Isolation using Virtualization in Secure World

Achin Gupta
Arm Ltd.
Advent of Arm TrustZone

Arm TrustZone was introduced in ARMv6K in 2003
- Enabled system wide partitioning of resources between Secure and Normal worlds

A TEE was enabled through a combination of
- TrustZone based hardware isolation
- Trusted Boot
- Trusted OS

TEE offers security properties of confidentiality and integrity to Trusted Apps.

Trusted Apps provide security services for:
- Authentication and crypto
- Integrity Management
- Payment
- Content protection
- Mobile device management
- And more use cases....
Advent of EL3 in Armv8-A

EL3’s separate translation regime and exception handling enabled:
  • Isolation of secure monitor in a separate binary image
  • Standardization of key platform management functions e.g.
    – Power & Errata management, SiP services etc

Standardization paved way for specifications e.g.
  – SMC Calling Convention, PSCI, SDEI etc

These developments lead to the Trusted Firmware open source project
  • Provides a reference implementation of specifications and Trusted boot.
Secure world is getting bigger and diverse!

Firmware components are being provisioned by multiple vendors
- E.g. generic, silicon vendor, OEM, Trusted OS vendors
- Use of EL3 firmware for platform management functions is increasing
- Components from different vendors should be isolated from each other

Silicon vendors and OEMs have expressed requirement for multi-tenancy in S-EL1
- There are Trusted application ecosystem challenges
  - Trusted Applications are Trusted OS specific and this limits their portability
  - OEMs want to ship a rich set of applications, with different applications tied to different Trusted OSs
- There are challenges in integrating code from multiple vendors in a Trusted OS
  - Silicon vendor drivers could be integrated into a 3rd party Trusted OS
  - Silicon vendor could package drivers in its Trusted OS. This is integrated with a 3rd party Trusted OS
- Normal world drivers for each Trusted OS have to coexist
- S-EL1 tenants need to be isolated from each other to limit the available attack surface
Root causes of difficulty

Inability to apply principle of least privilege

• A component must access only those resources that are necessary for its correct operation
• Cannot apply this principle in EL3 and S-EL1 in Armv8.3 and earlier
  – Both ELs have same visibility of physical address space and interrupts
  – Not possible to isolate EL3 firmware from a Trusted OS
  – Not possible to isolate components within EL3 firmware and Trusted OS from each other
  – Normal world cannot be protected from privilege escalation attacks on Trusted OSs
  – Hardware resources cannot be isolated to a particular software entity
• This increases the complexity of auditing and certification
  – All software components need to implicitly trusted each other, and therefore, cannot be audited separately

Lack of standard interfaces at component boundaries

• Increases difficulty of integration and interoperability between SW components
Required solution

Architectural support to provide hardware isolation between software components

• Isolation requires restricting access to physical address space & registers from
  – Processors
  – Direct Memory Access (DMA)-capable peripherals.
• This removes the need for mutual trust and allows components to be audited separately.

A software architecture that provides standard interfaces at component boundaries

• Enable the ecosystem of vendors to work together
• Enables distinct software to interoperate. This promotes generalization and componentization of code.
• Enables removal of Trusted OS vendor specific code from secure firmware and EL2.
Virtualization in Secure world

Armv8.4 architecture adds virtualization support in the Secure state
- Brings all virtualization features available in the Non-secure state to the Secure state
- Adds the Secure EL2 (S-EL2) exception level
  - This allows hypervisor control visibility of physical memory from a virtual machine

Arm System MMU architecture 3.2 adds support for Secure Stage 2 translations
- This provides address translation for non-processor masters
- Uses same translation table format as the processor

Arm GIC architecture 3.1 adds GIC Secure Virtualization(GSV) extension
- Adds support for a virtual GIC in the Secure state
Secure SW architecture based on virtualization

Virtualization provides hardware enforced isolation that enables:

- Isolation of EL3 software from Secure EL1 software
- Isolation of Normal world software from Secure EL1 software
- Isolation of distinct Secure EL1 software components from each other

This is not enough to avoid vendor specific code for communication and interoperability

Proposed SW architecture = Generalisation of existing SW architecture in Secure world

- Enables generalisation of communication between software components through standard ABIs for
  - Message passing
  - Memory sharing
- Provide a generic framework for sharing resources between components
Standard building blocks

Secure Partitions

• A secure world virtual machine with an isolated address space to only access resources it needs
  – Can be used to host a Trusted OS or a driver stack for a Trusted hardware resource
• Exports security services that Normal world clients and other SPs access

Resource description

• A manifest that describes the resources a SP needs and services it provides
  – E.g. list of devices, interrupts, memory regions it needs.

Secure Partition Manager

• A generic firmware component in S-EL2 for managing secure partitions
  – Can be thought of as a minimal partitioning hypervisor that replaces need for multiple Trusted OS dispatchers
  – Responsible for enforcing principle of least privilege by using a SP’s resource description
  – Responsible for initializing a SP at boot time and managing its requests at runtime
  – Responsible for enabling communication between service requestors and providers at runtime
It starts to look like this!

Diagram showing the normal world and secure world, with isolation boundary, SEL2 Firmware, Secure Partition Manager, EL3 Firmware, Resource Description, and various components like Trusted Application, Client Application, Client Library, Operating System Kernel, Hypervisor (optional), and Secure Partition with Trusted OS and SiP drivers.
Standard building blocks

Secure Partition Client Interface (SPCI)
- Describes ABIs between clients and providers of services in secure partitions to:
  - Enable message passing and memory sharing between them
- Avoids vendor specific drivers in the Normal world hypervisor and EL3 firmware
- Provides a SMC based transport for vendor specific drivers in Rich OS e.g. a Trusted OS driver

Secure Partition Run time (SPRT)
- Describes the run time model that each SP depends upon to implement secure services
  - E.g. request dispatch policy, allocation of cpu cycles etc
- Specifies information to be included in a Resource description
- Describes ABIs between SPs and SPM to:
  - Initialize SPs
  - Dispatch requests to a SP and obtain corresponding responses
  - Dispatch interrupts to a SP
How it all fits together!

- **Normal World**
  - Client Application
  - Client Library
  - Operating System Kernel
  - Trusted OS Driver
  - Hypervisor (optional)

- **Secure World**
  - Client Application
  - Client Library
  - Trusted Application
  - TA Library
  - Trusted OS Kernel
  - Secure Partition Manager
  - SEL2 Firmware
  - Platform Firmware
  - SEL3 Firmware
  - Secure partition with trusted OS
  - Resource Description
  - Secure partition silicon vendor drivers

- **Isolation Boundaries**
  - EL0
  - EL1
  - EL2
  - EL3

- **Resource Description**
  - Application trusted OS specific
  - Application provider specific
  - Generic software
  - Silicon Vendor specific software
  - Isolation boundary

Confidential © Arm 2018
Specification status

S-EL2 whitepaper available on developer.arm.com
  • https://developer.arm.com/products/architecture/security-architectures

Secure Partition Client Interface specification Alpha1 available on DropZone
  • https://connect.arm.com/dropzone/systemarch/DEN0077A_Secure_Partition_Interface_Specification_1.0_Alpha_1.pdf

Secure Partition Run Time Alpha specification under development
  • Expected to be available end of Oct’18
Some impacts on Secure world software

Trusted OSs assume access to physical address space for memory sharing with Normal world

- Likely to assume that Normal world sees the same range of physical addresses as they do and pass PAs in shared buffers

Trusted OS implementations assume access to physical interrupts

- Use this capability to prevent uncontrolled preemption by non-secure interrupts e.g.
  - Implement critical sections while handling Yielding calls
  - Use a critical section to handle Fast calls and secure interrupts
  - Control exit to normal world in response to non-secure and EL3 firmware interrupts

Virtualization invalidates some of these assumptions
Some impacts on Secure world software

Can a Trusted OS run in a VM just like a Rich OS VM under the control of a Hypervisor?

SPM constraints access to physical address map using the SP's resource description and stage 2 translations
  • Memory has to be shared in cooperation with SPM
  • Additional translation regime could invalidate assumptions about access to physically contiguous memory
  • Memory has to be mapped with consistent translation table attributes across all translation regimes

SPM manages physical GIC and exposes only the virtual GIC to a SP
  • Trusted OS can no longer mask Normal world interrupts
  • Trusted OS can no longer mask physical secure interrupts
  • Preemption of Trusted OS has to be managed by SPM
  • Runtime model achieved through control of physical interrupts is not possible any longer

SPM replaces dispatcher for dispatching requests to the SP and obtaining responses
Some impacts on Normal world software

SPCI describes a generic message passing and memory sharing interface

- Trusted OS specific message passing and memory sharing interfaces in high level OS drivers can be replaced by SPCI
- Hypervisors can implement a generic SPCI driver to ferry communication between Guest VMs and Trusted OSs
The Prequel: S-EL0 Partitions
Trusted OS adoption is not universal

There are market segments that lack Trusted OS adoption e.g. enterprise, data centers. In these segments firmware (BIOS) is expected to incorporate more services:

- Security related services e.g. access to Secure storage, Verified & Measured boot, Cryptographic services
- Management related services e.g. Error handling, BMC Communication

“Offload” of these services to a controller is one approach but not ubiquitous.

As firmware on the AP is getting bigger and more complex to audit:

- There is a need for separation of liability amongst services through isolation
- There is a need to restrict the level of privilege available to each service
Secure partitions on Armv8.3 and earlier

In the presence of a vacant S-EL1, isolation* can be achieved through SPs in S-EL0
SPM runs in EL3 and firmware owns S-EL1
Role of SPM remains the same but there is no virtualization
S-EL0 partitions can be used for use cases such as:
  • UEFI authenticated variable storage
  • Capsule update
  • RAS error handling
  • BMC communication
SPCI and SPRT enable a SW architecture where both types of SPs can co-exist

*System wide isolation is still not possible in the absence of Arm SMMU support
Confidential © Arm 2018
SP Origins: MM S-EL0 Partitions

Single UP migrate-able secure partition
- UEFI image with Standalone MM support
- Included in ARM TF boot flow as BL32 image
- Run-to-completion runtime model

Normal world uses MM_COMMUNICATE SMC to request partition services

ARM TF BL31 stage owns EL3 and S-EL1

Secure partition resources are described in BL31 platform port

Secure Partition Manager in BL31 exports standard ABI to
- Initialize the partition
- Delegate SMC requests to the partition
Thank You!
Danke!
Merci!
谢谢!
ありがとうございます!
Gracias!
Kiitos!