* Software Update (SW-Update) Integration
* CVE Mitigation and Security Enhancements
* Modbus Protocol Implementation
* Board Boot Time Optimization

# Proposal for Software Update (SW-Update) Integration

## 1. Project Overview

The objective of this project is to implement a secure software update (SW-Update) mechanism for the AM437x board using TCP and UDP protocols. The system will utilize a dual-OS approach with a Main OS (OS1) and a Tiny OS (OS2) to ensure reliable firmware updates. The project is expected to be completed within 4 weeks, including server-client SW-Update code development and thorough testing.

## 2. Scope of Work (SOW)

* Implement a secure software update mechanism using SW-Update.
* Establish client-server communication using secure sockets over TCP and UDP.
* Design a dual-OS architecture to manage firmware updates.
* Allocate NAND memory with 80% for OS1 and 20% for OS2.
* Develop applications for both Main OS (Main\_OS\_app) and Tiny OS (firmware-update).
* Implement a PC-side management application (Final\_client\_pc).
* Perform rigorous testing to ensure reliability and performance.

## 3. Features & Functionalities

### NAND Memory Allocation

* 80% allocated for the Main OS (OS1) to handle primary functions and user interactions.
* 20% allocated for the Tiny OS (OS2) responsible for firmware updates.

### Update Process

* OS1 runs the Main\_OS\_app to coordinate the update process and manage user input.
* OS2 runs the firmware-update application to receive, validate, and flash firmware images.
* After successful updates, the system reboots into OS1 with the new firmware.

### Boot Sequence

* Default boot into OS1 for normal operations.
* On initiating an update, the system reboots into OS2.
* OS2 performs the update and reboots back to OS1.

### PC-Side Management

* The Final\_client\_pc application manages the update process from the PC.
* Communicates with both Main\_OS\_app on OS1 and firmware-update on OS2.
* Transfers firmware images securely using TCP or UDP.

### Update Approaches

#### MainOS\_TinyOS\_512MB\_Ram-store\_flash

* Uses RAM to create a temporary filesystem (tmpfs).
* Stores the received firmware image in RAM before flashing to NAND.

#### MainOS\_TinyOS\_512MB\_Direct\_flash

* Directly writes the firmware image to NAND in 32KB chunks.
* Reduces RAM usage and ensures faster updates.

## 4. Phased Execution Plan

| Phase | Description | Tasks | Deliverables | Timeline (Weeks) |
| --- | --- | --- | --- | --- |
| 1 | Requirements Analysis | Identify communication protocols, memory allocation | Requirement Document | 1 |
| 2 | Development | Implement SW-Update mechanism and server-client code | Working Prototype | 2 |
| 3 | Testing and Validation | Perform unit tests, integration tests, and system tests | Test Reports, Bug Fixes | 0.5 |
| 4 | Deployment | Deploy and monitor on target boards | Final Deployment Report | 0.5 |

## 5. Assumptions & Dependencies

* Secure sockets implementation will follow standard encryption practices.
* Network connectivity is stable for updates using TCP/UDP.
* Sufficient NAND memory is available as per specified allocation.
* PC-side application compatibility with Linux and Windows systems.

## 6. Next Steps

* Approve the proposal and finalize the requirements.
* Start development of SW-Update client and server applications.
* Perform memory allocation and partitioning.
* Conduct testing and validation.
* Deploy and monitor in real-world scenarios.

# Secure Vulnerability Mitigation Plan for Kernel Source and RootFS Library

## 1. Project Overview

The objective of this project is to analyze and mitigate vulnerabilities found in the kernel source and RootFS library. Our approach includes identifying applicable patches for reported CVEs and implementing necessary updates to ensure a secure and stable system.

## 2. Scope of Work (SOW)

* Conduct a detailed vulnerability analysis to verify security risks.
* Identify patch files for reported CVEs and assess feasibility.
* Apply patches or replace libraries with updated versions if necessary.
* Ensure system stability and compatibility post-update.
* Provide a structured timeline for implementation and validation.

## 3. Features & Functionalities

* Security Fixes: Patch or upgrade vulnerable components.
* Compatibility Assurance: Ensure kernel and RootFS remain stable.
* Performance Validation: Conduct thorough testing post-updates.
* Customer Confidence: Provide transparency in applied security measures.

## 4. Phased Execution Plan

| Phase | Description | Tasks | Deliverables | Assumptions | Timeline (Weeks) |
| --- | --- | --- | --- | --- | --- |
| 1 | Vulnerability Analysis | Identify CVEs, assess impact | Report on security risks | Customer provides full vulnerability details | 2 |
| 2 | Patch Identification | Search for available patches or updates | Patch Feasibility Report | Patches are available for affected components | 3 |
| 3 | Implementation & Testing | Apply patches or replace libraries | Updated Kernel & RootFS | System remains stable post-update | 5 |
| 4 | Validation & Deployment | Security testing, final verification | Test Reports, Deployment Package | Approval from customer | 2 |

## 5. Assumptions & Dependencies

* Customer provides detailed vulnerability information.
* Official patches exist and are applicable.
* System testing is required post-update.
* Customer approval is necessary before final deployment.

## 6. Next Steps

* Confirm affected kernel source and RootFS libraries.
* Approve the timeline and methodology.
* Proceed with patching, testing, and deployment.

# Modbus Protocol Implementation Proposal

## 1. Project Overview

The goal of this project is to implement and validate the Modbus protocol on the AM437x board. This involves establishing seamless communication between a Modbus TCP server (running on a PC) and a Modbus TCP client (running on the AM437x board) using CAN Bus for data transmission. The solution ensures reliable data exchange and supports future expansion to other CAN devices.

## 2. Scope of Work (SOW)

* Develop and configure a Modbus TCP server application on a Linux PC.
* Implement a Modbus TCP client application on the AM437x board.
* Enable CAN Bus communication using the AM437x board's CAN interface.
* Perform data exchange tests using a CAN Analyzer.
* Replace the CAN Analyzer with a customer-provided CAN device for final validation.
* Provide necessary support and documentation for demonstration and deployment.

## 3. Features & Functionalities

* Modbus Communication: Establish TCP/IP-based Modbus communication between the server and client.
* CAN Bus Integration: Implement data transfer between the AM437x board and the CAN Bus.
* Request and Response Management: Handle Modbus read/write requests and responses.
* Data Validation: Verify correct data transmission using the CAN Analyzer.
* Future Expandability: Allow easy integration with other CAN devices.

## 4. Phased Execution Plan

| Phase | Description | Tasks | Deliverables | Timeline (Weeks) |
| --- | --- | --- | --- | --- |
| 1 | System Setup and Configuration | Install server and client applications | Initial Environment Setup | 1 |
| 2 | Modbus Client Implementation | Develop Modbus client on AM437x | Client Application | 2 |
| 3 | CAN Bus Communication Setup | Configure CAN Bus and test data flow | Working CAN Communication | 2 |
| 4 | Testing and Validation | Perform end-to-end data transmission tests | Test Reports | 1 |
| 5 | Demo and Client Review | Conduct live demonstration | Client Feedback and Approval | 1 |

## 5. Assumptions & Dependencies

* Client provides the necessary network environment and access to the AM437x board.
* CAN Analyzer is available for initial testing.
* Final validation will be conducted with the customer-provided CAN device.
* Client provides feedback on demo and test results for further adjustments.

## 6. Next Steps

* Confirm system environment and hardware availability.
* Proceed with initial setup and configuration.
* Develop and validate the Modbus TCP client and server applications.
* Conduct end-to-end communication testing.
* Schedule a demo session with the client for validation and approval.

# Board Boot Time Optimization Proposal

## 1. Project Overview:

The objective of this project is to optimize the boot time of the AM437x board. The current boot time is approximately 1 minute and 20 seconds. Through systematic improvements in U-Boot, Linux kernel configuration, and root filesystem (RootFS) optimization, we aim to reduce the boot time to around 30 to 35 seconds.

## 2. Scope of Work (SOW):

* U-Boot Optimization:
  + Streamline the bootloader process.
  + Reduce unnecessary initialization tasks.
  + Configure optimal boot parameters.
* Kernel Optimization:
  + Disable unwanted drivers and modules.
  + Reduce kernel debugging logs.
  + Optimize kernel configurations for faster boot.
* RootFS Optimization:
  + Create a minimal filesystem with only required libraries.
  + Remove unnecessary applications and services.
  + Ensure compatibility with the optimized kernel.
* Performance Validation:
  + Measure and validate boot time improvements.
  + Conduct stability and functionality tests.

## 3. Features & Functionalities:

* Faster system boot time within the target range of 30 to 35 seconds.
* Reduced memory footprint by minimizing kernel drivers and libraries.
* Efficient use of NAND storage with an optimized RootFS.
* Seamless user experience with no loss of essential functionality.

## 4. Phased Execution Plan:

| Phase | Description | Tasks | Deliverables | Timeline (Weeks) |
| --- | --- | --- | --- | --- |
| 1 | Analysis & Assessment | Evaluate current boot time and identify bottlenecks | Assessment Report | 1 |
| 2 | U-Boot Optimization | Optimize bootloader configurations | Updated U-Boot Binary | 2 |
| 3 | Kernel Optimization | Disable unwanted drivers and logs | Optimized Kernel Image | 3 |
| 4 | RootFS Minimization | Reduce filesystem size, keep essential libraries | Minimal RootFS | 2 |
| 5 | Validation & Testing | Measure boot time, validate stability and function | Test Report, Boot Time Results | 2 |

## 5. Assumptions & Dependencies:

* Customer provides existing U-Boot, Kernel, and RootFS sources.
* Access to the AM437x hardware for testing and validation.
* Boot time improvements depend on actual hardware configuration and limitations.
* No major hardware changes are required.

## 6. Next Steps:

* Review and approve the proposed timeline and scope.
* Begin assessment and analysis of the current boot process.
* Implement phased optimization tasks.
* Perform testing and validate results.
* Deliver final optimized boot configuration and documentation.

We are committed to achieving the targeted boot time reduction through systematic analysis and implementation. Please let us know if you have any questions or further requirements.