Secure IoT Firmware For Cortex-M Processors

Sandro Pinto
Universidade do Minho
Guimarães, Portugal
sandro.pinto@dei.uminho.pt

Cesare Garlati
Hex Five Security
Redwood City, CA, USA
cesare.garlati@hex-five.com

***Abstract* — Developing secure IoT applications is challenging. Traditional Arm(v7) Cortex-M devices lack TrustZone like functionality for the safe execution of the many 3rd party components of the software stack. And the upgrade to new Cortex-M devices with TrustZone inexorably leads to a lengthy and expensive system redesign. In this paper, we introduce an alternative path to a TrustZone upgrade, based on an innovative hardware-enforced software-defined Trusted Execution Environment. We describe and detail all software components necessary to build a complete state-of-the-art secure IoT firmware for any Cortex-M device – with or without TrustZone. These include the MultiZone Trusted Execution Environment, TCP/IP connectivity, TLS/ECC cryptography, and MQTT client providing telemetry and OTA updates. All components are built on free and open standards, distributed under permissive licensing, and freely available for download from GitHub.**

Keywords — IoT, firmware, security, secure boot, firmware updates, cloud, trusted execution environment, TEE, TCP/IP, TLS, MQTT, OTA, lwIP, mbedTLS, FreeRTOS, open source, Armv7-M, Cortex-M, TrustZone.

I. Introduction

Building secure IoT firmware is challenging. State-of-the-art security features like secure boot, authenticated access to commercial cloud services, and over-the-air (OTA) firmware updates require a number of complex 3rd party software components [1,2]. These libraries are difficult to integrate, expose the system to increased attack surface, and inevitably lead to the dangerous execution of trusted and untrusted code in the same chip - where one single faulty instruction has the potential to compromise the integrity of the whole system - i.e., software vulnerabilities and backdoors [1,3].

Arm Cortex-M processors are widely used in general purpose microcontrollers (MCUs) and are embedded in System on Chip (SoC) devices that collectively ship in billions of units annually. However, widespread (Armv7-M) Cortex-M MCUs lack TrustZone like functionality required for the safe execution of untrusted applications [4,5]. And the upgrade to new (Armv8-M) Cortex-M devices that implement TrustZone is likely to result in lengthy and expensive system redesigns - with the granularity of one "secure world" still inadequate to provide separation for the multiple untrusted components of a modern IoT stack [5,6].

To meet the high-grade security requirements imposed by new IoT regulations, without forcing a complete system redesign, we propose the MultiZone Secure IoT Firmware as a quick and safe way to add high-grade security and separation to any IoT applications. The secure IoT firmware is based on the innovative MultiZone Trusted Execution Environment (TEE) [5] optimized for Cortex-M MCUs with memory protection unit (MPU). In this paper, we describe all software and hardware components required to build a reference application that securely controls a small robotic arm via an MQTT broker in the cloud. The reference application includes hardware drivers, the TEE, TCP/IP connectivity, TLS/ECC cryptography, and MQTT client providing real-time monitoring, device management, telemetry, and OTA applications deployment and remote firmware updates. The secure IoT firmware is built on free and open standards and its open source components are distributed under permissive licensing for any use including commercial.

II. Cortex-M Security Primitives

The Armv7-M architecture specifies a set of hardware primitives that enable the implementation of security-oriented architectures, e.g., TEE, in all Cortex-M MCUs with the exception of the tiny Cortex-M0.

A first group of hardware primitives available in Armv7-M CPUs is represented by the *privilege levels*. An Armv7-M MCU runs, at any point in time, at a specific execution mode and privilege level. The Armv7-M Architecture Reference Manual specifies Cortex-M processors can run in two modes: *Handler* and *Thread*. Handler mode is always privileged, while the Thread mode can have privileged and unprivileged access levels. The Handler mode executes exception handling code, and it is more privileged than the Privileged Thread mode. For instance, a few registers are just accessible in Handler mode (e.g., *IPSR*). The separation of privileged and unprivileged access levels allows for the development of reliable and robust systems. It provides a basic security mechanism by controlling memory accesses to specific regions.

Stack Component	Features	Size	License
IDE ▪ Renesas e^2 Studio 7.8.0 ▪ Hex Five's reference projects	▪ MultiZone IoT Firmware: MQTT, TLS, TCP/IP, RTOS, TEE, robot, terminal ▪ MultiZone SDK: TEE, USB Robot, uart terminal, bare metal buttons & leds ▪ MultiZone Blinky: TEE, uart terminal, bare-metal buttons & leds ▪ MultiZone Minimal: TEE, 4 zones available for user applications		Limited for use with Renesas products
BSP & Drivers library ▪ Renesas FSP 1.1.0 ▪ Hex Five's USB patch	▪ USB – optional, required for the robotic arm app ▪ UART – optional, required for the MultiZone terminal app ▪ Ethernet – optional, required for MQTT / TLS access to cloud services	120KB 32KB	Limited for use with Renesas products
TCP/IP library ▪ LWIP 2.1.1 ▪ Hex Five security patches	▪ IP, ICMP, UDP, TCP, ARP, DHCP, DNS, SNTP, MQTT ▪ Light weight single threaded execution ▪ Fully integrated with SSL stack	40KB 16KB	Modified BSD permissive commercial use ok
SSL library ▪ mbed TLS 2.23.0 ▪ Hex Five secure config	▪ TLSv1.2, Cipher TLS_AES_128_GCM_SHA256 ▪ ECC: prime256v1, Private Key NIST CURVE: P-256 ▪ Mutual authentication, Cert expiration verification, TLS large fragment	64KB 32KB	Apache 2.0 license permissive commercial use ok
Real Time OS (optional) ▪ FreeRTOS 10.3.0 ▪ Hex Five integration with TEE	▪ Secure unprivileged execution of kernel, tasks, and interrupt handlers ▪ No memory shared with TCP/IP and SSL library code ▪ No memory shared with other applications running in separate zones	32KB 16KB	MIT open source license permissive commercial use ok
Trusted Execution Environment ▪ MultiZone Security TEE 2.0 ▪ RA6M3 hw optimizations	▪ 4 separated Trusted Execution Environments (zones) enforced via MPU ▪ 8 memory-mapped resources per zone – i.e. ram, rom, i/o, uart, gpio, ... ▪ Secure inter-zone messaging – no shared memory, no buffers, no stack ▪ Protected user-mode interrupt handlers mapped to zones – up to 128	4KB 4KB	free for evaluation, commercial license priced per design – perpetual, no royalties

Fig. 1. Multi-zone Secure IoT Firmware for Cortex-M: summary of software components, features, size, and license

The *Memory Protection Unit (MPU)* is the second built-in security feature**.** It is optional but widely available in all Cortex-M0+/M3/M4/M7 processors. The MPU is fully programmable and can be configured to enforce permissions to specific memory regions according to privilege levels. This allows the partitioning of functionality between execution environments and the misuse of some particular resources - i.e. define RAM space as non-executable (eXecute Never, XN) to limit buffer misuse and prevent code injection attacks.

III. MULTIZONE SECURE IOT FIRMWARE FOR CORTEX-M

The MultiZone Secure IoT Firmware for Cortex-M includes the following software components: MultiZone TEE, lwIP TCP/IP stack, mbed TLS crypto library, FreeRTOS operating system (optional), MQTT gateway, and three bare metal applications demonstrating secure access to hardware drivers. Fig. 1 presents a summary of the overall software components, features, size, and license. The MultiZone Secure IoT Firmware is designed to work with any Cortex-M processors with MPU and a few tens of kilobytes of program memory (~100KB). These include most Cortex-M0+, Cortex-M3, Cortex-M4, Cortex-M7, Cortex-M23, and Cortex-M33 MCUs and SoC. The architecture is modular and can be easily configured to add or remove individual components to fit a broad range of commercial IoT applications.

MultiZone TEE. The MultiZone Security Trusted Execution Environment [5] guarantees hardware-enforced software-defined separation of multiple functional areas within the same chip. MultiZone is completely self-contained, exposes an extremely small attack surface, and is policy-driven, meaning that no coding or security expertise are required. MultiZone main components include: (i) the TEE Configurator development utility that extends the GNU toolchain, (ii) the TEE Runtime, a small binary providing secure boot, separation kernel, and secure communications; and (iii) the TEE API, a free and open API providing static wrappers for system calls. The MultiZone IoT Firmware for Cortex-M includes MultiZone Security TEE 2.0 for Arm, 4 separated Trusted Execution Environments (zones) enforced via MPU, 8 memory-mapped resources per zone to protect programs, data, and access to peripherals, secure inter-zone messaging with no shared memory structures like buffers, hype, or stack, and protected user-mode interrupt service routines mapped to zones [7] - up to 128. Fig. 2 illustrates the overall system architecture for the MultiZone Security TEE for Cortex-M.

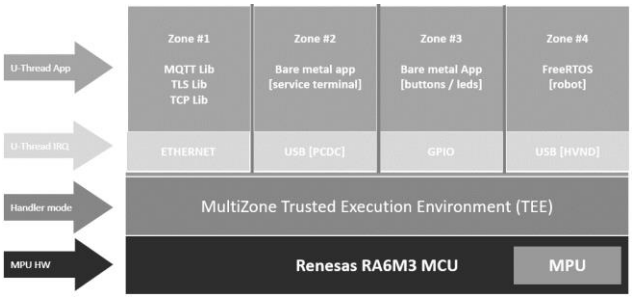

Fig. 2. Multi-zone Security for Cortex-M: system architecture

TCP/IP Library. The lightweight IP (lwIP) library is a permissive open-source implementation of the TCP/IP protocol stack. lwIP is designed to minimize code footprint making it suitable for embedded systems with limited resources. The MultiZone IoT Firmware includes a modified version of lwIP 2.1.1 optimized for security and performance. In particular, lwIP is tuned for single-threaded execution, avoiding the overhead and the security risks typically associated with multi-tasking operating systems – FreeRTOS is optional and included to facilitate the security upgrade of legacy applications. The connectivity stack is configured to provide IP, ICMP, UDP, TCP, ARP, DHCP, DNS, SNTP, and MQTT protocol support, although additional protocols and services can be enabled if necessary, i.e., IPV6, PPP, HTTP server. The connectivity stack is fully integrated with the cryptography library to provide advanced SSL functionality, including mutual authentication and TLS large packet fragmentation.

TLS Library. Mbed TLS is a permissive open-source library that implements cryptographic primitives, X.509 certificate manipulation, and the SSL/TLS and DTLS protocols. The MultiZone IoT Firmware integrates an optimized version of mbed TLS 2.23.0. The library is configured to provide TLS 1.2 connectivity optimized for Cortex-M devices. The default cipher suite provides the state-of-the-art security required to connect to commercial IoT clouds like AWS, Azure, and similar: Advanced Encryption Standard with 128bit key in Galois/Counter mode (AES 128 GCM) and Secure Hash Algorithm 256 (SHA256). Keys are generated based on the 256-bits NIST curve of the Elliptic Curve Cryptography (ECC) implementation. The standard configuration supports client authentication, server name authentication, certificate expiration verification, TLS large fragments, and high-grade entropy.

MQTT Gateway. The MultiZone IoT Firmware zone #1 implements a single-threaded bare-metal MQTT client in the form of a gateway. The gateway is responsible for forwarding MQTT messages (including binary files) to and from the MQTT broker. The gateway is written in C and it is fully configurable. Internal messages sent to zone #1 are forwarded to the broker topic mapped to the device-id and source zone. External messages sent via the broker to the topic device-id/zone are forwarded internally to the respective zone.

Individual applications deployment can be triggered manually via authenticated MQTT posts or automatically upon device startup via MQTT persistent messages. This allows to ship devices with a minimum software image and automatically provision the most up-to-date version of the application software upon initial connection to the broker.

Real-Time Operating System. FreeRTOS is an open-source real-time operating system (RTOS) for microcontrollers and small microprocessors. FreeRTOS includes a multi-tasking kernel and an extensive number of software libraries suitable for use across industry sectors and applications. Thanks to the built-in trap & emulation engine, the MultiZone IoT Firmware can run one or more instances of unmodified FreeRTOS binaries and relative tasks (kernel 10.4.1). Kernel, tasks, and interrupts run in protected unprivileged U-mode. FreeRTOS is included in the firmware to facilitate security upgrades of existing legacy applications. It is not a requirement as the MultiZone TEE provides its on safety-critical preemptive scheduler – se for example the alternative implementation of the 3-task real-time controller of the robotic arm which is provided as a FreeRTOS application (zone 4.2) or MultiZone TEE application (zone 4.1).

IV. Use Cases

The MultiZone Secure IoT Firmware is suitable for a wide range of use cases. In the following, we describe three of the most common IoT applications of the integrated TEE/TLS stack.

Remote firmware updates. Firmware updates are one of the most important features to take into account while developing a secure IoT device. In particular, in some countries, there is already strict legislation that dictates remote firmware updates (OTA) as mandatory. For IoT device makers concerned about time, cost, and security risks of developing a DIY solution, MultiZone Secure IoT Firmware provides high-grade security OTA updates via open standard MQTT and TLS protocols. The OTA update mechanism is based on open standards to avoid the vendor lock-in typical of commercial cloud providers – and in fact suitable to inexpensive private clouds based on open source infrastructure.

Real-time monitoring and device management. IoT devices connect to the network to provide information they gather from the environment through sensors, or to allow other systems to reach out and act on the physical world through actuators. MultiZone Secure IoT Firmware provides secure bidirectional access to/from the device via standard MQTT protocol and enables telemetry, real-time monitoring, and device management. To mitigate the recurring cost of commercial web services, the firmware serves a broader audience by working with public and private clouds, i.e., OEM owned PKI and open-source backend infrastructure.

Secure access to commercial IoT clouds. IoT devices generate a vast amount of data. The Cloud, as part of the IoT ecosystem, manages the flow, process, analysis, and storage of these data. Thus, secure access to commercial IoT clouds is critical for several IoT service providers and device makers, in particular those concerned about backdoors and the lack of separation of consolidated 3rd party software. State-of-the-art technology relies on lightweight communication protocols (i.e., MQTT) atop secure and encrypted communications. MultiZone Secure IoT Firmware provides built-in secure connectivity (TLS/ECC, mutual authentication) to the commercial cloud provides like AWS and Azure, as well as secure and separate execution of 3rd party components (via separated hardware-enforced execution environments).

V. Reference Application and Evaluation

Fig. 3 depicts the MultiZone Secure IoT Firmware Reference Application (RA) for the Arm Cortex-M MCUs. The RA includes all software and hardware components necessary to build a complete IoT application that securely controls a robotic arm via a standard MQTT broker in the cloud. The robotic arm is optional. *Zone 1* connects to a private or commercial IoT cloud via Ethernet or wirelessly if a Wi-Fi router is connected to the Ethernet port. Operating in zone 1 is the fully integrated, single-threaded, secure network stack providing TCP/IP, TLS, and MQTT. *Zone 2* connects to a local host via USB Peripheral Communications Device Class Driver (PCDC). Operating in zone 2 is a bare metal ANSI terminal application written in C. It presents the user with a command-line interface to send and receive MQTT messages, to assess the enforcement of the separation policies, and to measure performance overhead. Zone 3 blinks an LED and interfaces two additional LEDs using local buttons to demonstrate real time multi-tasking, protected interrupt handling, and secure messaging. It has two interrupts mapped to button S1 and S2 that toggle LED2 and LED3 and send a notification message to both zone 1 (forwarded to the broker) and zone2. In addition, zone3 implements a few message responders to verify separation policies, non-interference, and preemptive execution. Zone 4 operates the optional robot using the USB Hardware Vendor Class Driver (HVND). Robot commands are received from zone 1 (remote broker) or zone 2 (local terminal) and the status of the robot reported back via secure messaging and published to the broker. Multi-tasking is provided by the bare metal application itself or by an optional RTOS.

A. Evaluation

The MultiZone Secure IoT Firmware for Arm Cortex-M was evaluated on a Renesas EK-RA6M3 board. The EK-RA6M3 is equipped with a Cortex-M4 R7FA6M3AH3CFC processor running at 120MHz. In addition to extensive tests for security, separation, and reliability, we assessed firmware size and connectivity round-trip time (RTT).

MultiZone Secure IoT Firmware Size. By design, the MultiZone Secure IoT Firmware for Cortex-M is extremely optimized for resource-constrained devices. The default configuration requires approximately 100 KiB of FLASH and 32KiB of RAM, enabling the deployment of connected solutions in devices with few KiB of memory.

Round-Trip Time. To assess the round-trip time, we ran the *ping* command from a remote terminal. For the system under

test, configured for four zones, the ping statistics have reported an average RTT of 1.20 milliseconds

REFERENCES

[1] Chris Conlon and Cesare Garlati. “A New Zero-Trust Model for Securing Embedded Systems”. In Proceedings of the Embedded World Conference, Nuremberg, Germany, 2019.

[2] N. Asokan, T. Nyman, N. Rattanavipanon, A. Sadeghi, and G. Tsudik, "ASSURED: Architecture for Secure Software Update of Realistic Embedded Devices," in IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, 11, pp. 2290-2300, Nov. 2018.

[3] O. Alrawi, C. Lever, M. Antonakakis and F. Monrose, "SoK: Security Evaluation of Home-Based IoT Deployments." IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2019.

[4] D. Kwon, J. Shin, G. Kim, B. Lee, Y. Cho, and Y. Paek, “uXOM: Efficient execute-only memory on ARM Cortex-M,” in 28th USENIX Security Symposium (USENIX Security 19), USENIX Association, pp. 231–247, 2019.

[5] Sandro Pinto and Cesare Garlati. “MultiZone Security for Arm Cortex-M Devices”. In Proceedings of the Embedded World Conference, Nuremberg, Germany, 2020.

[6] David Cerdeira, Nuno Santos, Pedro Fonseca, Sandro Pinto. " SoK: Understanding the Prevailing Security Vulnerabilities in TrustZone-assisted TEE Systems." IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2020.

[7] Sandro Pinto and Cesare Garlati. “User Mode Interrupts: A Must for Securing Embedded Systems”. In Proceedings of the Embedded World Conference, Nuremberg, Germany, 2019.