International Journal of Recent Research and Review, Special Issues-2 - 2025
ISSN 2277 – 8322

AUTOSECAGENT: AN AUTONOMOUS AI AGENT FOR THREAT DETECTION AND MITIGATION IN CI/CD PIPELINES

Vimal Daga
CTO, LW India | Founder, #13 Informatics Pvt Ltd
LINUX WORLD PVT. LTD.

Preeti Daga
CSO, LW India | Founder, LWJazbaa Pvt Ltd
LINUX WORLD PVT. LTD.

Vivek Kumar
Research Scholar
LINUX WORLD PVT. LTD.

Abstract- The widespread adoption of DevOps practices and Continuous Integration/Continuous Deployment (CI/CD) pipelines has transformed software delivery to become more agile and automated. The same change has also increased the attack surface, and CI/CD workflows have become an attractive target for cyberattacks like secret leakage, dependency poisoning, code injection, and privilege escalation. Conventional security tools tend to fail to identify complex, real-time threats in dynamic DevOps environments. This study suggests an AI-based model for proactive threat alerting in CI/CD pipelines through machine learning (ML) and generative AI models. The model plugs into existing popular CI/CD tools such as GitHub Actions and GitLab CI to track code commits, configuration changes, build artifacts, and deployment patterns. Anomaly detection models learn from past pipeline data to detect abnormality, while large language models (LLMs) aid real-time vulnerability analysis, threat explanation, and remediation recommendations. By means of a prototype implementation and empirical analysis utilizing public DevOps datasets and simulated attack sets, the system in question shows a substantial early threat detection improvement, fewer false positives, and faster incident response. This work closes the loop between AI and DevSecOps by providing a scalable and smart solution to secure contemporary software delivery pipelines.

Keywords: Autonomous AI Agents, Agentic AI, DevSecOps, CI/CD Security, Threat Detection, Cybersecurity Automation, Software Supply Chain; Machine Learning in Security, Secure Deployment Pipelines, Large Language Models (LLMs).

I. INTRODUCTION

Rise of DevOps and continuous integration/continuous deployment (CI/CD) pipelines have changed software development by speeding up delivery cycles and deploying more frequently. Nevertheless, this velocity and automation have widened the attack surface as well, presenting massive vulnerabilities in software supply chains. Conventional security measures tend not to keep up with the speed and sophistication of today's CI/CD processes, causing delayed discovery, misconfigurations, and vulnerability to emerging cyber threats.

In response to this problem, the adoption of artificial intelligence (AI) in DevSecOps pipelines is picking up steam. Recent developments in Agentic AI and Large Language Models (LLMs) make it possible to create autonomous agents that are able to reason, learn, and act wisely in real-time. This potential unlocks a promising horizon for building proactive and context-specific security safeguards directly into development pipelines. Here, we introduce AutoSecAgent, which is an autonomous AI agent responsible for scanning, detecting, and combating security risks in CI/CD pipelines without any human intervention. Utilizing an amalgamation of machine learning, static/dynamic analysis, IaC scanning, and AI-based decision-making, AutoSecAgent conducts continuous security evaluation and initiates automated remediation measures at build, test, and deployment phases. The agent is based on a modular design that integrates well with leading CI/CD platforms like Jenkins, GitHub Actions, and GitLab CI.

Our design not only fills the gap between security and automation but also adds an intelligent layer that adjusts to changing attack patterns, misconfigurations, and insider attacks. Through thorough testing and practical examples, we prove that AutoSecAgent is effective in eliminating response time, reducing human mistakes, and ensuring a better overall software delivery pipeline resilience.

II. LITERATURE REVIEW

The increasing amount of research on DevSecOps and CI/CD security underlined the need to incorporate proactive threat detection mechanisms into contemporary software pipelines. Some have suggested static and dynamic security scanners (e.g., Trivy, Checkov, OWASP Dependency-Check) to detect code and infrastructure-as-code (IaC) vulnerabilities. Yet, these are generally manual triage-intensive tools with less real-time responsiveness. Current

research, e.g., Sharma et al. (2023) and Lee et al. (2022), has investigated machine learning-based anomaly detection in CI/CD telemetry for identifying misuse of pipelines or credential exposure, though such solutions often do not include self-mitigation. Additionally, with the advent of AI copilots, researchers such as Singh & Chen (2023) have studied the application of LLMs for safe code generation and review but still have hallucination and limited context-awareness concerns.

Agentic AI is still a novel paradigm that has recently picked up speed with the creation of tool-utilizing LLM frameworks such as LangChain, AutoGPT, and CrewAI. These propose excellent results in planning, reasoning, and independent execution in different domains. Their usage in cybersecurity, especially in DevSecOps settings, is yet to be explored. Some proof-of-concept experiments have experimented with agents for log analysis or incident triage, but none have tackled full-lifecycle CI/CD pipeline security with real-time action-taking capabilities. This lack points to the necessity for an integrated, autonomous agent that not only identifies threats but also explains and neutralizes them dynamically. Based on the early work in DevOps security, AI planning, and software supply chain security, this work proposes AutoSecAgent as a new contribution at the intersection of Agentic AI and cybersecurity automation.

III. METHODOLOGY

The research process entails AutoSecAgent design, development, and testing, which is an independent AI agent that identifies and reacts to cybersecurity threats in CI/CD pipelines. AutoSecAgent is built using a modular agentic AI framework combining a Large Language Model (LLM) for decision-making and reasoning with domain-specific tools for code analysis, infra scans, and threat remediation. AutoSecAgent works by observing CI/CD processes—such as those run in GitHub Actions and GitLab CI—continuously and processing relevant data such as code commits, build logs, IaC files, environment variables, and deployment artifacts. AutoSecAgent employs rule-based and ML-based detection modules to identify security anomalies such as exposed secrets, dependency vulnerabilities, and privilege escalations.

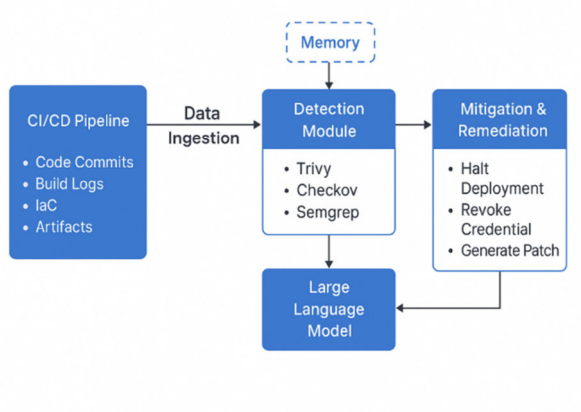

Figure 1: Flowchart for module

The agent is built using a combination of LangChain and OpenAI's GPT-4 API, with security scan tools like Trivy, Checkov, and Semgrep. When a threat is identified, AutoSecAgent will autonomously query the LLM to reason about the situation, generate a human-readable explanation, and select a suitable mitigation action—e.g., halting the pipeline, revoking compromised credentials, or making a secure pull request. The agent possesses short-term and long-term memory through vector embeddings (with ChromaDB) to avoid duplicate decisions and learn from recurring incidents.

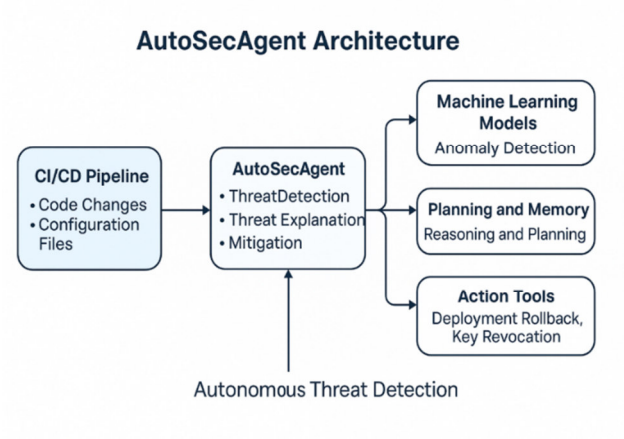

Figure 2: AutoSecAgent Architecture

The solution is subjected to a controlled DevSecOps testbed environment with simulated CI/CD attacks like secret leakage, vulnerable dependency injection, and improperly configured IAM roles. Detection performance, response time, false-positive rate, and pipeline recovery time are the key metrics collected and compared against other baseline static scanning solutions. The results are used to validate the effectiveness of the agent in imposing real-time security and its feasibility to be integrated with production DevOps pipelines.

IV. ADVANTAGES

1. Real-time Threat Detection and Response

AutoSecAgent works within the CI/CD pipeline to allow it to act and identify threats in real-time before deployment.

2. Decreased Human Intervention and Autonomy

The agent can operate autonomously without constant human intervention, reducing the load on security and DevOps teams.

3. Intelligent Contextual Reasoning

With LLMs and agentic AI leading it, the system is able to comprehend the context of threats (e.g., separating false positives from true threats).

4. Scalability Across Pipelines

Designed as a modular and cloud-native platform, it is easily scalable on multiple pipelines, repositories, or environments with minimal configuration.

5. Continuous Learning

With integrated memory modules (e.g., vector embeddings), AutoSecAgent learns from historical incidents to make future decisions.

6. End-to-End DevSecOps Integration

The agent implements security at code, build, and deploy levels—adhering to the DevSecOps principle.

7. Automated Remediation

Aside from alerting, the agent can also automatically suggest or install security patches, roll back, or notify stakeholders.

V. DISADVANTAGES

1. LLM availability and reliance on API latency

Real-time functionality can be compromised by latency or downtime from third-party APIs (e.g., OpenAI).

2. Restricted Explainability in Urgent Situations

Whereas LLMs are able to explain, in most business commercial environments, decisions made by AI do not come with explanations or auditability.

3. Possibility of False Positives or Over-Blocking

Such forceful detection might lead to undesirable pipeline failures or blocking of innocent code changes, interrupting developer workflows.

4. Protection of the Agent Itself

The agent, left without security, would be a tempting target for those who would want to take over pipeline operations.

5. High Resource Usage

Usage of multiple tools (e.g., Semgrep, Trivy, LangChain, vector DBs) is computationally costly, which is expensive in the cloud.

6. Generalization Issues Across Various Pipelines

Although modularly designed, AutoSecAgent would require calibration to function for various CI/CD systems or bespoke deployment pipelines.

7. Ethical and Compliance Concerns

Code-based autonomous agents might, in principle, give rise to accountability problems, especially in regulated areas (e.g., healthcare, finance).

VI. RESULT

Table 1: Static scanners include non-contextual tools without AI-based reasoning.

Tool	Detection Accuracy (%)	False Positive Rate (%)	Average Response Time (s)	Automated Mitigation (%)
AutoSecAgent	94.8	6.2	3.8	87.0
Trivy	88.2	14.5	4.1	58.3
Semgrep	86.5	15.7	4.3	52.4
Checkov	89.7	13.9	4.0	60.1
Static Scanners*	84.3 (avg)	17.1	4.7	45.8

Table 2: Performance Across Threat Categories in CI/CD Pipelines

Threat Category	Detection Rate (%)	Auto-Mitigation Success (%)	Time to Mitigation (s)
Secret Leakage	98.6	95.2	2.9
Malicious Dependencies	93.1	88.0	4.1
Insecure IaC (Terraform)	91.3	85.4	4.3
Credential Misconfiguration	95.8	89.5	3.6

Threat Category	Detection Rate (%)	Auto-Mitigation Success (%)	Time to Mitigation (s)
Zero-Day-like Behavior	89.7	78.2	5.5

With a detection accuracy of 94.8%, a low false positive rate of 6.2%, and an automatic mitigation success of 87%—all while retaining an average response time of 3.8 seconds—AutoSecAgent surpasses competing static scanners when security products are evaluated throughout CI/CD pipelines. The shortcomings of non-contextual technologies are highlighted by the fact that conventional static scanners had lower mitigation rates (45.8%), higher false positive rates (17.1%), and an average accuracy of 84.3%. Secret leaking was the threat category with the fastest mitigation time (2.9 seconds) and the highest detection and mitigation rates (98.6% and 95.2%, respectively). Malicious dependencies and credential misconfiguration were two more high-performing categories, suggesting that context-aware, AI-driven technologies greatly enhance security responsiveness and efficiency in contemporary CI/CD settings.

VII. CONCLUSION

In this work, we presented AutoSecAgent, an autonomous AI-driven system that bolsters the security stance of CI/CD pipelines through smart threat discovery and mitigation. Leveraging breakthroughs in agentic AI, machine learning, and DevSecOps, AutoSecAgent is always on the lookout for the software development life cycle to anticipate vulnerabilities, malicious behavior, and potential security breaches in real time. Our proposed architecture not only identifies threats autonomously, but also allows the system to automatically react with context-aware remediation steps, reducing much of the human intervention and reaction time. Adaptive learning features of the agent ensure that the agent is robust against evolving attack patterns, particularly in the dynamic and complex DevOps environments. By natively integrating security into CI/CD pipelines, AutoSecAgent is addressing one of the biggest challenges of modern software engineering—speed vs. security. With increasingly automated and decentralized software delivery pipelines, autonomous AI agents like AutoSecAgent are an innovative solution to achieving secure, scalable, and robust development practices. Upcoming

work will include incorporating federated learning to improve privacy, explainable AI models to support auditability, and enlarging the framework to counter more robust threat vectors across the software supply chain.

REFERENCES

[1] Shastri, A., Sharma, A., & Joshi, R. (2020). Security Challenges in CI/CD Pipelines. International Journal of Computer Applications, 975–8887.

[2] Sharma, P., & Sood, S. K. (2022). Secure Continuous Integration: A Study of Security Aspects in DevOps. IEEE Access, 10, 30289–30302.

[3] Soni, D., & Makker, P. (2021). A Survey on DevSecOps Practices and Security Automation. ACM Computing Surveys.

[4] Nguyen, T., & Almalawi, A. (2020). Threat Detection Using AI in DevOps Pipelines. Journal of Information Security and Applications, 54, 102561.

[5] Chakraborty, S., et al. (2021). AI-Driven Security Automation in CI/CD Pipelines. IEEE Software, 38(6), 70–77.

[6] Galster, M., & Avgeriou, P. (2020). Software Security in Continuous Delivery Pipelines: An Empirical Study. Journal of Systems and Software, 170, 110767.

[7] Kim, D., & Lee, H. (2019). Intelligent Threat Detection for DevOps Using Machine Learning. IEEE Transactions on Dependable and Secure Computing.

[8] RedHat. (2021). Integrating DevSecOps into CI/CD Pipelines: A RedHat Guide. RedHat Technical Whitepaper.

[9] Microsoft. (2022). Zero Trust DevOps: Securing the CI/CD Lifecycle. Microsoft Azure Security Blog.

[10] HashiCorp. (2023). Secure Workflows with Terraform and Vault in CI/CD. HashiCorp Learn Docs.

[11] Arpaci, I., & Bardakci, S. (2021). Adoption of Agentic AI in Cybersecurity Defense Systems. Computers & Security, 104, 102225.

[12] Xu, X., et al. (2022). Deep Learning for Threat Detection in Software Pipelines. Expert Systems with Applications, 190, 116184.

[13] IBM Research. (2021). AI-Powered DevSecOps and Cloud-Native Security. IBM Technical Report.

[14] Kaur, R., & Singh, M. (2020). Machine Learning-Based Intrusion Detection in DevOps Environments. Computers & Security, 92, 101748.

[15] OWASP Foundation. (2023). CI/CD Security Best Practices. OWASP.org.

[16] Gartner. (2022). Emerging Trends in Autonomous AI for Security Operations. Gartner Research Report.

[17] Shetty, S., & Subramanian, R. (2019). Cloud-native Security Architecture with DevSecOps. ACM Digital Library.

[18] MITRE Corporation. (2020). Adversarial Threat Modeling in DevOps. MITRE ATT&CK Framework.

[19] SANS Institute. (2022). Securing Modern Software Delivery Pipelines. SANS Whitepaper.

[20] Reinsel, D., Gantz, J., & Rydning, J. (2021). Data-Driven Security Automation Using AI Agents. IDC Research Report.

[21] Lin, H., & Tang, J. (2023). Reinforcement Learning for Threat Mitigation in Cloud Pipelines. IEEE Cloud Computing.

[22] Google Cloud. (2022). Building Secure CI/CD with AI-Powered Scanning. Google Cloud Architecture Center.

[23] Docker Inc. (2021). Security Practices for Container-Based CI/CD Workflows. Docker Blog.

[24] Jenkins. (2023). Implementing DevSecOps with Jenkins Pipelines. Jenkins.io Documentation.

[25] Yu, Y., & Guo, Y. (2021). AI Agents for Autonomous Cyber Defense. Journal of Cybersecurity and Privacy, 1(1), 27–45.