1 Demystifying Arm TrustZone: A Comprehensive Survey

1.1 Introduction

1.1.1 Beginning

Manufacturers of TrustZone-enabled SoCs (System on Chip) were somewhat reluctant to disclose technical details or any architectural-related details. Research involving TrustZone has been further slowed down by the limited availability of development platforms in which all of TrustZone's capabilities were unlocked. For instance, certain boards were natively programmed to boot the processor directly into the normal world, thereby preventing system developers from deploying code inside the secure world.

1.1.2 Lately

A dynamic open source community has matured, working toward the definition of common API specifications to promote interoperability across TrustZone-based solutions. The research community, in particular, has been extremely active in exploring new ways to leverage TrustZone as a key-enabler technology for enforcing Trusted Execution Environments (TEEs) and hardware-assisted virtualization. TrustZone can provide fundamental primitives for securing sensitive data while benefiting from its wide deployment across a large number of mobile and low-end devices.

1.2 TrustZone for Application Processors

1.2.1 Overview

TrustZone for application processors refers to the hardware-based security built into SoCs to provide a foundation for improved system security for Cortex-A processors. The most important architectural change at the processor level consists in the introduction of two protection domains designated by the name of worlds: the secure world and the normal world. The world where the processor currently executes is determined by the value of a new 33rd processor bit, also known as the Non-Secure (NS) bit. The value of this bit can be read from the Secure Configuration Register (SCR) register, and it is propagated throughout the system down to the memory and peripheral buses.

1.2.2 Hardware Isolation

To reinforce hardware isolation between worlds, the processor has banked versions of the special registers, as well as some system registers. In the normal world, the security-critical system registers and processor core bits are either totally hidden or conditioned by a set of access permissions supervised by the secure world software.

1.2.3 Memory

The memory infrastructure has also been extended with TrustZone security features, in particular with the introduction of the TrustZone Address Space Controller (TZASC) and the TrustZone Memory Adapter (TZMA). The TZASC can be used to configure specific memory regions as secure or non-secure, such that applications running in the secure world can access memory regions associated with the normal world, but not the other way around. A similar memory partitioning functionality is implemented by the TZMA, but targeting off-chip ROM or SRAM.

1.2.4 Interrupts

TrustZone technology allows for system devices to be restricted to secure or normal worlds. This is achieved with the introduction of a TrustZone Protection Controller (TZPC). The TrustZone architecture extends the Generic Interrupt Controller (GIC) with support for prioritized secure and non-secure sources. Interrupt prioritization is important to prevent denial-of-service (DoS) attacks by non-secure software, since it enables secure interrupts to be handled with higher priority than the non-secure interrupts.

1.2.5 TrustZone for Microcontrollers

In microcontroller applications, low power consumption, real-time processing, deterministic behavior and low interrupt latency are mainstream requirements, which lead TrustZone for Armv8-M to be designed from the ground up instead of being reused from Cortex-A processors.

1.3 Trusted Execution Environments

1.3.1 TEE-Enabling Technology

Modern computer systems tend to depend on large trusted computing bases (TCBs). Typically, a TCB comprises complex software such as the OS kernel, privileged services, and libraries. Systems featuring a bloated TCB tend to be more vulnerable to attacks than small-sized TCB systems. To address the TCB bloating problem, TrustZone has appeared as a fundamental hardware mechanism that enables to provide a TEE in which critical applications can execute securely featuring a TCB several orders of magnitude smaller than the rich OS. More specifically, a TEE consists of an isolated environment in which trusted applications can execute without the interference of the local (untrusted) OS.

1.3.2 TEE Kernel & Service

Essentially, the secure world provides a restricted execution environment where the TEE can reside. Whenever the system boots, the processor enters the secure world to give to any privileged firmware the chance to set up its internal data structures, configure the interrupt controller of the entire system, and set up protections for secure memory regions and peripherals. Upon the completion of these operations, the processor switches worlds and yields control to the bootloader of the rich OS. Depending on the trusted program that runs in the secure world, we distinguish two types of TEE architectures: TEE kernel or TEE service. The trusted kernel is responsible for: managing memory of the secure word, enforcing memory protection for each TEE, handling communication between TEE and the OS, and providing an API to TEE applications. However, TEE services implement a specific function and do not require any low-level OS logic to manage their own memory and cross-world communication. To prevent mutual interference, only one TEE service can be deployed on the device. This is a disadvantage when compared to TEE kernels, which allow multiple applications to run in independent TEE instances. However, a downside of TEE kernels is that they normally depend on larger TCBs when compared to systems where a single TEE service is deployed.

1.3.3 Trusted Kernel

On TrustZone-enabled platforms, the runtime support for sustaining the lifecycle of such applications is typically provided by a privileged trusted kernel, which runs in the secure world. The communication between the rich OS and the trusted kernel requires context switch between worlds. Given that rich OS and trusted kernels are not necessarily developed by the same manufacturer and nevertheless need to interoperate with each other, a lot of TEE standardization efforts have been made.

1.3.4 Trusted Service

An important class of trusted service solutions aims to provide secure storage and access to sensitive files in the presence of a potentially compromised local OS. Another relevant category of trusted services aims to provide secure authentication and cryptographic functions. Among the most challenging requirements of building trusted services, we find the need to provide secure I/O channels to the user interface. The difficulty lies in that the UI is supported by device drivers of the rich OS, which are both untrusted and difficult to implement with a small code footprint. To address this problem, instead of implementing the required drivers from scratch, some systems allow the secure world domain to reuse untrusted drivers implemented inside the rich OS.

1.3.5 TEE Systems for the Cloud

While some of these solutions are designed to operate on a standalone basis, other systems have been conceived to be tightly coupled with a cloud backend. By relying on the TEE, an attacker that manages to compromise the local OS will not be able to recover the keys and the content of the files. Before provisioning the keys into the TEE, it allows the cloud endpoint to remotely attest the client's software thereby ensuring that keys are properly allocated into the TEE rather than to the untrusted domain controlled by the client's OS. Beyond relying on client-side TEE, researchers have proposed new applications of TrustZone-assisted TEE on the cloud backend itself.

1.3.6 Userspace TEE

One class of TEE hardware allows for securing user space (ring 3) programs without the need to trust in privileged OS code running at ring 0 or below. Intel SGX allows for the creation of memory regions named enclaves, which are protected from hardware and software access. Most notably, SGX implements hardware-enabled memory encryption.

1.3.7 OS TEE

Another class of TEE hardware aims to implement secure execution environments at the OS level (ring 0). In AEGIS, part of the OS is split and runs inside a protected environment established by the processor.

1.3.8 Hypervisor TEE

Whenever a TEE hardware provides mechanisms to instantiate a TEE stack based on a trusted hypervisor, we say that it operates at ring -1. An example is Bastion, a security architecture that relies on both a modified processor and a trusted hypervisor to provide confidentiality and integrity protection for security-sensitive software.

1.3.9 Processor TEE

Certain TEE hardware technologies implemented by the processor can operate below the hypervisor level in ring -2. Arm TrustZone technology, can be highlighted as one of its most representative examples. In fact, the virtualization extensions to the Armv8 architecture allow for the deployment of an untrusted hypervisor in the normal world. Enabled by TrustZone, an independent trusted TEE stack can then reside inside the secure world.

1.3.10 Coprocessor TEE

Last, we mention a class of TEE hardware that relies on independent coprocessors; hence, we say they allow for the implementation of ring -3 TEE software stacks. The most prevalent of such technologies is the Trusted

Platform Module (TPM). The TPM consists of a coprocessor, which is typically located on the motherboard. Its primary purpose is to serve for bootstrapping trust on the local platform: it is responsible for storing the software measurements computed during the trusted boot process of the system, and for securely storing cryptographic keys for remote attestation and data sealing operations.

1.3.11 Discussion

Although the reduction of TCB can help eliminate the presence of potential code vulnerabilities, that, by itself, cannot ensure its correctness. The latest efforts to overcome this challenge have leveraged software verification techniques to formally prove the correctness of privileged code residing within the secure world.

1.4 Trustzone-Assisted Virtualization

1.4.1 Overview

TrustZone technology, although implemented for security purposes, enables a specialized, hardware-assisted, form of system virtualization. With a virtual hardware support for dual world execution, as well as other TrustZone features like memory segmentation, it is possible to provide time and spatial isolation between execution environments. Basically, the non-secure software runs inside a VM whose resources are completely managed and controlled by a hypervisor running in the secure world. TrustZone-assisted virtualization is not particularly considered full-virtualization neither paravirtualization, because, although guest OSes can run without modifications on the non-secure world side, they need to co-operate regarding the memory map and address space they are using.

1.4.2 Single Guest

The single-guest configuration is the simpler system architecture of a TrustZone-assisted virtualization solution. The guest OS executes under the non-secure perimeter, and the hypervisor runs in the monitor mode. The hypervisor has a privileged view of the entire system, while the guest OS has limited access to system resources. The TCB of the system is confined to the code running on the secure world side, which means it just depends on the hypervisor size. Memory, devices, and interrupts assigned to the guest OS are configured as non-secure resources and they are directly managed by the guest OS, while the remaining secure resources are under strict supervision of the hypervisor. The guest OS manages its own MMU and cache lines.

1.5 Security Issues and Vulnerabilities

1.5.1 Overview

TrustZone provides several security primitives that developers can leverage to implement trust- worthy systems. A simplified but realistic multi-core prototype of the Arm TrustZone technology has been verified and proved to be secure from a hardware standpoint [30]; however, the poor us- age of TEEs coupled with some microarchitectural misconceptions have opened several security issues and vulnerabilities. While the former is a consequence of the lack of robust TEE runtime im- plementations, which results in failing to provide secure containers to applications, the latter is a direct consequence of architectural decisions or the existence of implementation-defined parts on TrustZone specification. Examples of specific microarchitectural attack vectors encompass hard- ware exceptions (SMC, IRQ, FIQ), caches, and power management modules.

1.5.2 TEE Vulnerabilities

although the security design of TEEs might be correct, i.e. secure architecture and perfect and robust isolation, the code running inside the TEE may contain vulnerabilities that can be exploited by attackers to corrupt the TEE and compromise the trust state of the entire system. Like Ning et al. [72], we agree that the current state of the art TEE research still lacks frameworks to verify and/or analyze the secure code, properly defense mechanisms within the trusted environments, methods for monitoring and detecting compromised TEEs, and resilient plans to recover and rejuvenate from attacks.

1.5.3 Hardware Vulnerabilities

A number of hardware-related vulnerabilities has also been uncovered over the last few years. The reported vulnerabilities affect different hardware parts of the platform, in particular, the compo- nents that constitute the platform's root of trust, caches, power management mechanisms, and FPGAs. Root of trust. While Intel and AMD specify the TPM as the root of trust for their systems, Trust- Zone, per se, does not specify where keys for authentication and decryption shall be stored.

1.6 Future Directions

1.6.1 IoT Devices

The problem is that securing IoT devices can be a quandary, with hardware requirements and cost limitations pushing different design directions. To address this problem, Arm decided to span TrustZone to the new generation of microcontrollers, by making security practical at scale and across the entire value chain. With TrustZone built-in on the tiniest of things, Arm is easing the economics of security, reducing risk, cost, and the complexity of implementing robust security measures. While virtualization in embedded systems started by primarily being deployed on high-end devices, the increasing adoption of virtualization technology also starts finding some applicability on low-end hardware, but with several performance limitations due to the lack of hardware support on such devices. To fill this gap, Arm has recently included virtualization extensions in the new generation Cortex-R processors. The Cortex-R52 is the first processor from the Cortex-R family introducing hardware support for virtualization. Hardware virtualization support on real-time processor series is slightly different from the one existing on application processors, due to the need of copying with hard real-time capabilities.