MANIPAL
ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

Static Code Analyzer Tool

Mini-Project Synopsis

submitted to

Manipal School of Information Sciences, MAHE, Manipal

Reg. Number	Name	Branch
241059027	Anusha S Patil	CYS
241059032	K Anusha Rao	CYS
241059033	Srinivas	CYS

26/01/2025

MANIPAL SCHOOL OF INFORMATION SCIENCES
MANIPAL
(A constituent unit of MAHE, Manipal)

Table of Contents

Project Description

Good code quality is a fundamental need in today's modern world. Since source code is used universally to develop every software that runs on an electronic device, it is essential that source code must be well written and maintained in order to handle updates and enhancements.

When a software developer or code security analyst needs to analyze source code to detect security flaws and maintain secure, high-quality code, manual review may not uncover all issues. Even experienced security analysts can overlook vulnerabilities. A source code analysis tool addresses this challenge by automatically and efficiently identifying potential security flaws without executing the code. It serves as a reliable complement to manual inspection, enhancing accuracy and thoroughness. A Static Code Analyzer is a software tool that examines source code for potential errors, vulnerabilities, and adherence to coding standards without executing the program. This report provides an in-depth view of a Python-specific static code analyser, detailing its architecture, functionality, and use in maintaining high-quality Python projects.

Objectives

- Identify Common Code Defects and Vulnerabilities.
- Develop a Basic Static Code Analysis Tool.
- Provide an interface for users to upload, scan, and review code analysis reports.

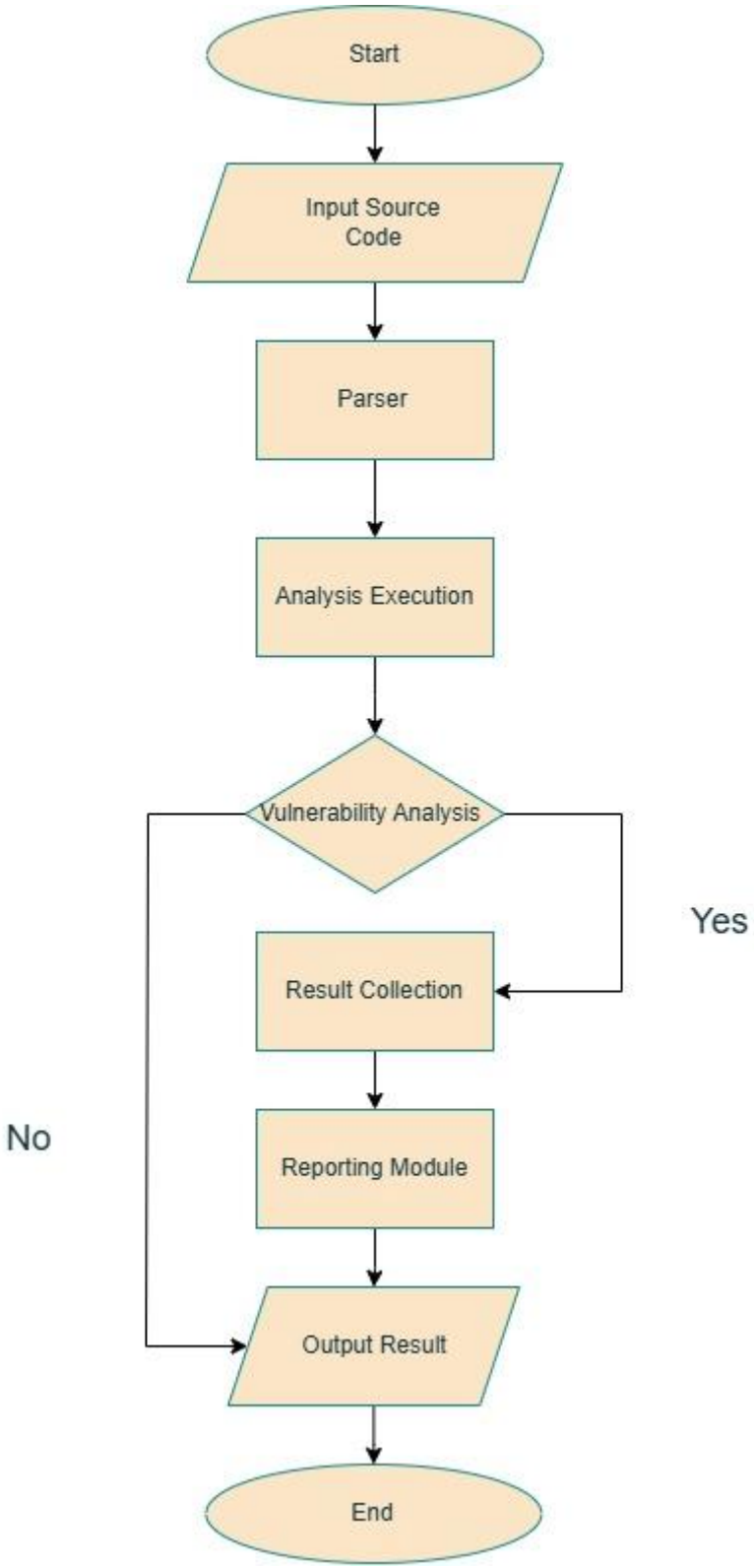
Literature Survey

The literature survey gives a brief overview about the various security vulnerabilities which can be detected by a Static code Analyzer and also the existing tools available.

Paper Title	Author(s)	Summery	Tools	Future Scope
Evaluating Python Static Code Analysis Tools Using FAIR Principles	H. B. Hassan, Q. I. Sarhan and Á. Beszédes	Evaluates Python static analysis tools using FAIR principles to enhance usability.	Prospector, Pylint, SonarQube, Pynocle etc	Create a user-friendly web application allowing developers to choose and evaluate static code analysis tools based on their specific programming languages and analysis needs
Static Code Analyser to Enhance Developer Productivity	D. R. I. Peiris and N. Kodagoda	Developed a tool to improve coding practices and developer productivity through static analysis. Used Abstract Syntax Trees to build the analyzer.	Nil	Improve the algorithm to handle issues such as tightly coupled code, collusion etc. A.I and ML based learning could be integrated to enhance the value of this software by sniffing issues using prediction-based learning.
Towards More Sophisticated Static Analysis Methods of	H. Gulabovska and Z. Porkoláb	Investigates advanced static analysis methods like symbolic execution	CodeSonar, Klocwork, Coverity, PyLint, Pyflakes	Symbolic execution-based analyzer tools for Python can use powerful SAT solvers, such as Z3.

Python Programs		compared to traditional ones.		
Static Type Analysis for Python	T. Dong, L. Chen, Z. Xu and B. Yu	Developed a tool, PType, to make type analysis including constructing a type system, preprocessing the source code, making type annotations for classes, functions and built-in modules, making type inference based on constraint graph.	PType	Expand type checking capabilities and integrate with other tools.
A Large-Scale Security-Oriented Static Analysis of Python Packages in PyPI	J. Ruohonen, K. Hjerppe and K. Rindell	Examines security issues in Python packages using static analysis to identify vulnerabilities.	Bandit	Software metrics can be collected to make the simple predictive approach more interesting theoretically. Longitudinal analysis is required for assessing the robustness of the effects reported.

Block Diagram/Flowchart



Applications

- **Immediate Feedback on Code Issues:** Source code analysis tools, also known as SAST tools, provide instant feedback to developers, helping them address vulnerabilities and errors early in the development process rather than later in the Software Development Life Cycle (SDLC).
- **Improved Code Quality from the Start:** These tools encourage developers to focus on creating secure and high-quality code right from the beginning, reducing the risk of introducing security flaws during development.
- **Cost-Effective Solutions for Security:** While many commercial tools are expensive, there are free and open-source static code analysis tools available, making them ideal for startups and freelancers who need robust security solutions without breaking the budget.
- **Wide Adoption Across Industries:** Due to the increasing demand for secure coding practices, these tools are widely adopted across organizations, ranging from individual developers to large enterprises, to ensure code safety and compliance with standards.

Software & Hardware Requirements

Hardware Requirements

Component	Specification
Processor	Dual-core CPU (e.g., Intel Core i3 or higher)
RAM	4 GB
Disk Space	10 GB

Software Requirements

Component	Specification
OS	Windows, Linux
IDE	VS Code
Plugins	PyLint

References

[1] H. B. Hassan, Q. I. Sarhan and Á. Beszédes, "Evaluating Python Static Code Analysis Tools Using FAIR Principles," in IEEE Access, vol. 12, pp. 173647-173659, 2024, doi: 10.1109/ACCESS.2024.3503493.

[2] D. R. I. Peiris and N. Kodagoda, "Static Code Analyser to Enhance Developer Productivity," 2023 IEEE 8th International Conference for Convergence in Technology (I2CT), Lonavla, India, 2023, pp. 1-6, doi: 10.1109/I2CT57861.2023.10126395.

[3] H. Gulabovska and Z. Porkoláb, "Towards More Sophisticated Static Analysis Methods of Python Programs," *2019 IEEE 15th International Scientific Conference on Informatics*, Poprad, Slovakia, 2019, pp. 000225-000230, doi: 10.1109/Informatics47936.2019.9119307.

[4] T. Dong, L. Chen, Z. Xu and B. Yu, "Static Type Analysis for Python," *2014 11th Web Information System and Application Conference*, Tianjin, China, 2014, pp. 65-68, doi: 10.1109/WISA.2014.20.

[5] J. Ruohonen, K. Hjerppe and K. Rindell, "A Large-Scale Security-Oriented Static Analysis of Python Packages in PyPI," in 2021 18th International Conference on Privacy, Security and Trust (PST), Auckland, New Zealand, 2021, pp. 1-10, doi: 10.1109/PST52912.2021.9647791.