Code Vulnerability/Security Agent

Tony Chan 3039174673, Arhaan Aggarwal 3036541288, George Ingebretsen 3037937450

Abstract

In an era where software security is of the utmost importance, traditional static code analyzers often miss subtle, context-dependent vulnerabilities. Prior work in this area has mainly relied on rule-based systems that effectively catch known issues but fail to find complex, nuanced security flaws. To overcome these limitations, we propose an LLM-Powered Code Vulnerability Detector that integrates conventional static analysis with semantic reasoning capabilities of LLMs. This hybrid approach not only scans source code for risky patterns but also provides natural language explanations and actionable code fix suggestions, and can be seamlessly integrated into CI/CD pipelines for real-time feedback. The expected output is a tool that enhances vulnerability detection, ultimately aiding developers in maintaining high code quality and bolstering software security.

Research Goals and Problem Statement

Traditional static analysis tools, while effective at identifying known vulnerability patterns through rule-based systems, struggle with many issues. They often generate numerous false positives and negatives due to their inability to fully comprehend the semantic context of code, leading to missed vulnerabilities or unnecessary alerts. Moreover, these systems usually offer limited explanations and lack actionable, context-aware recommendations, making it challenging for developers to fix issues efficiently. Additionally, their integration into modern CI/CD pipelines is often quite basic, limiting real-time feedback and continuous improvement.
To tackle these limitations, our project aims to integrate LLMs with conventional static code analysis. By leveraging the LLM's natural language understanding and reasoning capabilities, we intend to enhance vulnerability detection by capturing subtle, context-dependent security flaws that rule-based methods often overlook. The proposed solution will provide detailed, natural language explanations for identified vulnerabilities and generate precise code fix suggestions, thereby improving developer comprehension and response time. Furthermore, by embedding this hybrid system into CI/CD workflows, we aim to facilitate seamless, real-time code reviews and continuous security monitoring, ultimately enhancing software security and reducing the incidence of overlooked vulnerabilities.

Description of the Proposed Work and Expected Outcomes

Our proposed solution involves a hybrid system that integrates traditional static code analysis with LLMs to address the limitations of current tools. First, existing static analyzers will identify code segments that potentially contain vulnerabilities based on rule-based patterns. These segments will then be passed to an LLM, which has been fine-tuned on security best practices and coding patterns, to provide deeper semantic analysis. The LLM will generate natural language explanations for why each identified issue might be a vulnerability, offer precise code

fix suggestions, and even highlight potential edge cases that rule-based methods might miss. The system will be designed to integrate seamlessly into CI/CD pipelines, providing developers with real-time feedback during the code review and merge process.
To validate our approach, we plan to perform both quantitative and qualitative evaluations. Quantitatively, we will test the system on established open-source repositories with documented vulnerabilities, comparing our tool's precision, recall, and F1 scores against traditional static analyzers. Qualitatively, we will conduct user studies with developers to assess the clarity, usefulness, and actionable quality of the generated recommendations and explanations. We will also simulate CI/CD environments to ensure that our integration does not introduce performance bottlenecks and effectively improves the overall development workflow.
Expected outcomes include enhanced detection of subtle, context-dependent vulnerabilities, a reduction in false positives and negatives, and improved remediation times thanks to the detailed, actionable insights provided by the LLM. Ultimately, our system aims to improve software security and developer productivity by bridging the gap between traditional static analysis and advanced semantic reasoning.

Prior Work (How the Research Relates to Prior Work)

“A Survey on Automated Software Vulnerability Detection Using Machine Learning and Deep Learning” identifies 5 challenges / open problems in the field of automated vulnerability detection: (1) semantic representation, (2) prediction granularity, (3) false positive removal, (4) lack of training data, and (5) lack of model interpretability. Areas 1-4 are particularly well addressed by using frontier LLM agents to automate vulnerability detection. As shown in “Towards Explainable Vulnerability Detection with Large Language Models”, fine-tuning LLMs on annotated security code snippets can vastly increase vulnerability detection performance, even with a LoRa approximation of an only 13B param model.

Together, these indicate that LLM agents may be particularly well suited as a solution to automated code vulnerability detection, and a promising route to address the field's current shortcomings and needs.

References

Harzevili, Nima Shiri, et al. “A Survey on Automated Software Vulnerability Detection Using Machine Learning and Deep Learning.” *arXiv*, 2023, arXiv:2306.11673. https://arxiv.org/abs/2306.11673.

Mao, Qiheng, et al. “Towards Explainable Vulnerability Detection with Large Language Models.” *arXiv*, 2025, arXiv:2406.09701. https://arxiv.org/abs/2406.09701.

Sultana, Shaznin, et al. “Code Vulnerability Detection: A Comparative Analysis of Emerging Large Language Models.” *arXiv*, 2024, arXiv:2409.10490. https://arxiv.org/abs/2409.10490.