Ain Shams University
Faculty of Computer and Information Science
Computer System Department

Universal Secure Flash Over-The-Air Framework for Embedded and IoT Systems

Graduation Project Proposal

Table of Contents

1 Project Description

1.1 Problem Statement

In today's fast-paced business environment, there's a prevailing trend of prioritizing rapid development, leading to the swift creation of critical features within tight schedules. However, this accelerated pace often leaves little room for thorough testing and comprehensive security checks. Consequently, after a product's release, organizations commonly face two primary challenges: the need to add features initially discussed but not included, and the necessity to identify and rectify any post-deployment bugs or security vulnerabilities.

In this context, the challenge lies in the limitations of wired updates as a solution. This method typically relies on highly skilled personnel, often engineers, to oversee the upgrade process. Alternatively, it may require the recall and retrieval of all distributed products for in-house updates. Both of these approaches are not only time-intensive but also come with substantial costs.

A more efficient solution is to employ the Over-the-Air technique for device updates, eliminating the need for expert engineers or physical connections. However, the existing options are not fully prepared for use or may not be compatible with the specific hardware in use, necessitating additional development time and potentially raising security concerns.

1.2 Project Definition

A ready-to-deploy secure FOTA (Flash Over-The-Air) framework harmoniously integrates both backend distributed systems and the bootloader. It's designed for compatibility across a range of architectures and boards, making it ideal for embedded and IoT device systems.

The framework adheres to industry standards in bootloader and software update security, drawing on **u-boot** (Universal Boot Loader for Embedded boards supporting various processors like PowerPC, ARM, MIPS), **uptane** (a Secure Software Update Framework designed for the automotive sector), and **The Update Framework** (a system for securing software updates).

Within this framework, u-boot is extended to encompass OTA (Over-The-Air) update capabilities with a strong emphasis on security, addressing potential threats. It seamlessly integrates with the uptane secure distributed system structure, guaranteeing secure bootloader updates and safeguarding against tampering and eavesdropping.

Moreover, the framework caters to scenarios where the OEM (Original Equipment Manufacturer) manages the devices and ensures compliance with GDPR privacy regulations, which require user acceptance of incoming updates. This includes both OEM and end-user functionalities, facilitated through a web interface that monitors online device versions, allows bulk or individual firmware updates, and tracks user acceptance of updates.

1.3 Motivation

The rapidly evolving business landscape places immense pressure on organizations to deliver products and services quickly to stay competitive. While this focus on rapid development yields benefits in terms of feature deployment, it often leads to significant challenges post-release. Two primary concerns emerge:

1. **Feature Enhancement and Bug Resolution:** Swift development timelines can result in the exclusion of critical features initially planned, leaving organizations to address the need for feature enhancements post-deployment. Additionally, bugs and security vulnerabilities may surface after the product release, necessitating timely fixes.

2. **Overcoming Wired Update Limitations:** Traditional methods of software and firmware updates typically require expert personnel, like engineers, to oversee the upgrade process, or they may demand product recalls for in-house updates. These approaches not only consume time but also incur substantial costs.

The motivation for this project arises from the need for a more efficient and cost-effective solution to address these challenges. An Over-The-Air (OTA) update framework offers a promising alternative by eliminating the need for expert personnel and physical connections. However, current OTA options may not be fully prepared for use or compatible with specific hardware, necessitating further development to enhance their readiness and security.

This project aims to bridge these gaps, offering a ready-to-deploy and secure FOTA framework that seamlessly integrates with diverse hardware and software environments while adhering to industry standards in security and privacy regulations, such as GDPR.

1.4 Objective

The objective of the " Universal Secure Flash-Over-The-Air Framework for Embedded and IoT Systems" project is to achieve the following specific goals:

1. **Develop a Secure and Ready-to-Deploy FOTA Framework:** Create a robust Firmware Over-The-Air (FOTA) framework that ensures the security and integrity of software and firmware updates, making it ready for deployment in a variety of embedded and IoT device systems.

2. **Support Multiple Architectures and Boards:** Design the framework to be compatible with a broad range of architectures and boards commonly used in embedded systems, offering versatility to cater to diverse hardware requirements.

3. **Compliance with Industry Standards:** Adhere to established industry standards for bootloader and software update security. Incorporate best practices and security measures to protect against unauthorized tampering and data breaches.

4. **Seamless Integration with u-boot and uptane:** Extend the capabilities of u-boot, the Universal Boot Loader for Embedded boards, to include Over-The-Air (OTA) update functionality. Seamlessly integrate the framework with uptane, a secure software update framework designed for automobiles, to ensure the secure distribution of updates.

5. **User Acceptance and GDPR Compliance:** Develop a user-friendly web interface that allows both Original Equipment Manufacturers (OEMs) to manage and accept updates. Ensure compliance with GDPR privacy regulations, including user acceptance tracking for incoming updates.

6. **Efficient Device Management:** Enable device monitoring, bulk updates, and individual updates through the web interface, simplifying the management of device versions and firmware updates.

7. **Enhanced Security:** Focus on enhancing the security features of the framework to address potential threats and vulnerabilities associated with Over-The-Air updates, ensuring the integrity and confidentiality of updates during transmission.

8. **Documentation and User Guidance:** Create comprehensive documentation that encompasses codebase, configuration options, use cases, and deployment procedures to facilitate easy adoption and understanding of the framework.

By achieving these objectives, the project aims to provide a comprehensive and secure solution for device updates, mitigating the challenges associated with rapid development and ensuring the efficient management of software and firmware updates in the contemporary business landscape.

2 Project Team

In order to develop a framework with these capabilities, it is essential to assemble a proficient team consisting of members with expertise in embedded system development, network security, web development, penetration testing, and distributed systems knowledge. Our team possesses the required skill set, comprising the following members:

Name	Department	Email
Omar Hesham Elshopky	Computer System	20201700552@cis.asu.edu.eg
Ahmed Hossam Eltaher	Computer System	20201701048@cis.asu.edu.eg
Mahmoud Ahmed Nabil	Computer System	20201701177@cis.asu.edu.eg
Eslam Ashraf Ali	Computer System	20201701056@cis.asu.edu.eg
Enas Ahmed Abdelazem	Computer System	20201700169@cis.asu.edu.eg
Rana Anwar Ibrahim	Computer System	20201700267@cis.asu.edu.eg

(1st student is considered the team leader)

2.1 Supervision Team

Name	Department	Email
Dr. Karim Emara	Computer System	karim.emara@cis.asu.edu.eg
TA. Fatma El-wasify	Computer System	fatma.gamal@cis.asu.edu.eg

3 Scope and Work Plan

3.1 Main Modules/Functionalities

1. **Specifications Determination and Architecture Selection**
   Outline the system's architecture, specifications, and framework constraints and requirements.

2. **Threat Analysis and Threat Modeling**
   Evaluate the system from an attacker's perspective to pinpoint potential security threats and vulnerabilities, enabling effective risk mitigation from the outset of system development.

3. **Bootloader (based on u-boot)**
   - **3.1 Image Receiving Over Network**
     Establish a secure communication channel, validate connections from authorized servers, and receive new image blocks from the backend system within an isolated and separate process from the device's functional code.

   - **3.2 Failure Detection and Image Verification**
     Detect communication issues and ensure image authenticity in two methods, full and partial, base based on the device's energy consumption and computing power.

   - **3.3 System Image Update**
     Employ A/B update techniques to refresh system images and ensure mechanisms for recovery and rollback in case of update failures.

   - **3.4 User Accept Coming Updates**
     Provide an API to check for available updates and choose to initiate the update process.

   - **3.5 [Extra] Add Wireless Interface Card Support**
     Incorporate a versatile method to integrate Wireless Network Interface Cards (WNIC) into the core of uboot.

   - **3.6 [Extra] Master-Slaves Images Distribution**
     Distribute received images from the master MCU to surrounding MCUs (slaves) within the same device, guided by metadata from the backend system.

   - **3.7 [Extra] Extend U-Boot Configuration Menu GUI**
     Expand the configuration menu to include newly developed options, simplifying bootloader customization to meet device-specific requirements.

4. **Backend System (based on uptane)**

   4.1 Generalization and Integrability Changes
   Analyze the existing codebase of the uptane distributed system and implement necessary modifications to adapt it for broader usage, beyond its original focus on the automotive industry.

   4.2 Versions Management Web Interface
   Create a web interface for managing devices, tracking devices' firmware versions, and facilitating bulk or individual device updates.

   4.3 User Acceptance Management
   Develop a web interface and associated database to manage and monitor user acceptance of available updates, ensuring compliance with GDPR privacy regulations.

5. **Integration and Communication**
   Connect the distributed systems in the backend with the bootloader's API to facilitate smooth and secure communication between devices and the backend systems.

6. **Demos**
   Illustrate the framework's practical application, its various use cases, capabilities, and deployment procedures through a real-world prototype demonstration.

7. **Framework Testing**
   Execute a comprehensive testing plan covering the entire system to pinpoint functional, performance, stress, and security issues. It's important to note that unit testing occurs during the development phase, while this module assesses the framework as a whole.

8. **Documentation**
   Generate comprehensive documentation for the newly developed framework, encompassing the codebase, configuration choices, diverse use cases, and deployment instructions.

3.2 Estimated Timeline

The framework's development unfolds across five (5) phases, summarized as follows:

I. **Phase 1**
In this initial phase, the framework's specifications and constraints are established. The optimal architecture, informed by recent research and threat modeling results, is selected, and the codebase structure is prepared for further development.

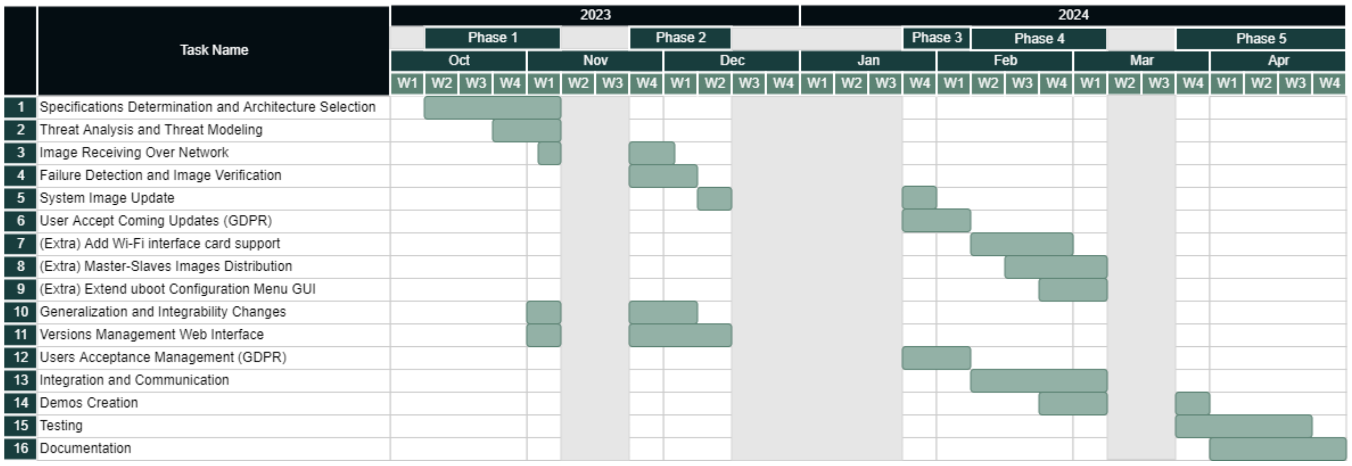
II. **Phase 2**
This stage focuses on the development of essential bootloader functionalities. Simultaneously, necessary modifications are made to uptane to prepare it for seamless interfacing and integration with the bootloader.

III. **Phase 3**
Phase 3 encompasses the development of features aligned with GDPR privacy regulations, including user acceptance of updates and the creation of web interfaces for this purpose.

IV. **Phase 4**
During this phase, integration between the backend systems and the bootloader is accomplished, along with the incorporation of additional functionality that can enhance the bootloader's capabilities.

V. **Phase 5**
The final phase involves comprehensive testing and the creation of detailed documentation for the framework.

The following is the initial Gantt chart that provides a precise schedule and work plan for each functionality within the framework.



(The gray areas represent the gaps period of midterm and final exams)

4 References

[1] Wolfgang Denk, DENX Software Engineering. “Universal Bootloader (u-boot) Documentation”

[2] Uptane Framework. “Secure Software Update Framework for Automobiles”.

[3] Nuvoton Technology Corporation. “OTA firmware update on u-boot”. (2021)

[4] Cebel, E., Donum, N., & Karacali, H. “Platform Independent Embedded Linux OTA Method”. *The European Journal of Research and Development*, *2*(4), 243–252. (2022)

[5] S. E. Jaouhari and E. Bouvet. “Toward a generic and secure bootloader for IoT device firmware OTA update”. International Conference on Information Networking (ICOIN), Jeju-si, Korea, pp. 90-95, doi: 10.1109/ICOIN53446.2022.9687242. (2022)

[6] Duca, Laurentiu-Cristian, Duca, Anton, Popescu, Cornel. “OTA Secure Update System for IoT Fleets”, International Journal of Advanced Networking and Applications, Vol. 13. (Dec 2021).

[7] Michael Opdenacker. “Implementing A/B System Updates with U-boot”. Embedded Linux Conference Europe. (2022).