TrustZone for Armv8-M

Course Description

This course provides information on how to design a secure IoT device using different Arm technologies including an Armv8-M processor with built-in security partitioning, TrustZone Cryptocell IP and techniques for developing software that is able to hide assets from attackers.
The training also covers how operating systems like RTX and mbedOS have been extended to provide a secure element capable of storing keys securely and executing crypto functions.
Workbooks 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.

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

Course Duration

2 days (3 days with hands-on labs)

Goals

1. Understand the security need and how TrustZone address it versus traditional secure architectures
2. AArch32 and AArch64 architecture review
3. Become familiar with the secure world
4. Handle secure and non-secure interrupts
5. Understand the requirements from secure boot
6. Understand the secure monitor role and behavior
7. Manage memory system with TrustZone
8. Become familiar with the virtualization extension for increased security
9. Become familiar with TBBR and trusted firmware
10. Utilize the TrustZone software stack
11. Use TrustZone IPs for secure system
12. Understand what is secure debug and how to implement it
13. Become familiar with TrustZone ecosystem

Target Audience

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

Prerequisites

- Knowledge of existing M-profile devices
- Knowledge of programming in C
- Experience of programming in assembler is useful but not essential
- Some knowledge of embedded systems

Course Material

- Arm official course book
- Arm official Workbook

Agenda

Main Topics:

- Introduction to Security
- Armv8-M Overview
- Introduction to the Armv8-M Security Extension
- TrustZone for Armv8-M System IP Overview
- Toolchain Support for the Armv8-M Security Extension
- Exception Handling for the Armv8-M Security Extension
- Security Attribution
- Armv8-M Security Extension Workbook

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
  - 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

❖ **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

Day #2

❖ **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 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
- **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