Healthcare and IoT Device Vulnerability Testbed

Software Requirements Document

Team Members
Justin Bower
Nathan Maloney
Ipule Pipi

Faculty Advisors
Dr. Snedah Sudhakaran
Dr. Abdullah Aydeger

Client
Prospective IoT and Medical Device Manufacturers and Researchers

1. Introduction
1.1 Purpose

The purpose of this document is to describe the functional and non-functional requirements for the Healthcare and IoT Device Vulnerability Testbed system. This document defines what the system is expected to do, how users will interact with it, and the expected performance and security constraints.

The requirements defined in this document will serve as a guideline for development and implementation of the system. They will also be used as a reference when evaluating whether the final system meets the expected functionality and performance.

1.2 Scope

The Healthcare and IoT Device Vulnerability Testbed is a platform designed to help researchers and engineers analyze the security of IoT and healthcare devices. The platform provides tools and infrastructure that allow users to connect devices, extract firmware, perform firmware analysis, and execute vulnerability testing scripts.

The system will provide a centralized environment that simplifies complex reverse engineering and vulnerability analysis tasks through a web interface. Users will be able to run automated tools for firmware inspection and vulnerability testing while the system manages the execution environment and stores analysis results.

The platform is intended to support tasks such as firmware extraction, entropy analysis, binary analysis, and vulnerability testing using both built-in tools and user-provided scripts. The system will provide secure execution environments to prevent unsafe code from affecting the host infrastructure.

This system is designed primarily for research, educational, and testing purposes.

1.3 Definitions and Acronyms

IoT – Internet of Things devices that connect to networks and communicate with other systems.

Firmware – Software embedded within hardware devices that controls device behavior.

Binary – Compiled machine code or firmware files.

Entropy Analysis – Statistical analysis used to detect encryption or compression within binary data.

Testbed – A controlled environment used to test software or hardware systems.

API – Application Programming Interface used for communication between system components.

GUI – Graphical User Interface used by users to interact with the system.

Sandbox – A secure, isolated environment where code can be executed safely.

1.4 Intended Users

The system is intended for use by the following types of users:

Security researchers who want to analyze device firmware and test for vulnerabilities.

Device manufacturers who want to evaluate the security of their products.

Students and educators studying embedded systems security.

Engineers who need tools for firmware extraction and vulnerability testing.

Users of the system are expected to have a basic understanding of cybersecurity concepts and device analysis tools.

2. Overall System Description

2.1 Product Perspective

The Healthcare and IoT Device Vulnerability Testbed will be implemented as a web-based platform that integrates multiple security analysis tools and execution environments. Users will interact with the system through a browser-based interface.

The frontend interface will allow users to manage devices, upload firmware files, execute analysis tools, and view results. The backend infrastructure will manage tool execution, data storage, and communication between system components.

The system architecture includes several major components:

The frontend web interface which allows users to interact with the system.

The backend API that processes user requests and coordinates system operations.

The execution environment where analysis tools and scripts are run.

Data storage systems used to store firmware files, analysis results, and device metadata.

External analysis tools used for firmware extraction and vulnerability detection.

These components work together to provide a secure and scalable environment for performing IoT device security analysis.

2.2 User Classes and Characteristics

Security Researchers
These users will perform firmware analysis and vulnerability testing. They are expected to be technically experienced and familiar with security tools.

Manufacturers and Engineers
These users will test their devices for vulnerabilities before deployment. They may have experience with firmware and embedded systems but may not be security experts.

Students and Educators
These users will use the system as a learning tool for studying device security and vulnerability analysis.

2.3 Operating Environment

The system will operate as a web-based platform accessible through modern web browsers.

The backend infrastructure will run in a cloud environment and may utilize services such as serverless functions, containers, and cloud storage.

The analysis tools will run in isolated execution environments to prevent unsafe code from affecting other parts of the system.

The platform must support execution of common firmware analysis tools and scripts while maintaining security and isolation.

2.4 Design Constraints

The system must safely execute potentially unsafe scripts and analysis tools.

The system must isolate user execution environments to prevent interference between users.

The system must limit resource usage to prevent abuse of compute resources.

The platform must maintain security controls to prevent unauthorized access to stored data and system infrastructure.

The system should be designed in a modular way so additional analysis tools can be integrated in the future.

2.5 Assumptions and Dependencies

The system assumes that users will provide valid firmware files or device access for analysis.

The system depends on the availability of external analysis tools used for firmware inspection and extraction.

The system assumes access to cloud infrastructure for storage and compute resources.

The system assumes that users will follow ethical and legal guidelines when performing security testing.

3. Specific Requirements

3.1 Functional Requirements

FR1 – User Access
The system shall provide a web interface that allows users to access the platform through a web browser.

FR2 – Device Registration
The system shall allow users to register devices by providing metadata such as device name, manufacturer, and device type.

FR3 – Firmware Upload
The system shall allow users to upload firmware files for analysis.

FR4 – Firmware Storage

The system shall store uploaded firmware files and associate them with the corresponding device and analysis session.

FR5 – Firmware Extraction
The system shall allow users to run firmware extraction tools to identify file systems and embedded components within firmware images.

FR6 – Entropy Analysis
The system shall provide entropy analysis tools that allow users to identify encrypted or compressed regions within firmware files.

FR7 – Binary Analysis
The system shall allow users to run analysis tools on extracted firmware components.

FR8 – Script Execution
The system shall allow users to execute pre-approved vulnerability testing scripts within an isolated environment.

FR9 – Custom Script Execution
The system shall allow users to upload and run custom scripts within a sandboxed execution environment.

FR10 – Result Storage
The system shall store analysis outputs, logs, and generated artifacts.

FR11 – Result Viewing
The system shall allow users to view analysis results through the web interface.

FR12 – Session Tracking
The system shall maintain a record of analysis sessions and associate results with the correct user and device.

FR13 – Job Status Monitoring
The system shall display the status of analysis tasks such as queued, running, completed, or failed.

FR14 – Data Retrieval
The system shall allow users to download generated artifacts and analysis outputs.

3.2 Interface Requirements
User Interface Requirements

The system shall provide a web-based interface that allows users to interact with the platform.

The user interface shall allow users to upload firmware files, start analysis tools, and view results.

The interface shall display the status of running tasks and provide feedback to users.

The interface shall provide clear navigation between device management, analysis tools, and results.

API Interface Requirements

The backend system shall expose APIs that allow the frontend interface to request analysis operations.

The API shall support operations such as device registration, firmware upload, tool execution, and result retrieval.

The API shall return structured responses that include operation status and result metadata.

External Tool Interface Requirements

The system shall integrate external analysis tools used for firmware extraction and analysis.

The system shall capture and store output generated by these tools.

The system shall provide a mechanism for adding additional tools in the future.

3.3 Performance Requirements

The system shall respond to standard user interface requests within a reasonable time frame.

The system shall be capable of handling multiple simultaneous analysis tasks.

The system shall store firmware files and analysis results without data loss.

The system shall limit the execution time of analysis scripts to prevent resource exhaustion.

The system shall monitor system resource usage and prevent tasks from exceeding allocated limits.

3.4 Security Requirements

The system shall isolate script execution environments to prevent unauthorized access to system resources.

The system shall restrict scripts from accessing sensitive infrastructure components.

The system shall authenticate users before allowing access to system features.

The system shall maintain logs of executed analysis tasks and script activity.

The system shall protect stored firmware files and analysis data from unauthorized access.

3.5 Reliability Requirements

The system shall store analysis results persistently so that results are not lost if tasks fail.

The system shall recover gracefully from execution failures.

The system shall ensure that failures in one user's execution environment do not affect other users.

3.6 Maintainability Requirements

The system shall be designed in a modular way that allows additional analysis tools to be integrated.

The system shall separate frontend, backend, and execution environments to simplify maintenance.

The system shall provide clear documentation for developers and system administrators.