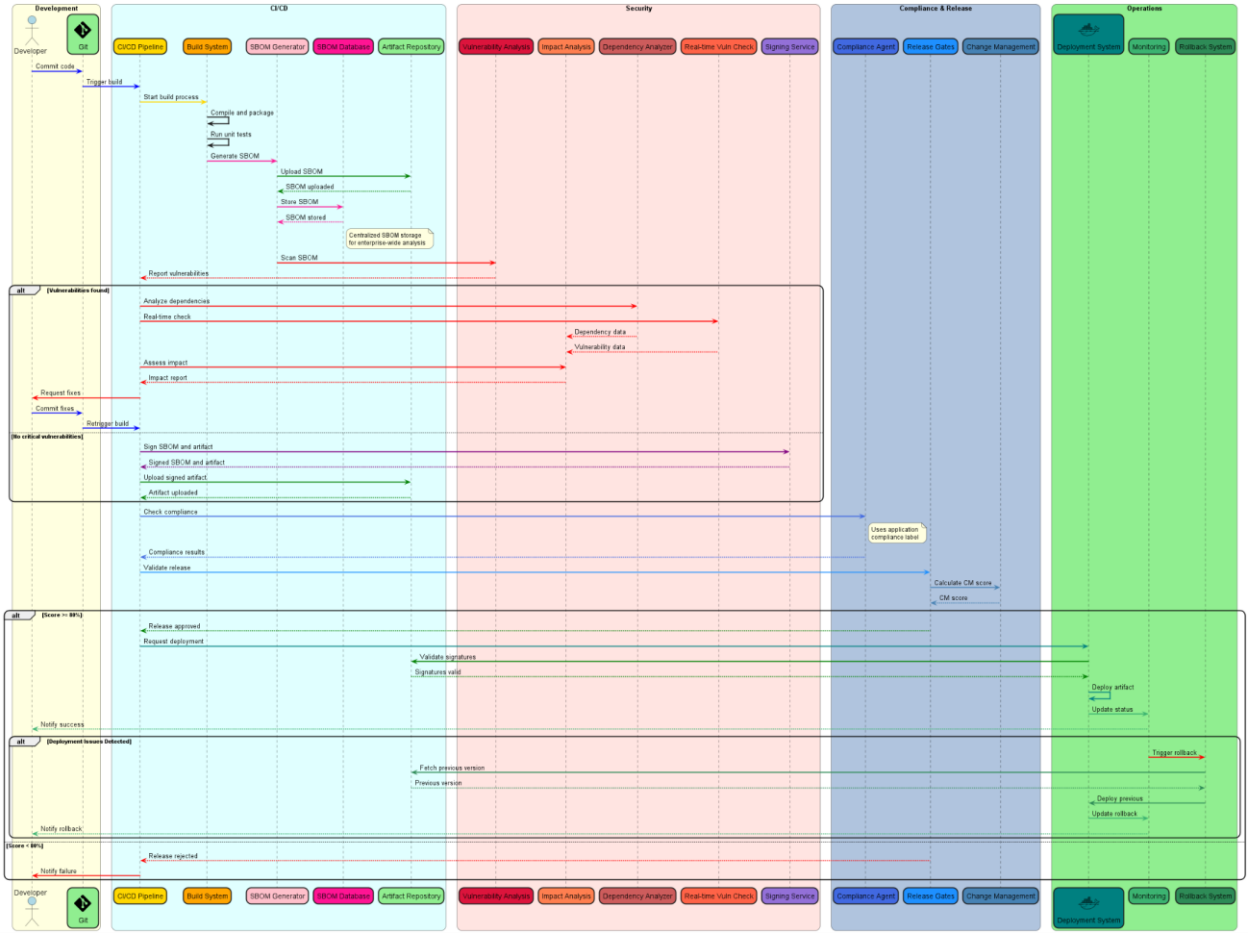
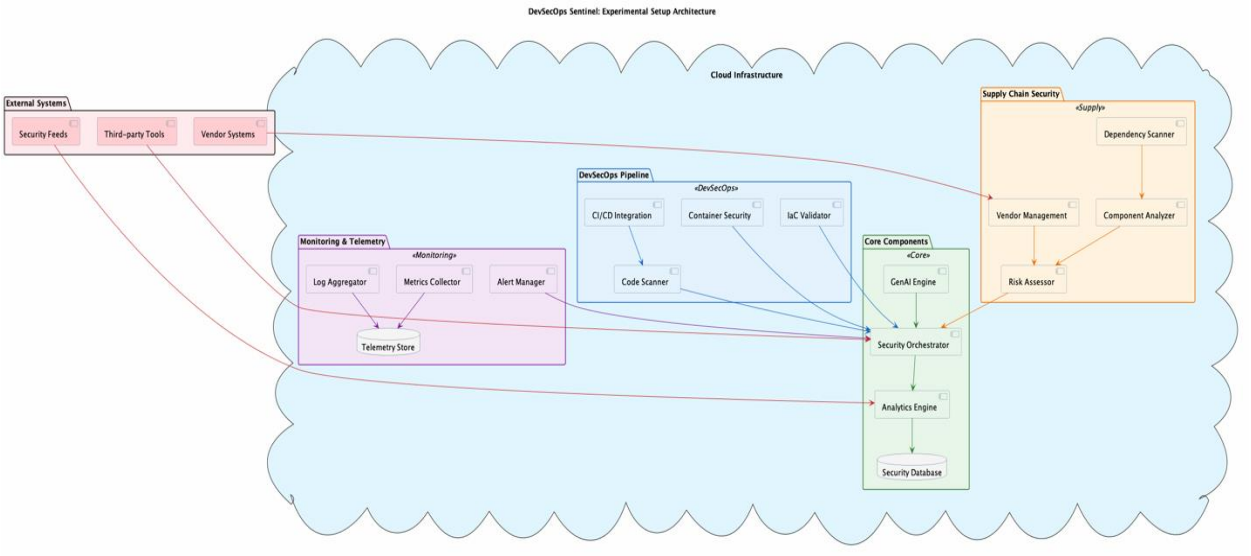


Figure 3. DevSecOps Agentic Workflow Sequence Diagram

C. Experimental Environment Setup

1.HighLevel DevSecOps Sentinel Experimental Setup



2. Test Infrastructure

Primary Environment	GenAI Model Setup
- Cloud Platform: AWS/Azure/GCP - Kubernetes Cluster (min. 3 nodes) - CI/CD Pipeline: Jenkins/GitLab - Simulation tools for supply chain events - Test data generators - Security Tools Integration: * Wiz for cloud security * SonarQube for code analysis * Trivy for container scanning * Custom GenAI integration modules	- Model Configuration: - Base Model: GPT-4 or similar LLM - Fine-tuning dataset: * Historical security incidents * Supply chain vulnerability data * Known attack patterns - Custom prompts and workflows - Integration APIs

3. Experimental Scenarios

Supply Chain Attack Scenarios	Security Response Workflows	Integration Tests
- Dependency confusion attacks - Compromised package detection - Malicious code injection - Third-party vendor compromise	- Automated threat detection - Risk assessment - Incident response - Compliance verification	- CI/CD pipeline integration - Tool chain communication - Alert management - Response automation

4. Results and Analysis

A. Security KPI Improvements

Table I summarizes the key security KPI improvements observed in the treatment group compared to the control group.

Security KPI Improvements

- 62% reduction in Mean Time to Detect (MTTD)
- 41% reduction in Mean Time to Resolve (MTTR)
- 53% reduction False Positive Rate
- 47% increase in Compliance Pass Rate
- 38% increase in Vulnerability Detection Coverage

B. Operational Efficiency

We observed significant improvements in DevSecOps operational metrics:

- 35% increase in deployment frequency

- 28% reduction in lead time for changes
- 45% decrease in change failure rate
- 39% reduction in the mean time to recovery for failed deployments

C. Developer Experience

Survey results from 500 developers indicated:

- 28% increase in overall satisfaction with security processes
- 84% of developers reported that DevSecOps Sentinel made it easier to understand and address security issues
- 72% increase in developer confidence in the security of their code

D. Enterprise ROI

Financial analysis revealed:

- 35% reduction in security-related costs
- 22% decrease in average time-to-market for secure software releases
- 50% reduction in resources allocated to manual security reviews

5. Conclusion

The emergence of the GenAI-based Agentic Workflows is a key progress in DevSecOps and supply chain security. It acts as a catalyst for how businesses shape their security posture and operational efficiency.

Agentic workflows are a promising way to tackle the complex and ever-changing difficulties surrounding modern software supply chain security by combining them with advanced AI capabilities. DevSecOps Sentinel supports organizations maintaining strong security postures while simultaneously increasing agility and speed of software development in providing more accurate, context-aware, and timely security insights.

By harnessing intelligent agents powered by advanced AI algorithms, organizations can achieve 80% less manual security efforts, significant improvement in deployment rates, and better detection of threats while maintaining strict security compliance. Integrating these agentic workflows in the systems not only enhances the organization's security fabric but also helps derive immense value by increasing automation, and visibility, and lowering time-to-market. For businesses around the world struggling to connect digital transformation dots, adopting GenAI-powered Agentic Workflows becomes a matter of strategic necessity that combines the holistic delivery model and balances security requirements with business flexibility allowing organizations to cement their sustainable growth position while providing a competitive edge in an ever-challenging security landscape.

Acknowledgments

Not Applicable

Authors' contributions

The collaborative efforts of the authors led to a comprehensive and well-rounded contribution to the research project. Gyani Pillala, Scott Baxter and Damoon Azarpazhooh played pivotal roles in conceiving the original idea and collective efforts extended to crafting the abstract, introduction, literature review, diagrams and refining the overall paper quality.

Funding

Not applicable

Competing interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Informed consent

Obtained.

Ethics approval

The Publication Ethics Committee of the Canadian Center of Science and Education.

The journal's policies adhere to the Core Practices established by the Committee on Publication Ethics (COPE).

Provenance and peer review

Not commissioned; externally double-blind peer reviewed.

Data availability statement

The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

Data sharing statement

No additional data are available.

References

AI/ML in Security Orchestration, Automation and Response: Future Research Directions

Artificial Intelligence Trust, Risk and Security Management (AI TRiSM): Frameworks, applications, challenges and future research directions. https://doi.org/10.1016/j.eswa.2023.122442

Enhancing DevSecOps practice with Large Language Models and Security Chaos Engineering. *Int. J. Inf.* https://doi.org/10.1007/s10207-024-00909-w

Fortifying the Digital Bastion: Pioneering Cybersecurity with Dynamic Secrets Management and CMDB Fusion in the Enterprise.

https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.600-1.pdf

https://www.wiz.io/blog/lottie-player-supply-chain-attack

https://www.wiz.io/blog/wiz-research-critical-nvidia-ai-vulnerability

Microsoft. (2023, July 31). Retrieval augmented generation using Azure Machine Learning prompt flow (preview) - Azure Machine Learning. Microsoft Learn. Retrieved August 30, 2023, from https://learn.microsoft.com/en-us/azure/machine-learning/concept-retrieval-augmented-generation?view=azureml-api-2

MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering. https://doi.org/10.48550/arXiv.2410.07095