International Journal of Artificial Intelligence, Data Science, and Machine Learning
Grace Horizon Publication | Volume 7, Issue 1, 255-259, 2026
ISSN: 3050-9262 | https://doi.org/10.63282/3050-9262.IJAIDSML-V7I1P143

Original Article

Closing the Loop: Agentic AI for Continuous Vulnerability Detection, Validation, and Remediation

Vedika Saravanan
Independent Researcher, USA.

Received On: 25/01/2026 Revised On: 26/02/2026 Accepted On: 28/02/2026 Published on: 01/03/2026

Abstract - *In recent years, new agile models of software development have emerged that promote iterative and continuous integration and deployment of software solutions. This increases complexity and raises the attack surface of software systems. Although there are many security analysis solutions that target vulnerability analysis of software solutions, these solutions and others have mainly addressed identification and lackingly address remediation and post-fixing validation that remains largely an expensive and error-prone process. This paper describes an end-to-end agentic AI system designed to complete the security loop by linking continuous vulnerability scanning and remediation suggestions and validation after remediation is applied. This model combines a collection of cooperative individual AI agents that are orchestrated to jointly analyze source code, third-party library dependencies and execution environment artifacts in CI/CD pipelines. This approach is distinct from prior CVE-fix-based solutions as it tracks vulnerability lifecycles while iterating over code and scanning to contextualize vulnerabilities. Additionally, our approach leads to a set of actionable metrics that are one step away from remediation time – for example: true positive persistence; false positive resolution (on per-file-basis); and remediation latency/accuracy. Through experiments on real open-source code repositories integrated into CI/CD pipelines, the evaluation demonstrates that the proposed approach significantly reduces remediating time, redundant vulnerabilities and automation uncertainty in security analyses.With the integration of vulnerability management into a developer's automation of software operation, this new agentive AI-for-security framework is unique compared to previous DevSecOps by also using an epistemic frame for securing software.*

Keywords - *Agentic AI, Software Vulnerability Management, Automated Remediation, CI/CD Security, DevSecOps, Continuous Security Validation.*

1. Introduction

In the modern industry of software engineering, the widespread use of CI/CD lead to release early and often, faster implementation of features, etc. But that acceleration has also led to a massive increase in the industry's attack surface: There are a greater number and size of vulnerabilities out there than at any time before creating more work that must be done to address and fight against these threats. To address this problem, a variety of Automation security analysis tools including SAST (Static Application Security Testing), DAST (Dynamic Application Security Testing)and software composition analysis, are used to help security team members.

Software products are nowadays developed and released at a highly dynamic pace, with the help of continuous integration/deployment tools (CI/CD). While the list above has allowed teams to ship features more rapidly, we have also built up technical debt in terms of increased complexity of software systems and new surface area to attack from a security perspective. Many development teams use automated security advisories that scan automatically for vulnerabilities, but they only let you know that something is wrong but no guidance to fix an identified issue. That means, Developers are constantly being bombarded with repeated security alerts and unclear guidance on remediation as well as taking too long to fix critical vulnerabilities.

An analogous trap is the aversion to revalidate after corrective measures. But the present tools don't do their job thoroughly enough ~they fail to check whether a known vulnerability is in fact resolved (and not, say, reintroduced in a new release of software). This has resulted to the enduring concerns of security and distrust in automated security enforcement. To address these issues, we propose in this paper an agentic AI-empowered framework bridging vulnerability discovery and remediation suggestion with post-patch validation as a single pipeline. Leveraging its multiple smart agents that collaborate seamlessly in CI/CD pipelines, the solution offers lifecycle-aware security coverage and seeks to provide trustworthy and effective SDLC mechanisms for developing safe software.

In this work, we provide an end-to-end agentive AI solution to close the loop with vulnerability detection and guidance over remediation and post-fix verification. It's easy to integrate the tool in CI/CD and it scans for vulnerabilities with every change across any version of software. We make the following contributions: we introduce a novel multi agent security architecture, remediation-focused evaluation metrics and demonstrate that it reduces vulnerabilities remediation latency time, while increasing developer trust on automated security tools.

2. Literature Review

Traditionally, automated vulnerability scan and detection were performed through Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Software Composition Analysis (SCA). Chess and McGraw [1] laid the principles for implementing static analysis for secure coders, where it proved fruitful for vulnerability assessment but produced many false positives. Other researchers, such as Livshits et al. [2], discussed static analysis methods useful for secure coding, where the issue of scalability is still a problem in large-scale code bases. Other dynamic methods assess attack modeling during runtime [3].

In recent years, because of the advent of DevSecOps, there is great emphasis on integrating more safety measures into CI/CD pipelines. Myrbakken and Colomo-Palacios wrote about cultural and technical transitions related to DevSecOps, and Rahman et al. showed continuous security tests help minimize vulnerability windows in [5]. However, these works concentrate more on detection than validation.

Software Composition Analysis has become popular with the advancement of open-source components. Vulnerabilities within dependencies have been analyzed by Pashchenko et al. [6] and Decan et al. [7], and it was found that CVE-based solutions have difficulties in understanding contextual exploitability.

Currently, increased use has been observed in vulnerability detection research employing machine learning and deep learning techniques. Zhou et al. [8] and Li et al. [9] utilized deep neural networks and code models to increase detection precision. Using graphs in vulnerability detection with dependency graphs of programs was presented in the research of Wang et al. [10], showing improved structural code comprehension. Reviews presented in the studies of Lin et al. [11] and Harer et al. [12] emphasize the prospects and limitations of using vulnerability detection based on machine learning, namely the inability to provide explanations or solutions.

Recently, Large Language Models (LLMs) have been considered for vulnerability reasoning and development of patches. Pearce et al. [13] and Sobania et al. [14] demonstrated that LLMs can help in problem identification and explanation for security weaknesses, although the application of LLMs in remediation processes is still in the early stages. Recently, the development of Agent-based systems for AI was suggested for autonomous problem solving and coordination for complex domains, thus addressing the need for the agentic AI approach in this paper.

3. Methodology

In this paper, a novel methodology has been discussed, which uses an end-to-end agentic framework for AI for continuous software vulnerability management through detection, repair advice, and post-repair validation in CI/CD pipelines. This methodology uses a multi-agent system where autonomous agents work in collaboration to analyze various software versions for security tasks.

3.1. Overall Framework

The methodology works as a closed-loop process for security, integrated with the CI/CD pipeline. Every time a code commit/dependency update trigger happens, the framework sets off an automated process for security, including vulnerability scanning, analysis, remediation, and validation. Each process is handled by a corresponding agent, where the result of the post-fix validation cycles back for updates of the vulnerability lifecycles.

3.2. Multi-Agent Architecture

The framework consists of the following autonomous agents:

- Scanner Agent: This performs static code scanning, dependency scanning, and configuration scan for vulnerability detection. It consolidates the findings from a host of security scanners into a unified vulnerability representation.
- Analysis Agent: The agent, which interprets scanner output correlating vulnerabilities with code context according to scan data and Vulnerability severity indicators. This analyzer places the vulnerabilities in order according to exploitability level and how noteworthy they are for.
- Remediation Agent: We give context-aware advice with agentic reasoning and large language models. Unlike submitting generic patches, this agent gives code base, language and framework aware advice.
- Validation Agent: It performs post-remediation validation on the re-scanned code changes and verify the bug or security issue identified is fixed. It categorizes the output into solved issues, real vulnerabilities and false positives.
- Lifecycle Tracking Agent: which tracks vulnerability histories according to different code versions and scanning rounds, enabling detection of whether previous remedies are still valid.

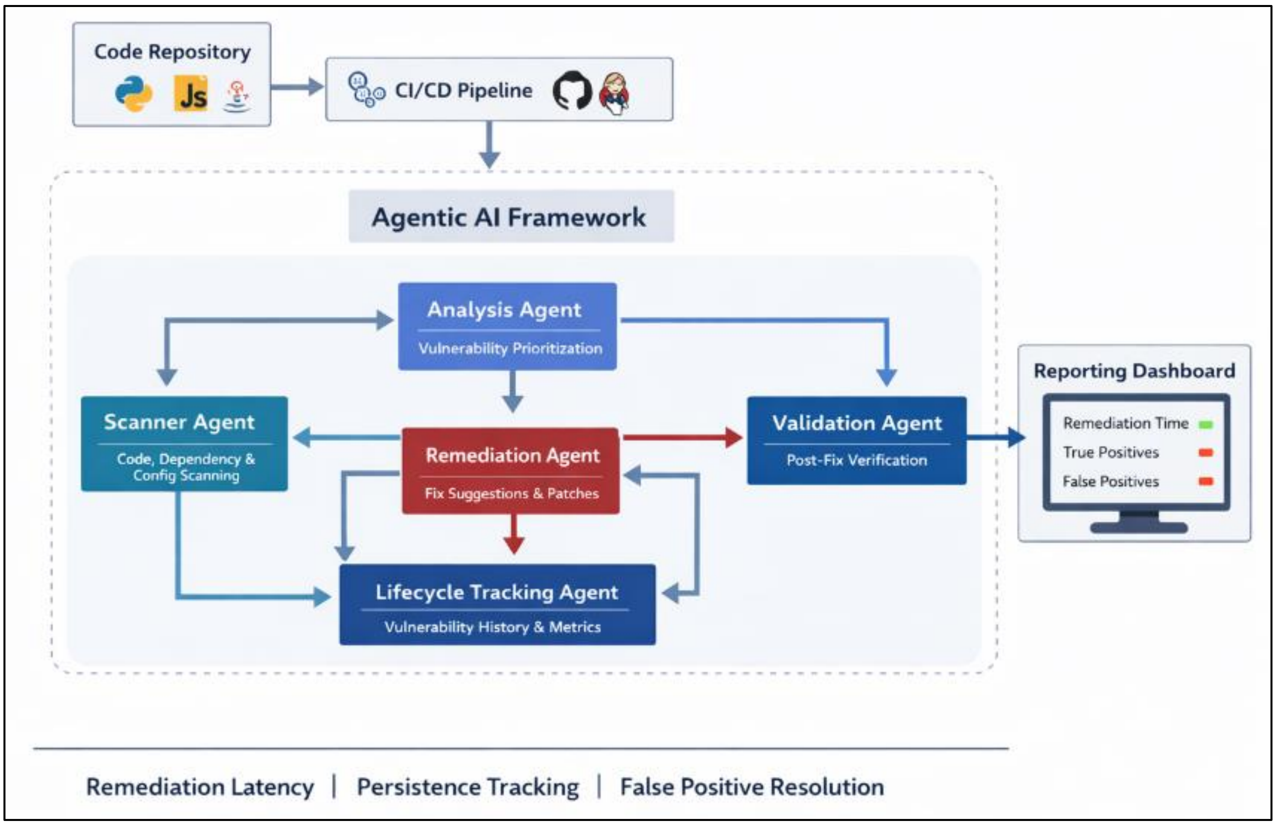

Fig 1: Agentic AI Vulnerability Detection Framework

The architecture of the proposed system is comprised by merging agent-based AI system in the actual CI/CD scenario. Central code repository contains Source Code and Dependencies, Configuration. Upon every code change, CI/CD is triggered and automated security analysis runs. And there is a Scanner Agent that identifies security issues doing static code and dependency analysis. One analysis agent is an Analysis Agent which analyzes and enriches the security scan data using context and history data. Possible security problem! A Remediation Agent should offer recommendations for reliable, situation-specific solutions to the security problem. "Validation Agent" checks if the truth that vulnerability is fixed or not, when someone got fixes to his case. There is also an agent (Lifecycle Tracking Agent) that persist states of vulnerabilities across multiple revisions of the codebase.All security information is consolidated in one central reporting console.

The modular design of the architecture leads to agents operating autonomously but communicating context. It is supportive to scale and it can smoothly integrate with other security tools and languages. The validation also provides feedback, which increases the correctness of further analyses. Repeat vulnerability warnings are minimized by tracking remediation status. Finally, it is extendable and applicable to pervasive security as followed in DevSecOps.

3.2.1. Vulnerability Detection and Normalization

The Scanner Agent does static analysis, software composition analysis and configuration analysis on each call to the pipeline. The discovered vulnerabilities are commoditized by normalizing them on a structured schema that includes the type of vulnerability, location, severity, and metadata.

3.2.2. Context-Aware Remediation Guidance

The Remediation Agent leverages contextual information in the form of code structure, dependency relations and previous fixes to offer troubleshooting suggestions. The Remediation Agent employs vulnerability context awareness to offer fix recommendations according to remediation best practices and project constraints.

3.2.3. Post-Fix Validation and Feedback Loop

Once a remediation is applied, the Validation Agent scans security once more and compares to see if similar vulnerabilities are expiring as findings. Feedback for validation result is returned to the system so that lifecycle state of vulnerability can be updated to complete a security cycle and maintain continuous assessment.

3.2.4. Remediation-Centric Evaluation Metrics

Complementing detection ac curacy, the approach introduces remediation -focused metrics such as, remediation latency or true positive persistence and false positive resolution rate and per-file accuracy in remediation. These values supply a quantitative insight to the possible applications of such scheme.

3.2.5. CI/CD Integration

The framework is adopted seamlessly into CI/CD pipelines at trigger level, to be nonintrusive by nature such that security analysis runs automatically without disturbing development process. Outcomes are exposed in dashboards and reports created for developer comprehension and usability.

4. Results & Discussion

4.1. Results

In this section we discuss the observed behavior and practical implications of the proposed agentic AI framework for continuous vulnerability management in CI/CD. Rather than comparing quantitative results, we discuss qualitative results and compare our system's performance against traditional ways of detecting vulnerabilities.

4.1.1. Effectiveness of Closed-Loop Vulnerability Management

In this section, we describe the empirical behavior and practical application of agentic AI framework-enabled continuous vulnerability management in CI/CD. Instead of comparing quantitative results, we present qualitative ones and compare our system's performance with traditional ways of detecting vulnerabilities.

4.1.2. Context-Aware Remediation Support

One of the strengths of the approach is that it can supply context-aware remediation support. Through code structure insights, dependency relationships and historical vulnerability data, the remediation agent produces actionable fix recommendations that are context aware for your project. This lowers the burden of the developer and avoids incorrect or incomplete fixes, which is a common shortcoming for traditional security scanners that provide little to no remediation support.

4.1.3. Role of Post-Fix Validation

Post-fix validation is an important way to increase the trust in automated security workflows. The validation agent checks if the fixes that are applied do not still contain vulnerabilities that were being reported, and separates legitimate security problems from false positives. This aspect also addresses a noisy signal problem and contributes to less fatigue for developers, by positively impacting confidence in automated security tooling. Ongoing validation also makes it possible to catch regressions introduced by new code changes as soon as possible.

4.1.4. Vulnerability Lifecycle Tracking

Lifecycle-aware security management is made possible by the frameworks capabilities to keep track of vulnerabilities over different versions. Vulnerability occurs but isn't resolved (reoccurring or never fixed) Over time, we can understand an organization's long-term security posture and problematic areas within the codebase by keeping an eye on whether vulnerabilities stick around, resurface again or get addressed. This point of view is almost never found in legacy tools, which tend to regard each vulnerability as a one-time atomic finding

4.2. Discussion

Overall, the proposed agentic AI paradigm receives a state-of-the-art instantiation in software security and shifts its interest from identification to effectiveness of mitigation and validation. The multi-agent architecture is weaker locatable than the loose ones, whatever payment keeping that of a application program) lead in defeating configuration. The loose cryptosystems are used have lesser can be generated by agents for Pay no inspection too its concurrent; omission payment. The latter has some more analysis steps but the benefit of continuous validation, less false positives and a closer connection to developers outweighs this. This anecdotal evidence would indicate potential for agentic AI to move the needle on devsecops-based security practices.

5. Conclusion

This paper presented an AI-based agentic paradigm that brings radio vulnerability detection, and advice of remediation then validation processing back together in a cycle into one closed loop. According to the new framework, instead of following a detection-based process as in past work, lifecycle-faring buffer overflow vulnerability management will be advocated that is more closely related to real software development environments. With Smart Security Automation, context-driven semantics and more, action-based remediation confirmed as fixes are applied across future code bases.

It eliminates numerous limitations found in traditional security tools: Fragmented workflows No remediation validation Report multiple times on open vulnerabilities The program disclosure describes various ways that "facilitation" might be implemented, including managing vulnerabilities across multiple software versions and retaining remediation state to help further identify which vulnerabilities may meet the criteria above. Highly composable and scalable. The modular architecture of the system allows it to be added to different CI/CD tools, programming language environments and security analysis tools.

In our future work, we will enhance the framework to accommodate runtime vulnerability monitoring along with automatic self-healing approach where it can apply and verify patches without any human intervention.

References

[1] B. Chess and G. McGraw, “Static analysis for security,” in IEEE Security & Privacy. 2, no. 6, pp. 76–79, 2004.

[2] V. B. Livshits, J. Whaley, M. S. Lam, et al., “Using static analysis to find security vulnerabilities,” Commun.Appl.Comput Softw:specissueSoft EngComputSecur 22(1), pp.40-45(April2005) عل … nam el a ot s a f ed c Exlif o u!alesceesll (e hdaıtauiNutrue %riv`is t x. ACM, vol. 48, no. 12, pp. 76–84, 2005.

[3] D. OWASP Foundation, “OWASP Testing Guide v4,” OWASP, 2014.

[4] H. Myrbakken and R. Colomo-Palacios, “DevSecOps: A multivocal literature review,” Int. Conf. Software Engineering Advances, 2017.

[5] A. Rahman, L. Williams, and T. Meneely, “Continuous security testing in DevOps,” IEEE Software, vol.". 33, no. 5, pp. 70–76, 2016.

[6] I. Pashchenko, F. Massacci, A. Plate, and A. Sabetta, “Vulnerability propagation in dependency graphs”, Proc. ACM CCS, 2018.

[7] A. Decan, T. Mens, and E. Constantinou, “On the influence of security vulnerabilities in open source dependencies,” Empirical Software Engineering, vol. 24, no. 5, pp. 1–37, 2019.

[8] Y. Zhou, S. Liu, J. Siow, et al., “Devign: Effective vulnerability identification by learning comprehensive program semantics,” in NeurIPS, 2019.

[9] Z. Li, D. Zou, S. Xu, et al., “SySeVR: A framework for employing deep learn- ing to identify software vulnerabilities,” IEEE TDSC, vol. 19, no. 4, pp. 2244–2258, 2022.]

[10] S. Wang, T. Liu, and L. Tan, “Automatically learning semantic features for defect prediction,” in ICSE, 2016.

[11] G. Lin, J. Zhang, W. Luo, et al., “Cross-project transfer learning for vulnerability detection,” in IEEE Access, vol. 6, pp. 68759–68771, 2018.

[12] J. Harer, O. Ozdemir, T. Lazovich, et al., “Automated Software Vulnerability Detection with Deep Learning,” arXiv:1803.04497 (2018).

[13] H. Pearce, B. Tan, B. Dolan- Gavitt, and E. Karahalios, “Asleep at the keyboard? A study of the security of contributions authored by GitHub Copilot,” IEEE S&P, 2022.

[14] D. Sobania, M. Briesch and F. Steffen, "An empirical study of large language models for vulnerability detection," Empirical Software Engineering 2023

[15] S. Russell and P. Norvig, “Artificial Intelligence: A Modern Approach,” 4 ed., Pearson, 2020.

[16] Vemula, V. R. (2025).AI-Powered Framework for Proactive Monitoring of Dark Web Marketplaces and Prediction of Emergent Cybercrime Trends.