2019 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
GROUND SYSTEMS CYBER ENGINEERING TECHNICAL SESSION
AUGUST 13-15, 2019 - NOVI, MICHIGAN

HIGH PERFORMANCE TRUSTED EXECUTION ENVIRONMENT

Jonathan Kline

CTO, Star Lab, Huntsville, AL

ABSTRACT

This paper explores the construction of a Trusted Execution Environment (TEE) which doesn't rely on TrustZone or specific processing modes in order to achieve a high-performance operating environment with multiple layers of hardware enforced confidentiality and integrity. The composed TEE uses hardware intellectual property (IP) blocks, existing hardware-level protections, a hypervisor, Linux security module (LSM), and Linux kernel capabilities including a file system in order to provide the performance and multiple layers of confidentiality and integrity. Additionally, the TEE composition explores both open source and commercial solutions for achieving the same result.

Citation: J. Kline, "High Performance Trusted Execution Environment", In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 13-15, 2019.

1. INTRODUCTION

Security practitioners are playing a serious game of catch up with respect to the proven ability of nefarious actors to exploit our Nation's weapons systems [1] through cyber, logical, and side channel attacks. Common weapons in the fight against attackers, Trusted Execution Environments (TEEs) [2] are used in multiple industries for a variety of tasks including key management, digital rights management (DRM), and safety critical functions. While TEEs can easily be leveraged in the ground vehicle domain to enhance security, they frequently come with hits to performance and functionality, as they are generally implemented within the context of ARM TrustZone [3]. The use of TrustZone significantly inhibits the complexity and flexibility of these applications, which also increases the performance overhead associated with such solutions, making them less than desirable for safety-critical configurations and systems. The underlining premise of TrustZone is that it operates at a hardware-enforced privilege level separate from user facing applications and has dedicated memory thereby providing increased isolation and separation for these critical tasks.

This same premise of enhanced hardware isolation and privilege separation can be extended through the use of other hardware constructs in conjunction with a hypervisor, existing open source / commercial Linux solutions, and platform capabilities, in order to create high performance TEE environments with full access to peripherals and interconnects for a variety of ground vehicle applications. Using an internally funded effort as an exemplar, this TEE development and demonstration discussion will combine: 1) open source components including a hypervisor and Linux kernel capabilities; and 2) existing Xilinx IP / Application notes to establish a high-performance TEE with full peripheral support on the Xilinx UltraScale+™ MPSoC platform [4]. The exemplar and resulting TEE solution will be usable in safety critical environments, such as those used within the context of ground vehicles.

The resulting TEE will be equally applicable to current and future Xilinx parts as well as other ARM-based system on a chip (SOC)s both with and without programmable logic (PL). Similar solutions could also be developed and implemented in most modern Intel platforms, which have support for hardware virtualization and an Input / Output Memory Management Unit (IOMMU). In addition to establishing the requirements for high performance TEEs and discussing the attacks they thwart, this TEE construction and demonstration will also use an iterative, hands-on approach to establish a full TEE on a Xilinx MPSoC.

The built-up TEE will execute entirely within the application processor (i.e. A53) environment of the MPSoC, independent of ARM TrustZone. The application processor TEE will be established as a combination hypervisor / secured Linux environment. Additionally, the exemplar TEE construction will configure of all the platform memory protection units (MMU), that enables both TEE applications and traditional system-level applications to execute concurrently on the same platform (with multiple levels of enforced separation and isolation). The hypervisor and Linux protections used to create the TEE will be demonstrated (and walked through) using a commercial product, the Crucible Security Suite. Throughout discussions, it will be highlighted where existing Open Source and Commercial solutions diverge. However, it should be noted, a similarly high performant TEE could be created with Xen (or any other hypervisor) [5], as well as existing Linux Security modules (LSM), and authentication / encryption components.

The TEE will specifically leverage existing Xilinx security and isolation features within the MPSoC (and will identify what types of features would be required on other SOCs). The resulting TEE buildup will provide integrators and system developers with first-hand knowledge on how to integrate: 1) multiple open source and/or commercial components; 2) ARM capabilities; and 3) Xilinx IP blocks into high performance TEE solutions, with full access to peripherals, into their designs. The buildup and walkthrough will also highlight customer facing concerns of a TEE such as: 1) performance / throughput of peripherals, 2) impact on total system utilization, 3) ability to meet safety and/or security requirements, and 4) IP blocks used for separation and isolation of the TEE environment. The end result will be an easy-to-follow, repeatable process for seamlessly integrating isolation and security into the production environment of typical ground vehicle programs.

2. TEE Fundamentals

At its core, a TEE is a standalone execution environment which uses hardware isolation / security features and a separate run-level execution environment in order to perform sensitive operations. A TEE provides confidentiality and integrity services for their applications. Many processors (more specifically, ARM-based) and SoCs which use an ARM core utilize a separate execution mode (i.e. TrustZone) to provide a TEE. The use of TrustZone requires these processors to stop the main execution thread / OS, perform a context switch into the secure / trusted state, and perform TEE operations. Similarly, while not strictly a TEE, Intel processors utilize System Management Mode (SMM) in order to perform sensitive system management operations, such as power savings. Intel processors also provide the software guard extensions (SGX), which enable the execution of secure enclaves using TEE principles. Whether it's SMM, SGX, or TrustZone, the use of these operating modes provides hardware protected memory ranges, limited access to peripherals, and separate hardware-enforced privilege levels for execution.

Generally, a TEE consists of a dedicated hardware environment, and either a standalone application or a combination operating system (OS) and application stack. Additionally, most TEEs implement some form of the Bell–LaPadula model [6] with *read down, write up* for data. This enables the TEE to access all (or at least the majority of) memory on the device, while still protecting the TEE's memory itself.

One of the key challenges with a TEE is that of overall system performance, and most TEEs don't have a separate execution core, which prohibits concurrent execution with the main system processing. In order to enter into TEE execution, all (unsecure) processing on the host must be stopped, and the secure processing mode must be entered. This causes not only a context switch, but also forces the TEE to execute in a highly time constrained environment, much like a device driver, limiting the amount of processing which can be performed without impacting the rest of the executing applications (which may be safety critical, therefore have hard real-time constraints on execution).

On modern, multicore processors and SoCs, a hypervisor enables TEEs to be constructed in a highly performant, robust manner (see Figure 1) while still affording the key properties of hardware-enforced confidentiality and integrity for TEE applications.

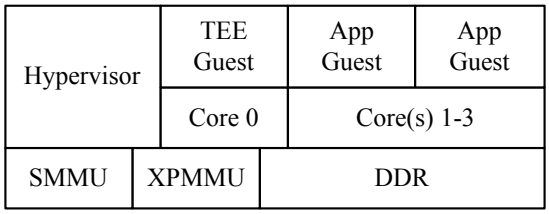

Figure 1: Hypervisor TEE Construction

The use of a hypervisor to isolate system resources enables a TEE to be dedicated to one or more execution cores, which are able to operate concurrently with main system operations that execute on the remaining system cores. In this way, TEE operations do not interrupt or hinder general system operations, while preserving the hardware enforced principals of confidentiality and integrity.

3. Hypervisor Overview

Modern computing architectures support the use of a hardware accelerated, virtual machine monitor (VMM). The VMM or hypervisor, enables software to assert positive control over all platform resources including memory, processors, and I/O devices. This positive control enables

resource assignment (and containment), access controls, and the sharing of system resources. Modern VMMs, such as Xen, rely on hardware acceleration for memory and process management, as well as device / guest isolation.

The VMM configures each guest through the creation of a control data structure and specialized machine instructions. This guest data structure identifies the physical resources (i.e. memory, CPU cores and/or timeslices, and peripherals) a guest has access to, and is enforced by the underlying hardware. Additionally, the guest data structure enables the hypervisor to define events (i.e. instructions, memory access faults, IOPORT access, etc.) which stop the guest and transition back into the hypervisor

System events such as a configured timer interrupt can be used by the hypervisor to task switch between multiple guests, thereby sharing the CPU's resources. Similarly, the hypervisor can elect to have a guest consume all of the CPU's resources by removing events (from the guest data structure) that would normally cause a trap into the hypervisor. Each physical CPU core has an active guest VM data structure pointer, which points to the currently active guest and corresponding guest data structure. The hypervisor can use a privileged set of instructions to have the processor update the guest pointer, and switch to a different guest as required for the overall platform configuration.

3.1. Commercial Hypervisor overview

Crucible is a secure execution environment based on the open source, Xen hypervisor. Crucible provides a trusted execution environment that addresses concerns unique to mission-critical computing (more specifically, the aerospace and defense (A&D) industry), including: secure boot, technology protection, cyber resilience, high-assurance operations, deterministic performance, and compatibility with common mission system hardware and software.

Crucible is designed for use in hostile computing environments and operates as trusted supervisory software within the processor – configuring and controlling both hardware resources and software execution in order to ensure and maintain the integrity of system operations. Crucible leverages hardware-based roots-of-trust IP to perform a secure boot process and establish the basis for isolation and separation. During system operation, the hypervisor enforces logical isolation such that software mission loads execute within private enclaves with application-level granularity of hardware-enforced separation. This logical isolation is enforced even though the applications may be running on a single physical processor board, or within a single COTS OS. This addresses safety, security and confidentiality concerns, and improves overall mission assurance – especially given the reality of software faults and sophisticated cyber-attacks. Finally, Crucible has strong technology protections and anti-reverse engineering features built directly into the hypervisor. These features ensure that sensitive software in the system remains protected against unauthorized access, theft, and malicious modification – and that mission-critical functionality continues to operate unimpeded – even in the face of dedicated attackers and reverse engineers.

During boot, Crucible takes over control of the central processing unit (CPU), IOMMU / system memory management unit (SMMU), memory management units / controllers, system configuration, peripheral allocation, and processor management. This enables Crucible to fully control the underlying hardware, and prevent a variety of logical, virtual, and physical attacks against the hypervisor, or the protected operational flight program (OFP) images. This policy is further enabled by the application of Flux Advanced Security Kernel (FLASK) [7] policies which enforce separation between the hypervisor and guests as well as between guests. Additionally, the FLASK policy to configure the hardware, separation, and memory management functionality is incorporated into the secure boot mechanism to enforce confidentiality and integrity requirements.

Crucible provides run-time logical and virtual protection for guest VMs and protected applications. Crucible prevents protected applications from being reverse engineered, analyzed, removed from the system, or transferred to a different platform / system.

4. TEE Construction and Overview

4.1. TEE Requirements

In order to establish confidentiality and integrity for the TEE, the following must be afforded: secure boot, configuration integrity, deterministic performance, mandatory access controls, isolation and partitioning. All of these requirements can be met through a combination of platform (i.e. Xilinx) IP, and commercial or Open Source software stacks. It is essential that these requirements be enforced within the TEE guest(s) as well as guests that are running external to the TEE, otherwise additional side-channel attack vectors against the TEE may be introduced.

4.2. TEE Overview

The highly performant TEE will execute completely within the application processing unit (APU) (at exception level (EL)1/EL2) and will provide freedom of interference from other applications in the application processing unit (APU). Further, the TEE will be isolated from applications executing within the real-time processing unit (RPU), and peripherals across the system (including those within the fabric itself). The TEE will not utilize TrustZone within the APU in order to provide a higher performing solution with full access to system peripherals. By not utilizing TrustZone, the TEE is able to execute concurrently with the main applications

executing on the APU, and support safety-critical applications.

The TEE is composed of the Xen hypervisor (configured as a separation / isolation kernel), one or more Linux Security Modules (i.e. SELinux [8] and/or AppArmor [9]) executing at EL1, and TEE applications executing at EL0. Additionally, to complement the LSM, either a filesystem which encrypts and authenticates files, or if required, various Linux capabilities such as IMA to provide integrity services will be used. Alternatively, commercial solutions such as the Crucible Security Suite, combine a hypervisor tailored for secure boot, configuration integrity, deterministic performance, mandatory access controls, isolation and partitioning can be used to establish and provide the TEE, while also enabling a wide variety of data at rest and general cyber security concerns to be met.

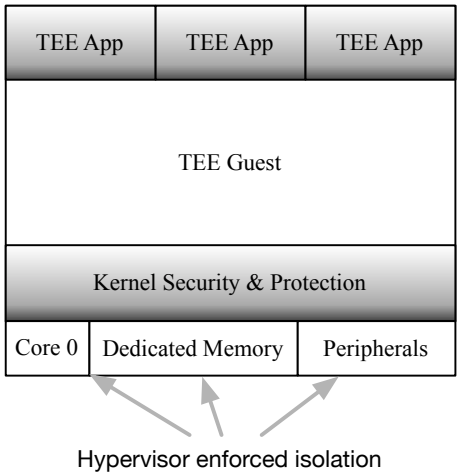

Figure 2: TEE Design

The TEE is established (i.e. secure boot) through a combination of Xilinx IP blocks and existing processes, and is used to authenticate, isolate and launch TEE applications (see Figure 2) while continuing to provide confidentiality and integrity.

Using the previously developed exemplar, the TEE construction and demonstration will walk the user through configuring Xen on ARM to function as a real-time separation and isolation kernel. This will enable the TEE to be isolated to one or more APU processing cores, while also enabling the TEE to execute concurrently with the main applications on the system. This configuration will highlight the industry best practices [10] for real-time systems and apply the principles of: (1) real time scheduler use for guests with RT workloads; (2) no paging / memory translation; (3) no hardware emulation; and (4) minimal interrupt latency. This will enable deterministic performance for executing TEE applications and enable developers to better utilize the full resources of the MPSoC or similar ARM-based SoCs. As part of this configuration, 1-core on the platform will be dedicated to TEE operations, thereby removing time sensitive constraints of the rest of the system. This will enable the TEE applications to execute continuously, without interruption and without impacting the performance of the rest of the system. Additionally, this will enable the TEE to have full access to system peripherals, independent of other guests and/or applications.

4.3. TEE Construction

Bootgen [11] will be used to create a secure boot image. The image will contain Xilinx memory management and separation unit configuration (i.e. XPPU / XMMU / SMMU), configuration for an application processor hypervisor (i.e. Xen), applications running lockstep in the RPU, RPU application, u-boot, and an extensible linker format (ELF) image containing the application processor hypervisor and DOM-0. During secure boot, u-boot will authenticate and decrypt the hypervisor, including DOM-0 if necessary, for system construction.

SoCs such as the Xilinx MPSoC provide a wide variety of mechanisms that can be used for key (i.e. encryption and authentication keys for secure boot) storage. While the key storage location is largely dependent on individual system characteristics and the threat model, it is also largely irrelevant to the construction of a TEE, as long as one or more encryption / authentication keys are made available. The Xilinx App Note 1323 will be used to configure secure boot for the reference implementation.

One key aspect of a TEE that is important to achieve is that of reduced attack surface. Xen on ARM can be reduced to around 50KLOC, with opportunities to further reduce this based on operational constraints and further use of the KBUILD environment to remove unneeded features and functionality from the core hypervisor being used to establish the TEE runtime. The construction and exemplar demonstration will also highlight ongoing work within the Xen community to develop dom0-less and safety-critical capabilities for the ARM platform, which have the potential to enable even more fined grained resource utilization and deterministic execution, further reduce the attack surface, and provide enablement for the generation of safety critical hypervisor artifacts for the core hypervisor.

The hypervisor will configure the system memory management unit (SMMU) to assign peripherals and memory access requests to specific guests. This will enable the guests, stored on the platform secure digital (SD) card, to be authenticated and launched in a secure fashion thereby extending trust from platform secure boot up to virtualized guests executing in the application processor. The Xilinx secure boot will be used to authenticate and decrypt the hypervisor (and associated configuration), as well as enable the key release to continue the boot process. Once the hypervisor's control domain has been authenticated and

decrypted, the individual guests (including the TEE) can be read from the platform's SDCARD and authenticated and/or decrypted as required by each guest. The authenticated and decrypted guests will then be executed with hardware enforced separation / isolation and unique system resources.

During boot of the application processor, the guests and individual TEEs, the guest's key encrypting keys (KEKs) will be retrieved from the platform's secure key storage. A chain of trust (authentication / encryption) is then established from the TEE applications to platform secure boot with isolation being enforced at multiple levels using hardware IP blocks, ARM EL2/1, and software.

Through the use of an LSM, the kernel (EL1) in each guest (including the TEE) will further restrict peripheral and bus access (configured in the XPMU to restrict access to the application processor) to limit peripheral access to a specific TEE application. The LSM in the TEE (guest) will also be responsible for enforcing mandatory access controls around the TEE application(s), in order to further decrease the attack surface of the TEE. In order to provide additional confidentiality and integrity for the TEE applications, several different options will be presented: (1) the use of an encrypted filesystem and Linux IMA; (2) the use of device mapper encrypt and integrity layers; and (3) commercial filesystems such as Star Lab's *fortifs* which provide both integrity and confidentiality of the underlying data. These options will enable confidentiality and integrity to be preserved for the TEE applications from rest through runtime and into retirement. It should be noted that encrypting file systems such as *ecryptfs* do not provide for integrity of data and are only focused on confidentiality, therefore they provide an incomplete solution for use with a TEE. Regardless of the approach chosen, key management will be offloaded to the platform (and will be implemented external to the high-performance TEE).

5. High Performance TEE Summary

This exemplar TEE will lay the ground work for using a hypervisor and Linux kernel capabilities for establishing a high-performance TEE on the Xilinx MPSoC, that operates independent of ARM's TrustZone, providing benefit of improved complexity, flexibility, and access to peripherals to the trusted applications, as well as improved concurrence with non-trusted safety critical applications thereby improving their performance as well. The demonstration and exemplar TEE will establish the background required for the establishment of a TEE external to TrustZone, the use of a hypervisor in a real-time environment, and provide a hands-on approach to integrating multiple components from industry and the community. The benefit to the ground vehicle domain will be an easy to follow recipe for configuring, implementing, and verifying enhanced security in a variety of critical use cases where performance and functionality cannot be degraded with the addition of security features. The construction of a TEE external to TrustZone will provide the ground vehicle domain with multiple paths and design flows for integrating security into the production process and leveraging all of the features of complex SOCs

6. References

[1] US Government Accountability Office, "WEAPON SYSTEMS CYBERSECURITY: DOD Just Beginning to Grapple with Scale of Vulnerabilities," GAO, 2018.

[2] Global Platform, "Introduction to Trusted Execution Environments," Global Platform, 2018.

[3] ARM Ltd, "ARM Trust Zone," 2019. [Online]. Available: https://developer.arm.com/ip-products/security-ip/trustzone. [Accessed 14 06 2019].

[4] Xilinx Corporation, "Zynq UltraScale+ Device Technical Reference Manual," Xilinx Corporation, 2019.

[5] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt and A. Warfield, "Xen and the Art of Virtualization," in *ACM SOSP '03*, 2003.

[6] D. E. Bell and L. La Padula, "SECURE CCMPUTER SYSTEM: UNIFIED EXPOSITION AND MULTICS INTERPRETATION," The MITRE Corporation for Deputy for Command and Management Systems Electronic Systems Division, AFSC , Bedford, MA, 1976.

[7] Secure Computing Corporation, "Assurance in the Fluke Microkernel Final Report," Secure Computing Corporation, Roseville, 1997.

[8] S. D. Smalley, C. Vance and W. Salamon, "Implementing SELinux as a Linux Security Module," NAI Labs, Ft Meade, 2006.

[9] V. Danen, "Immunix System 7: Linux security with a hard hat (not a Red Hat)," Tech Republic, 2001.

[10] X. Fan, Real-Time Embedded Systems, Newnes, 2015.

[11] Xilinx Corporation, "Bootgen User Guide," Xilinx Corporation, 2018.