Using SiFive WorldGuard for Deploying a TEE/REE System

Version 1.3

Using SiFive WorldGuard for Deploying a TEE/REE System

Proprietary Notice

Release Information

Version	Date	Changes
1.3	January 20, 2021	• Updates to diagrams, terms
1.2	December 2, 2020	• Updates to wording
1.1	September 25, 2020	• Updates to format, wording
1.0	July 27, 2020	• Initial release

Contents

1

Using SiFive WorldGuard for Deploying a TEE/REE System

1.1 Introduction

In this article, we present how the SiFive® WorldGuard solution can be used as an alternative to Arm® TrustZone® technology to isolate two software domains using the same core. More precisely, we consider using the SiFive WorldGuard technology to enable a Trusted Execution Environment (TEE; also known as the "Secure World") coexisting with a Rich Execution Environment (REE; also known as the "Non-secure World") on the same core.

1.2 TEE and REE Definitions

From [2], "A TEE is an isolated execution environment that provides security features such as isolated execution, integrity of applications executing with the TEE and confidentiality of their assets." A TEE guarantees the authenticity of the executed code, the integrity of the runtime state, and the confidentiality of its code, data and runtime states.

Untrusted applications (UAs) run within the REE on top of the Rich OS (Linux, freeRTOS), while trusted applications (TAs) run within the TEE on top of the Trusted Kernel. Trusted applications are able to fully access the core, memories, and peripherals, therefore the TEE usually hosts security-oriented applications and services such as secret keys handling, DRM, biometric processing, and payment processing.

Conversely, hardware isolation mechanisms prevent untrusted applications from accessing sensitive resources.

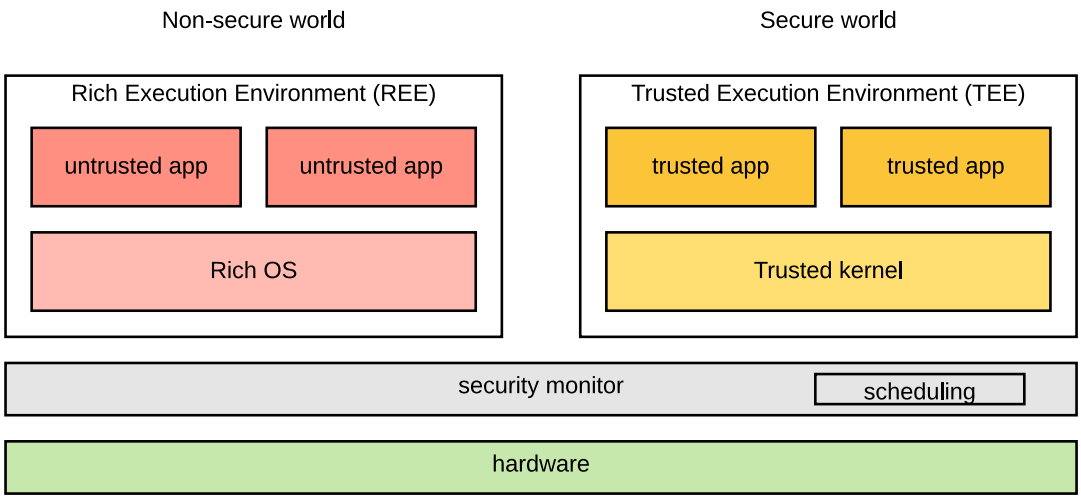

Figure 1: *Generic TEE/REE Architecture*

Additional services make the functioning secure and reliable. Secure scheduling assures the efficient coordination between the TEE and the REE while inter-environment communication (IEC) defines a communication interfaces between the TEE and the REE.

1.3 Requirements and Constraints

We have the usual hierarchy requirements between the OS and applications to prevent applications from polluting the OS functioning.

Beyond that, such systems have to guarantee a strong isolation between the two worlds and protect the integrity and confidentiality of the assets handled by the trusted apps. A strict control on the resources (peripherals and memories) shall be enforced through a strict hardware partitioning between the two worlds. Furthermore, a strong isolation between the trusted apps themselves is also required. Finally, the efficiency of those mechanisms is critical, as the systems are often real-time (e.g., interrupts management and contexts switching).

Last, but not least, a Root of Trust is mandatory to guarantee that any mechanism satisfying the above requirements is correctly set up. This Root of Trust includes a secure boot mechanism, attestation services, and keys management.

> **Note**
>
> The hypervisor layer is not depicted here for sake of simplicity.

1.4 Arm Solution

Such TEE/REE systems are frequently implemented on Arm-based chips. Simple Arm-based implementations offer three privileges modes: *user*, *kernel*, and *hypervisor*. The Arm TrustZone solution[3] adds an additional mode, the *secure monitor* mode. This mode is used by the *secure monitor* code (the "Trusted Firmware"), having the highest privileges, to manage the transitions between the Secure World and the Non-secure World. Beyond that, a NS bit is used to discriminate physical addresses between the Secure World and Non-secure World. The interrupts can be directly trapped to the secure monitor.

These mechanisms combined with the previously available privilege modes addresses the following requirements: the NS bit protects the trusted app's assets and partitions the resource access; the secure monitor code guarantees world isolation, including when managing interrupts, and makes the overall execution efficient; and the privilege modes isolate software within worlds.

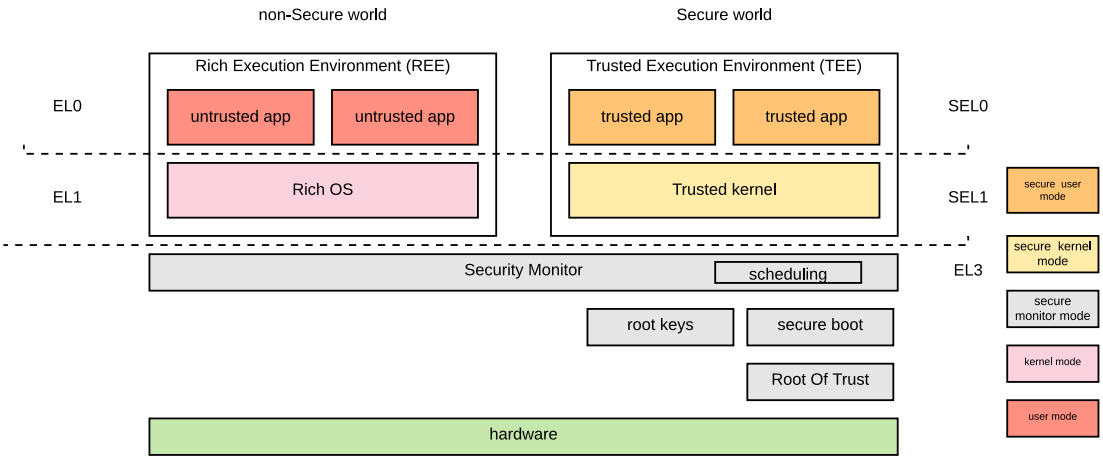

Figure 2: *Arm TrustZone TEE/REE Architecture*

This architecture is used on Arm Cortex®-A cores. The architecture on Cortex-M cores is slightly different in usage, and the operations of TrustZone for Armv8-M are very different as they are optimized for embedded systems that require real-time responsiveness. Function calls and returns are used to perform the switch between the two security worlds, instead of exception taking and return.

1.5 TrustZone Pros and Cons

Pros are often cons and vice versa:

- One secure world and one unsecure world, not more
    - Per processor
- Hardware resources are only accessible to secure world software

- Resources are statically allocated
- Secure world is a *slave*, not intended for whole applications
- Pre v8.4: *principle of least privilege* is not enforced as EL3 and SEL1 have the same level

1.6 WorldGuard Solution

While the WorldGuard solution[4] can easily address the TEE/REE architecture with two distinct cores using the core-driven scheme, it also applies to a single core with the process-driven scheme (note that it may be a business/technical choice or a constraint or benefit to have only a single core). To avoid confusion, the WorldGuard worlds are noted *worlds* (in italic) in order to distinguish from the TEE/REE "Secure and Non-secure Worlds."

Considering the generic TEE/REE architecture and requirements, the WorldGuard configuration is simple to define as software blocks and their associated resources (memories, peripherals) easily fit into the *worlds* definition.

Typically, SiFive uses the same architecture; we use privilege modes to vertically split the security monitor from the OS/kernel and apps, and the WorldGuard *worlds* to horizontally split the Secure World and the Non-secure World:

- The TAs are independent
- The Trusted Kernel and Rich OS are on the same privilege mode, the supervisor mode
- The Security Monitor is in M-mode, managing communication between the two contexts, managing the interrupts, and resource management
- One world in M-mode: the *trusted-WID*
- Two worlds in S-mode: *kernel-WID* and *rich-OS-WID*
- Two worlds in secure U-mode: *TA#1-WID* and *TA#2-WID*
- Two worlds in normal U-mode: *UA#1-WID* and *UA#2-WID*

After the secure boot execution, the WorldGuard configuration code is executed:

- WID-list registers in M-, U-, and S-modes
- WG-PMPs and WG-filters
- Masters configuration (if any, e.g., DMA)

The security monitor (SM) is started in M-mode. The SM can change the S-marker and U-marker registers, and the S-list and U-list registers. It then configures the S-WID-marker with *rich-OS-WID* and starts the rich OS in S-mode.

As in a regular software architecture, the normal OS runs and switches from UT#1 to UT#2, managing the U-WID-marker register (with *UT#1-WID* and *UT#2-WID* values, checked by the normal-U-list register).

If an interrupt occurs, the SM manages it in M-mode. Depending on the interrupt, the SM configures the registers and WID accordingly and executes the correct handler.

In fact, the WorldGuard solution goes far beyond the split between TEE and REE, as it also provides isolation between untrusted tasks and isolation between trusted tasks.

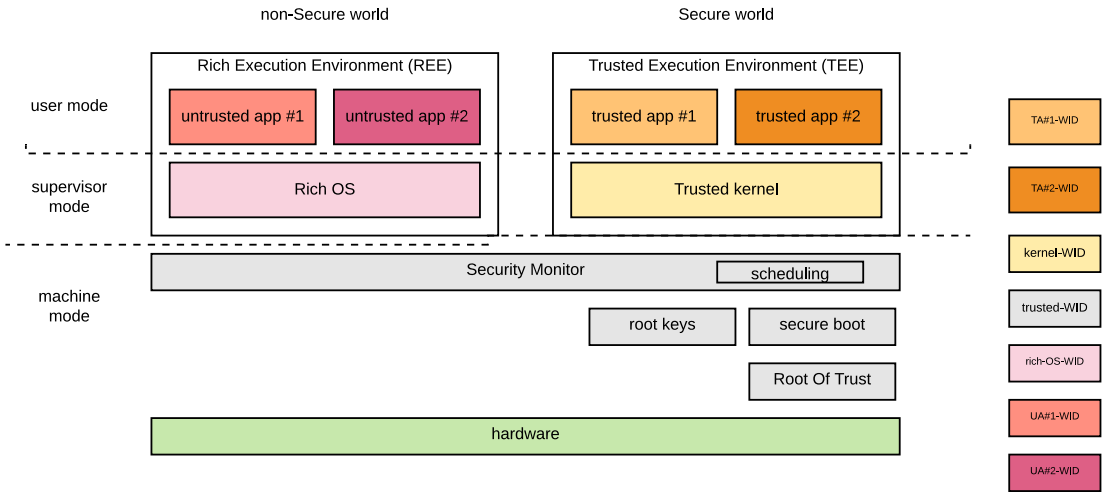

Figure 3: *WorldGuard TEE/REE Architecture*

1.7 Conclusion

We have described how the SiFive WorldGuard isolation solution can easily and efficiently address the strong security requirements of a TEE/REE architecture.

References

[1] SiFive, Inc., WorldGuard White Paper Version 1.2, December 2020.

[2] "Trusted execution environment." [Online]. Available: https://en.wikipedia.org/wiki/Trusted_execution_environment.

[3] Arm Ltd., "TrustZone." [Online]. Available: https://developer.arm.com/ip-products/security-ip/trustzone.

[4] SiFive, Inc., "SiFive Shield: An Open, Scalable Platform Architecture for Security," October 2019. [Online]. Available: https://www.sifive.com/blog/sifive-shield-an-open-scalable-platform-architecture.