HandsOn Training

arm

APPROVED TRAINING PARTNER

TrustZone for Armv8-M

Course Description

This course provides all necessary information on how to design a secure IoT device using Armv8-M architecture-based microcontroller.
Trustzone for Armv8-M technology adds security partitioning to Arm Cortex-M processors.
It can be used to design a secure IoT device using different Arm technologies including an Armv8-M processor, TrustZone Cryptocell IP and industry standard techniques for developing software.
Correctly combined these technologies can be used to design a system that can achieve Platform Security Architecture (PSA) certification.
This course covers the architectural features that underpin the security partitioning at a software level and how security can be implemented in the wider system using AMBA AHB5.
Extensive hands-on labs are used to give trainees some practical experience on how to create secure and non-secure applications mapped appropriately to secure and non-secure memories, using secure APIs and TrustZone-aware compiler toolchain, as well as demonstrate various attacks and their countermeasures.

At the end of the course the participant will receive a certificate from ARM.

Course Duration

3 days (with hands-on labs)

Goals

1. Understand the security need and how TrustZone address it versus traditional secure architectures
2. Be able to configure the security attribution unit and protected memory units
3. Become familiar with the secure world requirements
4. Handle secure and non-secure interrupts within various use cases
5. Understand the requirements from secure boot
6. Manage memory system with TrustZone
7. Become familiar with compilers support
8. Become familiar with various software attacks and their countermeasures
9. Become familiar with trusted firmware for Cortex-M
10. Become familiar with secure system design considerations
11. Become familiar with TrustZone system IPs for secure system

Target Audience

Hardware, software and security system architects who need to understand the issues in developing trusted systems using Armv8-M based microcontrollers.

Prerequisites

- Knowledge Armv8-M architecture
- Experience with C programming
- Experience of programming in assembler is useful but not essential
- Some experience with embedded systems software design

Course Material

- Arm official course book
- HandsOn-Training Virtual Machine with Jupiter Notebooks (contains all lab instructions), and all source files and required tools.

Agenda

Main Topics:

- Introduction to TrustZone Security
- TrustZone for Armv8-M Overview
- Armv8-M Security Attribution
- TrustZone for Armv8-M Toolchain Support
- Armv8-M Exception Handling
- Toolchain Support for the Armv8-M Security Extension
- Armv8-M Secure Software Design Considerations
- TrustZone System IP for Embedded Systems
- Trusted Firmware for Cortex-M
- Armv8-M Secure System Design Considerations
- Hands-On Labs

Day #1

❖ **Introduction to TrustZone Security**

- Security Principles & Concepts
  - What are we protecting?
  - What are we protecting assets from?
  - Summary of attacks and defenses
  - Initial Root of Trust & Chain of Trust
  - Secure domain
  - Examples of secure domain practices
  - Example: security in IoT applications
  - Typical processor secure hierarchy
  - Security magnitude
  - The goal and limitation of secure design
  - How the secure hierarchy works

- Existing Security Solutions for Arm MCUs and Application Processors
  - Memory protection/management units
  - Execute-only support
  - SecurCore – Security against physical attacks
  - Functional safety

- TrustZone for Armv8-M
  - Security on next-generation Cortex-M
  - API for interface to Secure state: CMSIS

❖ **TrustZone for Armv8-M Overview**

- Programmer's Model
  - Introduction to TrustZone for Armv8-M
  - Secure and Non-secure states
  - Calling between security states
  - General-purpose register banking
  - Special-purpose register banking

- Memory Configuration
  - Memory security
  - Memory security determination and MPU selection
  - Secure and non-secure view of SCS

- Switching Between Security States
  - Branching between secure and non-secure states
  - Function calls using branch instructions

When innovation meets expertise...

- Security state changes using software
- TT instruction

- Exceptions
  - Interrupts and exceptions

❖ Armv8-M Security Attribution

- Memory Security
  - How physical memory is split
  - Memory security determination
  - Memory Protection Unit
  - Memory access basics
  - Secure view of SCS
  - Non-secure view of SCS

- SAU Configuration
  - SAU registers
  - Boot security map
  - Runtime security map
  - SAU region configuration
  - Enabling the SAU
  - Configuring the SAU with CMSIS

- IDAU
  - Cortex IDAU implementation
  - Simple IDAU design example
  - Simple NSC arrangement

❖ TrustZone for Armv8-M Toolchain Support

- Arm C Language Extensions (ACLE)
  - Calling non-secure code from secure code
  - BXNS and BLXNS instructions
  - Calling secure code from non-secure code
  - Creating an import library in Arm Compiler
  - Using the import library
  - Secure gateway veneers
  - NSC veneers in Arm Compiler
  - Configuring the SAU with CMSIS
  - TT instruction

Day #2

❖ **Armv8-M Exception Handling**

- What Happens After Reset
    - Taking a reset exception
    - Vector table for ARMv8-M Baseline
    - Reset behavior
    - Non-secure boot

- Taking an IRQ
    - External interrupts
    - Taking an IRQ
    - Exception model
    - Secure -> non-secure exceptions
    - Stack frame layout
    - Register values after context stacking
    - Integrity signature

- Returning from an IRQ
    - Returning from an exception
    - IRQ exception return example
    - EXC_RETURN

- Configuring IRQs
    - NVIC configuration registers
    - IRQ security
    - Secure exception prioritization
    - Exception priorities overview

- Pre-Emption & Tail-Chaining
    - Chaining secure and non-secure exceptions
    - Pre-emption
    - IRQ nesting

- Other Exceptions
    - Internal interrupts
    - System handler priority
    - Fault exception in Armv8-M

❖ **Armv8-M Secure Software Design Considerations**

- Introduction to Low Level Software Attacks
    - Introduction to low level software security

  - Attack drill: format string attack
  - Defense against format string attack
  - Attack drill: timer bomb with format string attack
  - Attack drill: unauthorized access
  - Defense against unauthorized access
  - Defense against parameter tampering
  - Attack drill: stack smashing
  - Defense against stack smashing
  - Attack drill: code injection
  - Attack drill: Return Oriented Programming (ROP)

- Design for Testing
  - Favor simplicity rather than flexibility
  - Favor templates rather than meta-APIs
  - Last stand of defense – White Hat Team
  - Request/audit service model
  - FSM based user behavior constrain paradigm
  - Intra-secure domain isolation – sandbox
  - Conclusion

❖ TrustZone System IP for Embedded Systems

- SoC Security
  - Secure memory rules
  - System level memory partitioning
  - IDAU and IDAU-lite
  - CoreLink SIE-200
  - Legacy masters (non-security aware)
  - Programmable masters
  - AHB5 TrustZone master security controller
  - Peripheral gating
  - AHB5 TrustZone peripheral protection controller
  - AHB4 TrustZone peripheral protection controller
  - Memory gating
  - AHB5 TrustZone memory protection controller
  - Secure aware peripherals
  - Error reporting

Day #3

- **Trusted Firmware for Cortex-M**

  - TF-M Overview
    - Platform security architecture
    - PSA Firmware framework concepts
    - PSA Firmware isolation levels
    - TLS session example
    - Trusted Firmware -M
    - PSA certified levels
    - PSA functional API certification

  - TF-M Framework
    - Secure partition
    - Secure Partition Manager (SPM)
    - Secure service
    - Library mode
    - IPC mode
    - PSA-RoT in TF-M
    - TF-M Crypto service
    - Secure storage
    - PSA secure storage
    - PSA initial attestation
    - Secure boot flow
    - Bootloader in TF-M

  - TF-M Build System
    - Build environments
    - Support compilers
    - Build options
    - Debug tools

- **Armv8-M Secure System Design Considerations**

  - Overview
    - Security vs safety
    - Secure domain
    - System secure domain model
    - Interface-oriented security
    - Isolation of shared and non-shared resources
    - Access attribution management
    - Security requirements for resource manager

  - Security Hierarchy

  - Typical processor secure hierarchy
  - Level of security strength
  - The goal and limitation of secure design
  - How the secure hierarchy works
  - Summary of attacks and defenses
  - General principle of secure system design

- Single and Multi-Secure Domain Model
  - Security analysis
  - Single secure domain model
  - Armv8-M system example
  - Physical isolation in bus matrix
  - Software IP security
  - Requirements for secure boot
  - Low level attacks
  - Multi-secure domain model
  - Sandbox implementation

- Inter-Secure Domain Models
  - IoT network example
  - Communication security
  - Frame overflow attack
  - Denial of services and sleep
  - Puzzle player
  - OTA security
  - IAP/ISP security