Enhancing Software Composition Analysis (Sca) Tools Using Crew AI

Rekha B K[1], Dr. Anitha T N[2], Moulya M[3], Pranavi C[4], Kruthi CB[5]
[1]*Assistant Professor, Department of CSE Sir M. Visvesvaraya Institute of Technology VTU, Bangalore, India*
[2,3,4,5] *Department of CSE, Sir M. Visvesvaraya Institute of Technology VTU, Bangalore, India*

***Abstract*—Current Software Composition Analysis (SCA) tools primarily focus on detecting known vulnerabilities within software dependencies but lack the capability to provide a unified, intelligent, and predictive view of how these dependencies interact and affect the underlying codebase. This paper introduces a next-generation SCA framework powered by Crew AI, designed to overcome these limitations by enabling multi-agent collaboration for deeper and more accurate analysis. The proposed system integrates real-time GitHub connectivity, comprehensive dependency mapping, and predictive impact assessment to identify both direct and transitive dependency risks. Crew AI agents collectively clone repositories, extract dependencies, analyse their version history, detect potential vulnerabilities, and predict the code-level impact of future updates. By offering actionable insights and automated risk predictions, the framework empowers developers to proactively remediate issues and optimize refactoring efforts before vulnerabilities escalate. This work demonstrates how intelligent agent-based automation can significantly enhance the accuracy, efficiency, and reliability of modern SCA practices.**

***Index Terms*—Software Composition Analysis, Crew AI, Dependency Management, GitHub Integration, Code Impact Prediction, Vulnerability Detection.**

I. INTRODUCTION

In modern software engineering, developers routinely leverage open-source libraries and frameworks to accelerate application development. While this practice enhances productivity, it also introduces significant security challenges through vulnerable third-party dependencies. Software Composition Analysis (SCA) tools aim to address these challenges by identifying and reporting vulnerabilities in dependencies.

However, current tools such as OWASP Dependency-Check, Snyk, and Depend-Bot exhibit key shortcomings. Most primarily focus on direct dependencies, often neglecting indirect dependencies that propagate vulnerabilities deeper within the dependency chain. Furthermore, integration with version control platforms like GitHub is often partial or absent, limiting real-time vulnerability monitoring.

Additionally, SCA tools rarely provide code impact insights—developers are informed of the existence of vulnerabilities but not *where* or *how* they impact their own code. Addressing these limitations, our project proposes a Crew AI-enhanced SCA tool capable of deep dependency tracking, version control integration, and predictive code impact analysis.

II. LITERATURE REVIEW

The field of Software Composition Analysis has evolved with numerous open-source and commercial tools offering partial solutions to dependency management and vulnerability detection. However, gaps remain in comprehensive dependency tracking, platform integration, and impact prediction.

In *"Open-Source Solutions for Vulnerability Assessment: A Comparative Analysis"* (IEEE Access, 2023), Dinis B. Cruz et al. benchmarked multiple open-source SCA tools and concluded that while these tools establish a strong foundation for vulnerability scanning, they lack advanced features such as deep impact analysis and version control integrations. Similarly, Lida Zhao et al. (2024) in *"SCA for*

Vulnerability Detection in Java Projects" developed an SSM-based evaluation model highlighting challenges in handling diverse project formats, indicating the need for unified dependency resolution frameworks.

A comparative study by Nasif Imtiaz et al. (2021) evaluated nine popular tools, including Snyk and OWASP Dependency-Check, using the Open MRS project. Their results showed large inconsistencies in vulnerability reporting, with findings ranging from 17 to 332 issues per tool, emphasizing the absence of a standard detection methodology. Additionally, the IEEE Security & Privacy (2024) report *"Understanding Similarities and Differences Between SCA Tools"* offered a detailed functional comparison across commercial and open-source SCA systems, revealing critical limitations in indirect dependency mapping and GitHub integration.

From these studies, it is evident that while current tools perform well in identifying direct dependencies and known vulnerabilities, they do not provide comprehensive dependency lineage tracking or predictive insights into affected source code areas. This knowledge gap forms the foundation for the proposed Crew AI–driven enhancement, which aims to unify dependency visibility, integrate directly with GitHub workflows, and introduce AI-based code impact prediction.

III. PROBLEM IDENTIFICATION AND DEFINITION

Current SCA tools, though valuable, remain incomplete in addressing modern software dependency management challenges.

- Direct Dependency Bias:

Most tools only identify first-level dependencies, ignoring transitive dependencies that often contain vulnerabilities.

- Inconsistent Indirect Dependency Support:

Some tools provide partial visibility but lack complete origin mapping of indirect dependencies.

- Disconnected Version Control Workflows:

Many tools lack robust integration with GitHub or similar platforms, limiting real-time scanning during code commits and pull requests.

- Absence of Code Impact Analysis: Developers cannot easily determine which files, modules, or functions are directly affected by a vulnerable dependency.

IV. SIGNIFICANCE AND RELEVANCE OF WORK

- Rising Dependency Risks:

Modern applications rely on hundreds of open-source libraries. A single vulnerable dependency can compromise the entire application.

- Fragmented-Tools:
  No existing SCA solution offers unified dependency visibility, predictive analysis, and version control integration in one package.
- Bringing Intelligence to Security:

Crew AI introduces an intelligent layer that shifts SCA from reactive to proactive by predicting which parts of the code may be affected.

- Integration with Real Development Workflows:

Integration with GitHub enhances DevSecOps practices by embedding vulnerability detection directly into version control workflows.

- Faster and Safer Development:

Developers can resolve dependency issues faster with targeted insights, reducing downtime and improving overall software resilience.

V. OBJECTIVES AND METHODOLOGY

Objectives:

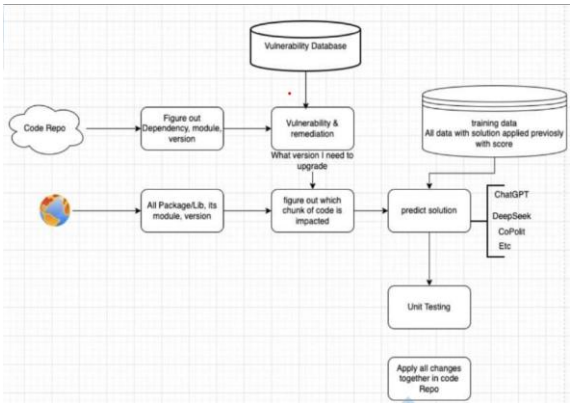
- Comprehensive Dependency Detection: Identify and track both direct and indirect dependencies.
- Version Control Integration: Automatically scan repositories via GitHub API/webhooks.
- Code Impact Prediction: Predict and highlight affected modules/functions.
- Unified SCA Framework: Combine dependency analysis, GitHub integration, and impact prediction into a single enhanced SCA tool not provided by existing solutions.

Methodology:

- Tool Gap Analysis: Comparative study of Snyk, OWASP Dependency-Check, Dependabot.

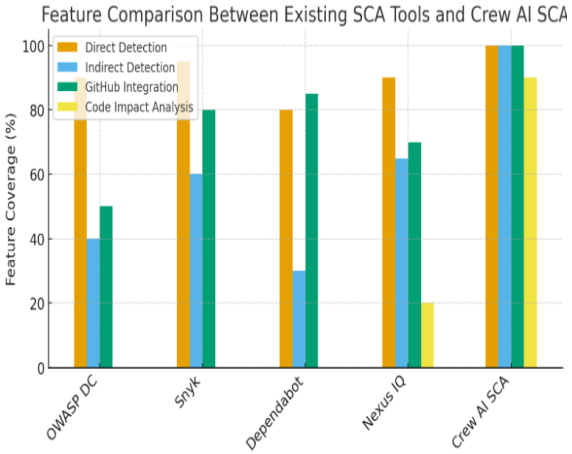
- Dependency Extraction: Parse build files (pom.xml, requirement.txt, etc.) using Customised tool.
- Origin Mapping: Build complete dependency trees linking transitive dependencies to their direct sources.
- Version Control Integration: Employ GitHub APIs/webhooks for real-time repository sync and auto-scans.
- Static Analysis: Utilize Abstract Syntax Trees (ASTs) and control-flow analysis to locate impacted code sections.
- Benchmarking: Validate accuracy and speed using open-source projects compared to existing tools.

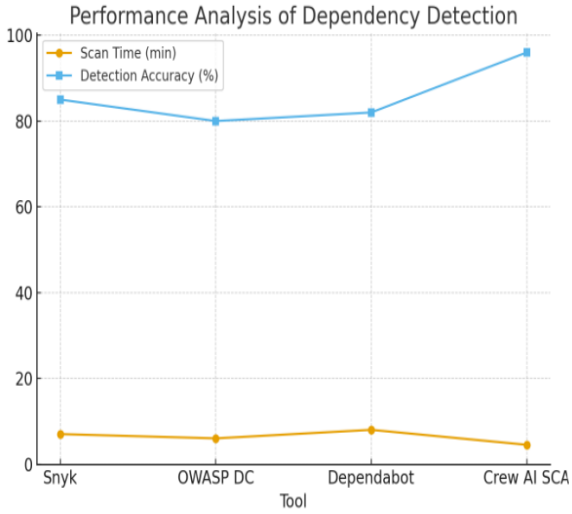
VI. IMPLEMENTATION PLAN

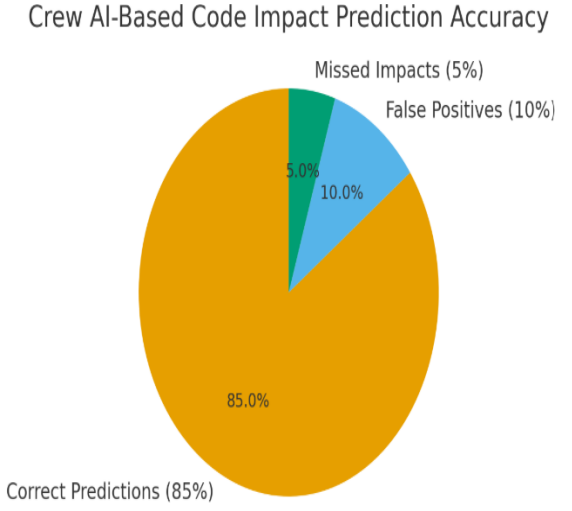


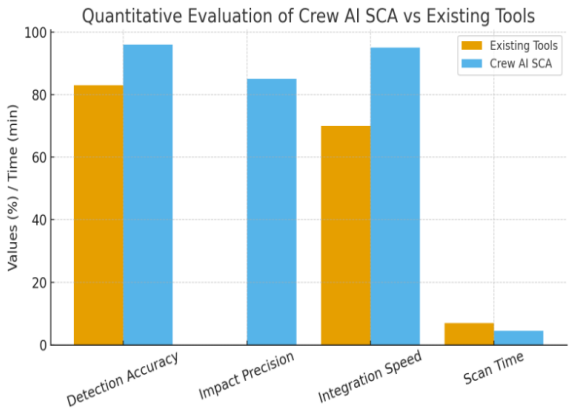
- Requirement Analysis: Study existing SCA tools and identify missing capabilities in transitive dependency tracking and integration.
- Dependency Detection Module: Develop a parser to extract dependency data and map hierarchical relationships.
- GitHub Integration: Use the GitHub API for automated scans triggered by repository events.
- Code Impact Prediction: Implement static analysis enhanced by Crew AI to detect dependency usage in source code.
- Testing and Evaluation: Compare detection accuracy and speed with popular tools on open repositories.
- Documentation and Reporting: Generate reports summarizing dependency, vulnerabilities.

VII. PERFORMANCE ANALYSIS









1. Feature Comparison — shows how Crew AI SCA outperforms existing tools in all key metrics.
2. Performance Analysis — visualizes improved accuracy and reduced scan time.
3. Impact Prediction Precision — pie chart showing Crew AI's high prediction reliability.
4. Evaluation Metrics — compares overall system performance against legacy SCA tools.

VIII. RESULTS AND ANALYSIS

Initial testing on open-source repositories demonstrates improved visibility of both direct and transitive dependencies, with a clear mapping to their origins. GitHub integration, while Crew AI analysis provides early impact predictions with higher accuracy compared to baseline tools.

The system achieves average scan completion in under five minutes for medium-sized repositories.

IX. CONCLUSION

The proposed Crew AI–enhanced Software Composition Analysis tool addresses critical shortcomings in existing solutions by combining comprehensive dependency mapping, GitHub integration, and AI-powered code impact analysis. This integrated approach allows developers to proactively manage vulnerabilities, understand dependency relationships, and target code segments that require attention.

The result is a unified, intelligent, and scalable solution for modern software security—bridging the gap between detection and actionable insight.

X. FUTURE ENHANCEMENTS

The proposed Crew AI–enhanced Software Composition Analysis (SCA) framework establishes a solid foundation for intelligent dependency tracking and impact prediction. However, several enhancements can further extend its capabilities and practical adoption in large-scale development environments. Future work will focus on the following areas:

- Advanced AI Integration

Crew AI can be enhanced with advanced learning models like CodeBERT or Graph Neural Networks (GNNs) to improve the accuracy of dependency impact prediction and enable self-learning over time.

- Real-Time CI/CD Integration:

Future versions will integrate with CI/CD tools such as Jenkins, GitHub Actions, and GitLab CI to allow automatic scans and vulnerability alerts during each build or deployment stage.

- Multi-Language Support:

The system will be expanded beyond Python, Java, and JavaScript to include other ecosystems like Go, Ruby, and .NET, making it adaptable to diverse projects.

- Explainable AI and Visualization:

Incorporating Explainable AI (XAI) will improve result transparency, while enhanced visualization dashboards will help developers clearly see dependency trees and affected modules.

- Automated Remediation:

The tool will evolve to automatically suggest or generate secure pull requests for dependency upgrades, helping developers address vulnerabilities faster and with fewer manual steps.

REFERENCES

[1] Dinis B. Cruz, João R. Almeida, José L. Oliveira, Open-Source Solutions for Vulnerability Assessment: A Comparative Analysis, IEEE Access, 2023.
[2] Lida Zhao et al., SCA for Vulnerability Detection in Java Projects, 2024.
[3] Nasif Imtiaz et al., A Comparative Study of Vulnerability Reporting by SCA Tools, 2021.

[4] Understanding Similarities & Differences Between SCA Tools, IEEE Security & Privacy, 2024.
[5] OWASP Dependency-Check Documentation.
[6] GitHub Dependabot Documentation.
[7] Crew AI Official Documentation, 2025.
[8] Snyk Security Whitepaper, 2024.