Network Monitoring System - Requirements Specification

Table of Contents

1. Functional Requirements
2. Non-Functional Requirements
3. User Stories
4. System Requirements
5. Interface Requirements
6. Data Requirements
7. Security Requirements
8. Performance Requirements
9. Compliance Requirements
10. File-to-Requirement Mapping

Functional Requirements

FR-001: Real-time Packet Capture

Description: The system shall capture network packets in real-time from specified network interfaces. **Priority**: High **Implementation**: `src/core/NetworkMonitor.cpp`, `include/core/NetworkMonitor.hpp` **Dependencies**: libpcap library **Acceptance Criteria**:

- Capture packets from any available network interface
- Support promiscuous mode for complete traffic monitoring
- Handle packet capture rates up to 1 Gbps
- Apply Berkeley Packet Filter (BPF) expressions for selective capture

FR-002: Protocol Analysis

Description: The system shall parse and analyze multiple network protocols. **Priority**: High **Implementation**: `src/protocols/Packet.cpp`, `include/protocols/Packet.hpp` **Supported Protocols**:

- Ethernet (Layer 2)
- IPv4/IPv6 (Layer 3)
- TCP/UDP/ICMP (Layer 4)
- HTTP/HTTPS/DNS/DHCP/ARP (Application Layer)

Acceptance Criteria:

- Correctly parse protocol headers
- Extract source/destination addresses and ports
- Identify protocol types and payload data
- Handle malformed packets gracefully

FR-003: Statistical Analysis

Description: The system shall provide real-time statistical analysis of network traffic. **Priority**: High **Implementation**: `src/analysis/Statistics.cpp`, `include/analysis/Statistics.hpp` **Metrics**:

- Total packet count and byte count
- Protocol distribution
- Top hosts by traffic volume
- Active connections tracking
- Bandwidth utilization over time

Acceptance Criteria:

- Update statistics in real-time (< 1 second latency)
- Maintain historical data for trending
- Support configurable time windows

FR-004: Data Persistence

Description: The system shall store captured packet data for historical analysis. **Priority**: Medium **Implementation**: `src/storage/DataStore.cpp`, `include/storage/DataStore.hpp` **Storage Features**:

- SQLite database for packet storage
- Configurable retention policies
- Batch insertion for performance
- Query interface for historical data

Acceptance Criteria:

- Store packets with full metadata
- Support time-range queries
- Implement automatic cleanup of old data
- Handle database corruption gracefully

FR-005: Graphical User Interface

Description: The system shall provide an intuitive GUI for monitoring and analysis. **Priority**: High **Implementation**: `src/gui/`, `include/gui/` **GUI Components**:

- Main window with tabbed interface
- Real-time statistics display
- Active connections viewer
- Packet list with details
- Bandwidth charts and graphs
- Filter configuration dialog

Acceptance Criteria:

- Responsive interface with < 1 second update intervals
- Support for multiple simultaneous views
- Configurable display options
- Export functionality for data and charts

FR-006: Command Line Interface

Description: The system shall provide a CLI for automated and scripted operations. **Priority**: Medium **Implementation**: `src/cli/CommandLineInterface.cpp`, `include/cli/CommandLineInterface.hpp` **CLI Features**:

- Start/stop monitoring commands
- Real-time statistics display
- Filter configuration
- Data export capabilities

Acceptance Criteria:

- Support all major GUI functions
- Provide scriptable interface
- Return appropriate exit codes
- Support batch operations

FR-007: Configuration Management

Description: The system shall support flexible configuration options. **Priority**: Medium **Implementation**: `src/config/ConfigManager.cpp`, `include/config/ConfigManager.hpp` **Configuration Options**:

- Network interface selection
- Capture filters and parameters
- Storage settings and retention
- GUI preferences and themes
- Logging levels and destinations

Acceptance Criteria:

- Support configuration files and command-line arguments
- Validate configuration parameters
- Provide default configurations
- Support runtime configuration updates

FR-008: Logging and Monitoring

Description: The system shall provide comprehensive logging for operations and debugging. **Priority**: Medium **Implementation**: `src/utils/Logger.cpp`, `include/utils/Logger.hpp` **Logging Features**:

- Configurable log levels (DEBUG, INFO, WARNING, ERROR, FATAL)
- File and console output
- Structured log format
- Log rotation and archival

Acceptance Criteria:

- Log all significant operations
- Support multiple output destinations
- Implement log level filtering
- Provide performance metrics logging

Non-Functional Requirements

NFR-001: Performance

Description: The system shall maintain high performance under heavy network loads. **Requirements**:

- Handle packet rates up to 1 Gbps
- GUI response time < 1 second
- Database operations < 100ms average
- Memory usage < 1GB under normal load

Implementation: Multi-threaded architecture in `src/core/NetworkMonitor.cpp`

NFR-002: Scalability

Description: The system shall scale to handle increasing network traffic and data volumes. **Requirements**:

- Support multiple network interfaces simultaneously
- Handle databases up to 100GB

- Scale to 10,000+ concurrent connections
- Support distributed deployment (future)

Implementation: Queue-based architecture with configurable buffer sizes

NFR-003: Reliability

Description: The system shall operate reliably with minimal downtime. **Requirements**:

- 99.9% uptime during monitoring periods
- Graceful handling of network interface failures
- Automatic recovery from database corruption
- Memory leak prevention

Implementation: Error handling throughout codebase, particularly in `src/core/NetworkMonitor.cpp`

NFR-004: Usability

Description: The system shall be easy to use for network administrators and analysts. **Requirements**:

- Intuitive GUI design following platform conventions
- Comprehensive help documentation
- Keyboard shortcuts for common operations
- Accessibility compliance (WCAG 2.1)

Implementation: Qt6-based GUI in `src/gui/` directory

NFR-005: Maintainability

Description: The system shall be easy to maintain and extend. **Requirements**:

- Modular architecture with clear interfaces
- Comprehensive code documentation
- Unit test coverage > 80%
- Coding standards compliance

Implementation: Modular design with separate directories for each component

User Stories

Epic: Network Monitoring

NET-001: As a network administrator, I want to monitor real-time network traffic so that I can identify performance issues and security threats. **Files**: `src/core/NetworkMonitor.cpp`, `src/gui/MainWindow.cpp`

NET-002: As a security analyst, I want to filter network traffic by protocol and host so that I can focus on specific security events. **Files**: `src/core/NetworkMonitor.cpp`, `src/gui/FilterDialog.cpp`

NET-003: As a network engineer, I want to view detailed packet information so that I can troubleshoot network problems. **Files**: `src/protocols/Packet.cpp`, `src/gui/PacketsWidget.cpp`

Epic: Data Analysis

NET-004: As a network analyst, I want to view traffic statistics and trends so that I can understand network usage patterns. **Files**: `src/analysis/Statistics.cpp`, `src/gui/StatisticsWidget.cpp`

NET-005: As a capacity planner, I want to monitor bandwidth utilization over time so that I can plan for network upgrades. **Files**: `src/analysis/Statistics.cpp`, `src/gui/BandwidthWidget.cpp`

NET-006: As a security analyst, I want to track active connections so that I can identify suspicious network activity. **Files**: `src/analysis/Statistics.cpp`, `src/gui/ConnectionsWidget.cpp`

Epic: Data Management

NET-007: As a compliance officer, I want to store network traffic data for historical analysis so that I can meet regulatory requirements. **Files**: `src/storage/DataStore.cpp`

NET-008: As a forensic analyst, I want to query historical packet data so that I can investigate security incidents. **Files**: `src/storage/DataStore.cpp`

Epic: System Administration

NET-009: As a system administrator, I want to configure monitoring parameters so that I can optimize system performance. **Files**: `src/config/ConfigManager.cpp`, `config/default.conf`

NET-010: As a DevOps engineer, I want to integrate the monitoring system with automation tools so that I can include it in deployment pipelines. **Files**: `src/cli/CommandLineInterface.cpp`

System Requirements

Hardware Requirements

- **Minimum**: 4GB RAM, 2-core CPU, 10GB storage
- **Recommended**: 8GB RAM, 4-core CPU, 100GB storage
- **Network**: Gigabit Ethernet adapter with promiscuous mode support

Software Requirements

- **Operating System**: Linux (Ubuntu 20.04+), Windows 10+, macOS 10.15+
- **Compiler**: GCC 10+, Clang 12+, or MSVC 2019+
- **Libraries**: Qt6.2+, libpcap 1.9+, SQLite 3.35+, Boost 1.75+
- **Build Tools**: CMake 3.15+, Make or Ninja

Network Requirements

- **Privileges**: Root/Administrator access for packet capture
- **Interfaces**: Support for Ethernet, Wi-Fi, and virtual interfaces
- **Protocols**: IPv4/IPv6 network stack

Interface Requirements

GUI Interface Requirements

Implementation: `src/gui/` directory

- **Framework**: Qt6 with native look and feel

- **Layout**: Tabbed interface with dockable widgets
- **Charts**: Real-time updating charts using Qt Charts
- **Tables**: Sortable and filterable data tables
- **Dialogs**: Modal dialogs for configuration and filters

CLI Interface Requirements

Implementation: `src/cli/CommandLineInterface.cpp`

- **Commands**: Start, stop, status, filter, export, help
- **Output**: Structured text output suitable for parsing
- **Input**: Interactive command mode and batch processing
- **Integration**: Support for shell scripting and automation

API Interface Requirements (Future)

- **Protocol**: gRPC with Protocol Buffers
- **Endpoints**: Monitoring control, statistics retrieval, data export
- **Authentication**: Token-based authentication
- **Rate Limiting**: Configurable request rate limits

Data Requirements

Packet Data Schema

Implementation: `src/protocols/Packet.cpp`, `src/storage/DataStore.cpp`

```
CREATE TABLE packets (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    timestamp INTEGER NOT NULL,
    protocol TEXT NOT NULL,
    source_address TEXT NOT NULL,
    destination_address TEXT NOT NULL,
    source_port INTEGER,
    destination_port INTEGER,
    length INTEGER NOT NULL,
    raw_data BLOB,
    is_fragmented BOOLEAN,
```

```
    is_malformed BOOLEAN
);
```

Statistics Data Schema

Implementation: `src/analysis/Statistics.cpp`

- Protocol statistics (packet count, byte count, error count)
- Host statistics (traffic volume, protocol distribution)
- Connection statistics (duration, packet count, retransmissions)
- Bandwidth history (timestamp, bits per second)

Configuration Data Schema

Implementation: `config/default.conf`

```
[general]
log_level = info
database = network_monitor.db

[monitoring]
interface = eth0
promiscuous_mode = true
buffer_size = 65536

[storage]
max_packets = 1000000
cleanup_interval = 3600
```

Security Requirements

SEC-001: Access Control

Description: The system shall implement proper access controls for sensitive operations.
Requirements:

- Require elevated privileges for packet capture

- Implement user authentication for GUI access (future)
- Provide role-based access control (future)

Implementation: Privilege checks in `src/core/NetworkMonitor.cpp`

SEC-002: Data Protection

Description: The system shall protect captured network data from unauthorized access.
Requirements:

- Encrypt stored packet data (future)
- Secure database file permissions
- Implement data anonymization options

Implementation: File permissions in `src/storage/DataStore.cpp`

SEC-003: Input Validation

Description: The system shall validate all user inputs to prevent security vulnerabilities.
Requirements:

- Validate BPF filter expressions
- Sanitize configuration file inputs
- Prevent SQL injection in database queries

Implementation: Input validation throughout codebase

Performance Requirements

PERF-001: Packet Processing

Description: The system shall process packets efficiently to avoid packet loss.
Requirements:

- Process 1M packets per second
- Maintain < 1% packet loss under normal load
- Use multi-threading for parallel processing

Implementation: Multi-threaded architecture in `src/core/NetworkMonitor.cpp`

PERF-002: Database Performance

Description: The system shall maintain database performance under high write loads.
Requirements:

- Batch database insertions
- Implement write-ahead logging
- Support database optimization

Implementation: Batch processing in `src/storage/DataStore.cpp`

PERF-003: GUI Responsiveness

Description: The system shall maintain responsive GUI under all conditions.
Requirements:

- Update GUI elements < 1 second
- Use background threads for data processing
- Implement progressive loading for large datasets

Implementation: Timer-based updates in `src/gui/MainWindow.cpp`

Compliance Requirements

COMP-001: Data Retention

Description: The system shall support configurable data retention policies.
Requirements:

- Automatic deletion of old packet data
- Configurable retention periods
- Audit trail for data deletion

Implementation: Cleanup routines in `src/storage/DataStore.cpp`

COMP-002: Privacy Protection

Description: The system shall protect personally identifiable information in network traffic. **Requirements**:

- Data anonymization options
- PII detection and masking
- Compliance with GDPR and similar regulations

Implementation: Privacy features in `src/protocols/Packet.cpp`

File-to-Requirement Mapping

Core Components

File	Requirements	User Stories
`src/core/NetworkMonitor.cpp`	FR-001, NFR-001, PERF-001	NET-001, NET-002
`src/protocols/Packet.cpp`	FR-002, COMP-002	NET-003
`src/analysis/Statistics.cpp`	FR-003, PERF-003	NET-004, NET-005, NET-006
`src/storage/DataStore.cpp`	FR-004, PERF-002, COMP-001	NET-007, NET-008

GUI Components

File	Requirements	User Stories
`src/gui/MainWindow.cpp`	FR-005, NFR-004, PERF-003	NET-001
`src/gui/StatisticsWidget.cpp`	FR-005, NFR-004	NET-004
`src/gui/ConnectionsWidget.cpp`	FR-005, NFR-004	NET-006
`src/gui/PacketsWidget.cpp`	FR-005, NFR-004	NET-003
`src/gui/BandwidthWidget.cpp`	FR-005, NFR-004	NET-005

src/gui/FilterDialog.cpp	FR-005, SEC-003	NET-002

Utility Components

File	Requirements	User Stories
src/cli/CommandLineInterface.cpp	FR-006, NFR-004	NET-010
src/config/ConfigManager.cpp	FR-007, SEC-003	NET-009
src/utils/Logger.cpp	FR-008, NFR-003	NET-009

Configuration Files

File	Requirements	User Stories
config/default.conf	FR-007, COMP-001	NET-009
CMakeLists.txt	NFR-005	-
README.md	NFR-004	-

Generated on: $(date) Requirements Version: 1.0 Last Updated: $(date)