Test Plan

Healthcare and IoT Device Vulnerability Testbed

Team Members
Justin Bower
Nathan Maloney
Ipule Pipi

Faculty Advisors
Dr. Snedah Sudhakaran
Dr. Abdullah Aydeger

1. Introduction

This document describes the testing strategy used to verify that the Healthcare and IoT Device Vulnerability Testbed behaves according to the system requirements. The goal of testing is to ensure that all required system features operate correctly, reliably, and securely before the final demonstration and deployment.

Testing will be performed throughout the development process and will include functional testing, integration testing, security testing, and usability testing. Each major feature of the system will have associated test cases designed to confirm correct operation under both normal and abnormal conditions.

In addition to verifying correct functionality, the test plan is also intended to identify bugs, unexpected behavior, or security vulnerabilities within the system. This includes testing unusual input values, invalid device connections, corrupted firmware data, and malicious scripts that attempt to break the sandbox environment.

2. Test Environment

Testing will be conducted using the same tools and infrastructure used in the project implementation.

Hardware Environment

Standard development workstation

Sample IoT device connected through USB or serial interface

Network connectivity for cloud services

Software Environment

React web frontend

Backend API written in JavaScript/TypeScript

Python scripts for firmware analysis tools

Docker containers or Lambda for sandboxed script execution

AWS services including EC2, S3, and database services

Testing Tools

Binwalk

Foremost

Python test scripts

Browser developer tools

Logging tools for backend services

These tools will allow the team to simulate device connections, upload firmware images, run analysis tools, and observe system behavior.

3. Test Strategy

Testing will be divided into several categories.

Unit Testing
Individual modules such as firmware extraction scripts, entropy analysis functions, and API endpoints will be tested independently.

Integration Testing
Integration testing will verify that components communicate correctly with each other. Examples include verifying that the backend successfully sends firmware data to analysis tools and that the frontend correctly displays results returned from the API.

System Testing
Full end-to-end workflows will be tested to ensure that the complete system functions correctly from device connection to vulnerability analysis.

Security Testing

Because the system executes analysis scripts and exploitation tools, security testing will ensure that scripts cannot escape the sandbox or access unauthorized system resources.

Usability Testing
The graphical interface will be tested to ensure users can easily perform common tasks such as connecting devices, extracting firmware, and running analysis scripts.

4. Requirement to Test Case Mapping

The main system features include device connection, firmware extraction, firmware analysis, exploitation testing, custom script execution, and a web interface.

Senior Project Project Plan

Each feature will be verified with multiple test cases.

Feature 1 – Device Connection
Users should be able to connect supported IoT or medical devices through standard interfaces.

Test Cases
TC-1.1 Normal device connection
Input: Connect a supported IoT device through USB
Expected Result: System recognizes device and allows interaction.

TC-1.2 Unsupported device
Input: Connect an unsupported device type
Expected Result: System rejects device and shows error message.

TC-1.3 Device disconnect during session
Input: Disconnect device while firmware extraction is running
Expected Result: System stops operation and displays appropriate error.

TC-1.4 Multiple devices connected
Input: Connect two devices simultaneously
Expected Result: System identifies both devices without conflict.

Feature 2 – Firmware Extraction
Users should be able to extract firmware from connected devices.

Test Cases

TC-2.1 Successful firmware extraction
Input: Extract firmware from a connected device
Expected Result: Firmware binary file is generated and stored.

TC-2.2 Corrupted firmware extraction
Input: Extraction process interrupted
Expected Result: System reports failure and does not store incomplete file.

TC-2.3 Large firmware file
Input: Extract firmware larger than typical test file
Expected Result: System completes extraction without crashing.

TC-2.4 No device connected
Input: Attempt firmware extraction without device
Expected Result: System displays error.

Feature 3 – Firmware Entropy Analysis
Users should be able to analyze firmware for entropy and compression patterns.

Test Cases

TC-3.1 Standard firmware file
Input: Run entropy analysis on valid firmware
Expected Result: Graph or report displaying entropy values.

TC-3.2 Non-firmware file
Input: Upload random text file for analysis
Expected Result: System processes file but reports no meaningful structure.

TC-3.3 Extremely small file
Input: Analyze very small binary file
Expected Result: System returns entropy result without error.

TC-3.4 Extremely large file
Input: Analyze very large firmware image
Expected Result: System completes analysis within reasonable time.

Feature 4 – Automated Exploitation Scripts
Users should be able to run predefined vulnerability testing scripts.

Test Cases

TC-4.1 Known vulnerability script
Input: Run script against firmware containing known vulnerability
Expected Result: Script reports detected vulnerability.

TC-4.2 Script with invalid input

Input: Script executed on incompatible firmware
Expected Result: Script terminates safely without system crash.

TC-4.3 Script timeout
Input: Script runs longer than allowed time
Expected Result: Execution is automatically terminated.

TC-4.4 Script attempting restricted actions
Input: Script attempts to access system files outside sandbox
Expected Result: Access is blocked.

Feature 5 – Custom Script Execution
Users should be able to upload and execute their own analysis scripts.

Test Cases

TC-5.1 Valid user script
Input: Upload valid Python analysis script
Expected Result: Script executes and output is returned.

TC-5.2 Script with syntax error
Input: Upload script with syntax error
Expected Result: System reports error without crashing.

TC-5.3 Malicious script attempt
Input: Script attempts to access host file system or network
Expected Result: Script execution blocked by sandbox.

TC-5.4 Resource exhaustion script
Input: Script attempts to allocate excessive memory
Expected Result: Execution is terminated safely.

Feature 6 – Web Interface
Users should be able to perform all system operations through a graphical web interface.

Test Cases

TC-6.1 Load main dashboard
Input: User navigates to the website
Expected Result: Main interface loads successfully.

TC-6.2 Run full workflow
Input: Connect device, extract firmware, run entropy analysis
Expected Result: Entire workflow completes successfully.

TC-6.3 Invalid form inputs
Input: User submits incomplete or incorrect data
Expected Result: Interface shows validation error.

TC-6.4 Browser compatibility
Input: Access website using different browsers
Expected Result: Interface functions correctly.

5. Performance Testing

Performance tests will ensure that the system remains responsive during heavy tasks such as firmware extraction or analysis.

Example Tests

Run entropy analysis on large firmware images

Execute multiple scripts sequentially

Upload multiple firmware images simultaneously

Expected Outcome
The system should remain stable and responsive with no crashes or major delays.

6. Security Testing

Because the system runs analysis and exploitation scripts, security testing is essential.

Security tests include

Attempting sandbox escape from scripts

Uploading malicious scripts

Attempting unauthorized access to stored firmware

Input validation testing

Expected Outcome
All malicious actions should be blocked and logged.

7. Acceptance Criteria

The system will be considered successfully tested when the following conditions are met:

All major functional test cases pass.

No critical system crashes occur during testing.

Scripts cannot escape the sandbox execution environment.

Firmware extraction and analysis work correctly on sample devices.

The web interface allows users to complete the full workflow from device connection to vulnerability analysis.

Successful completion of these tests demonstrates that the system meets the functional requirements and can be used as a reliable vulnerability testing platform for IoT and medical devices.