# Chapter 2: Analysis and Requirements

## 2.1 Problem Analysis

### 2.1.1 Problem Statement

Modern software developers need practical tools to help them understand and apply security analysis techniques to their code. Most available tools are either too complex, not integrated into the development environment, or do not support both static and dynamic analysis in a user-friendly way. This project aims to provide a VS Code extension that allows developers to easily run either static or dynamic security analysis on C/C++ code, helping them learn about vulnerabilities and secure coding practices.

### 2.1.2 Problem Domain Analysis

- \*\*Software Security\*\*: Detecting vulnerabilities in C/C++ code.

- \*\*Development Environment\*\*: Integration with VS Code, a popular editor among developers.

- \*\*Containerization\*\*: Using Docker to ensure analyses are reproducible and safe.

- \*\*User Experience\*\*: Simple commands and clear results for practical use.

### 2.1.3 Analysis Model (UML)

\*\*Domain Model Diagram:\*\*

```

classDiagram

Developer "1" -- "1" VSCodeExtension : uses

VSCodeExtension "1" -- "1" StaticAnalysis : triggers

VSCodeExtension "1" -- "1" DynamicAnalysis : triggers

StaticAnalysis "1" -- "1" DockerContainer : runs in

DynamicAnalysis "1" -- "1" DockerContainer : runs in

StaticAnalysis "1" -- "\*" Vulnerability : detects

DynamicAnalysis "1" -- "\*" RuntimeIssue : detects

```

---

## 2.2 System Requirements

### 2.2.1 Functional Requirements

| REQ-ID | Description |

|--------|-------------|

| REQ-1 | The system shall allow the user to run static analysis on C/C++ code. |

| REQ-2 | The system shall allow the user to run dynamic analysis on C/C++ code. |

| REQ-3 | The system shall integrate as a VS Code extension. |

| REQ-4 | The system shall run each analysis in a Docker container. |

| REQ-5 | The system shall display analysis results in the VS Code Problems panel. |

| REQ-6 | The system shall provide clear feedback and error messages. |

| REQ-7 | The system shall allow the user to configure analysis settings. |

| REQ-8 | The system shall support keyboard shortcuts for running analyses. |

### 2.2.2 Nonfunctional Requirements

| REQ-ID | Category | Priority | Description |

|--------|--------------|----------|-------------|

| REQ-9 | Usability | High | The extension should be easy to use for developers. |

| REQ-10 | Performance | Medium | Analyses should complete within 5 minutes for files up to 10,000 lines. |

| REQ-11 | Compatibility| High | The extension should work on Windows, macOS, and Linux with Docker installed. |

| REQ-12 | Security | High | Analyses must run in isolated containers. |

| REQ-13 | Maintainability | Medium | The codebase should be well-documented for future contributors. |

---

## 2.3 Functional Requirements Specification

### 2.3.1 Stakeholders

- \*\*Developers\*\*: Primary users, running analyses on their code.

- \*\*Technical Leads\*\*: May use the tool to demonstrate vulnerabilities or review code.

- \*\*QA Engineers\*\*: May use the tool to help review or test code.

### 2.3.2 Actors and Goals

| Actor | Type | Goals |

|------------|--------|-------|

| Developer | Human | Run static or dynamic analysis on their code, learn about vulnerabilities. |

| Technical Lead | Human | Demonstrate code analysis in team meetings or reviews. |

| VS Code | System | Host the extension and display results. |

| Docker | System | Provide isolated environment for analysis. |

### 2.3.3 Use Cases

\*\*Use Case 1: Run Static Analysis\*\*

- \*\*Actor\*\*: Developer

- \*\*Goal\*\*: Analyze code for vulnerabilities without running it.

- \*\*Scenario\*\*:

1. Developer opens a C/C++ file in VS Code.

2. Developer triggers static analysis (via command or shortcut).

3. Extension runs static analysis in Docker.

4. Results are shown in the Problems panel.

\*\*Use Case 2: Run Dynamic Analysis\*\*

- \*\*Actor\*\*: Developer

- \*\*Goal\*\*: Analyze code for runtime issues (e.g., memory leaks).

- \*\*Scenario\*\*:

1. Developer opens a C/C++ file in VS Code.

2. Developer triggers dynamic analysis (via command or shortcut).

3. Extension runs dynamic analysis in Docker.

4. Results are shown in the Problems panel.

\*\*Use Case Diagram:\*\*

```

usecaseDiagram

actor Developer

actor TechnicalLead

Developer --> (Run Static Analysis)

Developer --> (Run Dynamic Analysis)

TechnicalLead --> (Run Static Analysis)

TechnicalLead --> (Run Dynamic Analysis)

```

### 2.3.4 System Sequence Diagrams

\*\*Sequence: Run Static Analysis\*\*

```

sequenceDiagram

participant Developer

participant VSCode

participant Extension

participant Docker

Developer->>VSCode: Open C/C++ file

Developer->>Extension: Run Static Analysis

Extension->>Docker: Start container, run analysis

Docker-->>Extension: Return results

Extension->>VSCode: Show results in Problems panel

```

\*\*Sequence: Run Dynamic Analysis\*\*

```

sequenceDiagram

participant Developer

participant VSCode

participant Extension

participant Docker

Developer->>VSCode: Open C/C++ file

Developer->>Extension: Run Dynamic Analysis

Extension->>Docker: Start container, run analysis

Docker-->>Extension: Return results

Extension->>VSCode: Show results in Problems panel

```

---

## 2.4 Requirements Elicitation and Analysis Process

- \*\*Literature Review\*\*: Studied existing tools and VS Code extensions.

- \*\*Interviews\*\*: Asked developers and technical leads about their needs.

- \*\*Prototyping\*\*: Built early versions and gathered feedback from users.

- \*\*Iterative Refinement\*\*: Improved requirements and design based on feedback and testing.