https://careertiq.com/wp-content/uploads/2023/11/Logo.png

**TEAM NAME:** ALFOR

**DATE :** 16-JUNE-2024

**COLLEGE NAME:** KCE

**DEPARTMENT:** BE-CSE-CYBER SECURITY

**TITILE:**

**ENHANCING WEB APPLICATION SECURITY BY STATIC CODE ANALYSIS**

**TEAM MEMBERS:**

1. Charan. k -717821Y107
2. Suman Aadithya. R -717821Y155
3. Naveen. A – 717821Y133

**AIM:**

The aim of this project is to enhance web application security by integrating static code analysis into the development process. This will identify and mitigate potential security vulnerabilities early, reducing the risk of breaches and ensuring compliance with security standards**.**

**OBJECTIVE:**

* Conduct a comprehensive security assessment of the web application by reviewing its source code.
* Utilize static code analysis techniques to systematically scan the codebase.
* Identify potential vulnerabilities such as:

- Insecure data handling practices

- Weak authentication and authorization mechanisms

- Inadequate access controls

* Detect and address these vulnerabilities early in the development lifecycle.
* Ensure the web application is secure, reliable, and compliant with industry security standards.
* Mitigate the risk of security breaches and protect sensitive user data.
* Enhance the overall security posture of the web application.
* Implement the project using Python and JavaScript for static code analysis and remediation.

**METHODOLOGY:**

**Planning:**

**Gather Requirements:**

* **Functional Requirements**: List the features the tool must have (e.g., vulnerability detection, reporting).
* **Non-functional Requirements**: Consider performance, scalability, and usability aspects.

**Research Existing Tools**

* Study existing static code analysis tools to understand their strengths and weaknesses.
* Identify gaps that your tool can fill.

**Implementation:**

**Set Up the Development Environment**

* Choose a programming language (e.g., Python).
* Set up version control, testing frameworks, and CI/CD pipelines.

**Code and Documentation**

* Write clean, maintainable code following best practices.
* Document the code and provide user manuals and developer guides.

**Testing:**

**Unit Testing**

* Write unit tests for individual components to ensure they work as expected.

**Integration Testing**

* Test how the core components interact with each other.

**Performance Testing**

* Evaluate the tool’s performance, particularly on large codebases, to ensure it is efficient.

**Security Testing**

* Ensure the tool itself is secure and cannot be exploited.

**Deployment:**

**Package the Tool**

* Create installable packages or Docker images.

**Deploy to Users**

* Make the tool available to users via a distribution platform (e.g., PyPI for Python packages).

**Existing Tools Disadvantages:**

**Limited Language Support**

* **Explanation**: Not all tools support every programming language or framework.
* **Impact**: This can be problematic for projects that use multiple languages or less commonly used technologies.

**Limited Scope**: Focuses primarily on security issues, so it may miss other types of code quality issues.

**TOOLS:**

**Programming Languages:**

Python

JavaScript

**Development Environment:**

**IDE**: PyCharm or Visual Studio Code.

**Version Control**: Git + GitHub.

**Integration Testing:**

**Docker**: To create consistent testing environments.

**Reporting:**

**Documentation**: Sphinx.

**Reports**: Generate HTML reports with details on findings.

**GITHUB REPOSITORY LINKS:**

s