# A Proposal for Security Assessment of Trustzone-M based Software

Antonio Ken Iannillo, Radu State
*Interdisciplinary Centre for Security, Reliability and Trust (SnT)*
*University of Luxembourg*, Luxembourg, Luxembourg
{antonioken.iannillo, radu.state}@uni.lu

***Abstract*—With the advent of the Internet of Things (IoT) paradigm, computing and networking capabilities are extending to devices that are not considered as computers, enabling them to interact with the physical world or other software entities with minimal or no human input. This fast abstract proposes a methodology for the security assessment of software based on TrustZone-M, the ARM hardware security extension for microcontrollers. The methodology consists of the exploitation of a verification and validation framework to automatically test TrustZone-M based software.**

***Index Terms*—IoT, security, assessment, trustzone-m**

## I. Introduction

Trusted hardware technologies are commonly used as anti-tamper technologies to make software more resistant against attack and protect critical program elements. It is generally more difficult successfully attack trusted hardware than a software-only protection scheme. In these years, several technologies have been proposed and implemented in computers processors. The most spread solutions are Intel SGX[1] and ARM Trustzone-A [1] [2]. They both have been largely used for implementing the security and privacy of software running in both the cloud servers and the mobile devices. With the advent of the Internet of Things (IoT) paradigm, computing and networking capabilities are extending to devices that are not considered as computers, enabling them to interact with the physical world or other software entities with minimal or no human input. Every IoT device can potentially talk to other related devices in an environment to automate home and industry tasks, and to communicate usable sensor data. Furthermore, they can be found essentially everywhere.

These devices are powered by embedded computers: small hardware (microcontrollers) equipped with specialized sensors and actuators that run a constrained software to handle data and external communication. Microcontrollers processors have much more limitations than application processors. Indeed, the main requirements for microcontroller applications are low power consumption, real-time processing, deterministic behavior, and low interrupt latency. Thus, hardware security extensions for application processors cannot be directly applied, because they have been developed for more relaxed use cases. However, we strongly require IoT devices to be secure.

Research supported by EC H2020 Project CONCORDIA GA 830927

[1]https://software.intel.com/en-us/sgx/

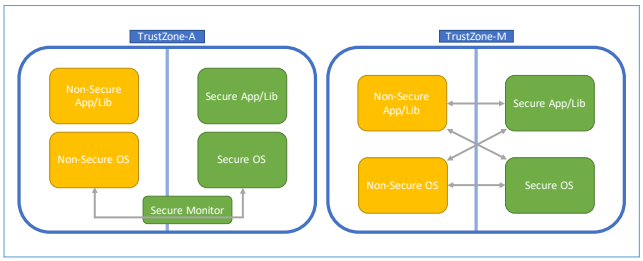


Fig. 1. TrustZone technologies

Lately, ARM Holding, that already owns the largest share of mobile and embedded markets (60%), has further extended TrustZone-support for the tiniest low-end devices, which it estimates to reach nearly 1 trillion by 2035 [3]. To reach this objective, ARM designed an hardware security extension from the ground up, instead of reusing it from application processors, for microcontrollers with the name of TrustZone Technology for Cortex-M profile or TrustZone-M [4]. The first and only architecture using it so far is the ARMv8-M architecture.

There are several security applications that can be enabled by TrustZone-M, for example: IoT device makers can use it to store intellectual property in secure memory while still allowing non-secure application to access it via APIs; secure storage of critical information; a root of trust implementation provides a secure foundation for over-the-air (OTA) firmware updates and mutual authentication between devices in a system.

As anticipated, these two technologies are very different. In both TrustZone-A and TrustZone-M, the process has the ability to execute either in the secure or the non-secure state, whereas the non-secure software cannot access secure resources directly. On the other hand, TrustZone-M implements the two worlds in a completely different way. The secure state is not determined by the value of a processor bit, but the separation is memory map-based, i.e. the processor is in the secure state when executing code from the secure memory regions. Furthermore, communication between worlds are not handled by a secure monitor that resides in both worlds, but the transitions take place automatically in exception handling code allowing multiple entry points to the secure world. Fig. 1 shows the differences of these two technologies in the way

secure and non-secure worlds communicate.

We aim to a methodology for the security assessment of software based on TrustZone-M technology and a novel verification and validation framework to implement this methodology. The methodology consists of new methods for the security assessment: the detection of the attack surface of the software running in the secure world of an ARMv8-M device; the generation of test inputs for the target interfaces exploiting the unique feedback of the target; the detection of security violations in executed test case (TrustZone-M test oracles). These methods will be implemented in a novel verification and validation framework specifically tailored for software based on trusted hardware technology for microcontrollers.

## II. Related Work

Few studies on the security properties on TrustZone-based systems uncovered notable vulnerabilities in TrustZone-A [5] [6]. Reasonably, the infancy of TrustZone-M causes the scarcity of available information. The main focus is on security by design, e.g. Kinibi-M [2], CFI CaRE [7], ASSURED [8].

ARM Holding is investing strongly on low-end secure devices, not only in term of hardware architecture design. Indeed, it recently started the Platform Security Architecture (PSA) [3] and an accompanying open source software project, named Trusted Firmware-M[4].

## III. Research Questions and Proposed Approach

We want to answer the following research questions:

RQ1 In a software system based on TrustZone-M technology, is it possible to automatically detect the entry points of the secure world from the normal world?

RQ2 What classes of vulnerabilities affect secure code exploiting TrustZone Technology in ARMv8-M architectures?

RQ3 If the proposed framework is applied against a deployed software implementation based on TrustZone-M technology, is it able to find vulnerabilities (efficacy) with a reasonable time (efficiency)?

We identified different steps to answer these three research questions.

**Discovery Phase** First, we need to develop a new method for the detection of the entry points of the secure world from the normal world in secure software implementations based on TrustZone-M. We will start by studying the ARMv8-M specification and deeply understand what the hardware mechanisms for the communication between the secure and non-secure worlds are. Then, we extract the unique features of the entry points and build a detection tool based on them. The entry points of the secure code constitute its attack surface. Attack surfaces detection is an important step to security assessment, because it unveils where the adversary can attack the target from. The approach we propose is based on a systematic analysis of the target binary code. For each detected entry function, we then analyze the first instructions and extract its signatures. Furthermore, we craft an input sequence and test the entry point to be accessible from the normal world. This approach can be completely automatized, and it is complementary to the analysis of the source code and documentation, but most of the time the attacker has no access to them because it is proprietary or absent. The results of this phase will automatically answer RQ1.

**Test Design Phase** Then, we develop a framework for the verification of secure software implementations based on TrustZone-M. We exploit the knowledge obtained by the detection of entry points and we test them through fuzzing techniques, exploiting and extending the state-of-the-art. We define the operators and the feedbacks to construct the test case (input generation) and the oracles to define test failures (vulnerability detection). The input generation benefits from the debugging utilities provided by the architectures and creates inputs based on code constants and semantic knowledge, both extracted from the target. The vulnerability detection is based on the monitoring of the security properties guaranteed by the secure function. For example, the oracle check that a secure memory region is not accessed by the running test, instead of looking for specific vulnerabilities or determining the correct output for a every generated given input. The attack surface detector is added to the framework to help the user to initialize the testing campaign.

**Analysis Phase** Finally, we can perform an experimental campaign on secure software implementation based on TrustZone-M and analyze the results. We execute a testing campaign on Trusted Firmware-M and measure the efficacy (number of vulnerabilities found) and efficiency (testing time) of the framework. The found vulnerabilities are classified by their nature and severity. The impact on the embedded device will be deeply analyzed from the root cause to the failure to provide useful information for the failure-tolerant mechanisms. Once detected, a vulnerability will be also communicated to the Common Vulnerability and Exposure (CVE) repository [5].

[2]https://www.trustonic.com/news/blog/not-just-droning-rise-kinibi-m/

[3]https://www.arm.com/why-arm/architecture/platform-security-architecture/

[4]https://www.trustedfirmware.org/about

[5]https://cve.mitre.org

## References


[1] T. Alves, "Trustzone: Integrated hardware and software security," *White paper*, 2004.

[2] A. ARM, "Security technology building a secure system using trustzone technology (white paper)," *ARM Limited*, 2009.

[3] P. Sparks, "The route to a trillion devices," *White Paper, ARM*, 2017.

[4] *TrustZone technology for ARMv8-M Atchitecture*, ARM, 2017.

[5] A. Machiry, E. Gustafson, C. Spensky, C. Salls, N. Stephens, R. Wang, A. Bianchi, Y. R. Choe, C. Kruegel, and G. Vigna, "BOOMERANG: Exploiting the Semantic Gap in Trusted Execution Environments," in *Proceedings 2017 Network and Distributed System Security Symposium*, 2017.

[6] D. Rosenberg, "QSEE TrustZone Kernel Integer Overflow Vulnerability," Tech. Rep., 2014.

[7] T. Nyman, J.-E. Ekberg, L. Davi, and N. Asokan, "Cfi care: Hardware-supported call and return enforcement for commercial microcontrollers," in *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 2017, pp. 259–284.

[8] N. Asokan, T. Nyman, N. Rattanavipanon, A.-R. Sadeghi, and G. Tsudik, "Assured: Architecture for secure software update of realistic embedded devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2290–2300, 2018.