



SEC8 - Secured Embedded Linux Platform Build

Objectives

- Implementing secure boot
- Verifying the authenticity of system components before they are loaded and executed.
- Ensure the authenticity and integrity of the bootloader, kernel
- Implements the Trusted Boot
- Provides a secure environment for the secure monitor firmware
- Run OP-TEE on secure environment that runs alongside the main operating system

Labs are conducted on QEMU ARM-based board

Prerequisite

- C Language knowledge (see for example our L2 training course)
- Embedded Linux Build knowledge (see for example our D1 training course)
- You may be interested also by the SEC9 Advanced Embedded Linux Security course
- You may be interested also by the SEC1 Secure Development for Embedded System course
- You may be interested also by the SEC2 Advanced Embedded Systems Security course

Equipment

- Training manuals and software exercises
- One Linux PC for two trainees
- One target platform for two trainees

Duration

- Total: 2 days
- From 40% to 50% of training time is devoted to practical activities

Course Environment

- Theoretical course
 - PDF course material (in English) supplemented by a printed version.
 - The trainer answers trainees' questions during the training and provide technical and pedagogical assistance.
- Practical activities
 - Practical activities represent from 40% to 50% of course duration.
 - Code examples, exercises and solutions
 - One PC (Linux ou Windows) for the practical activities with, if appropriate, a target board.
 - ▶ One PC for two trainees when there are more than 6 trainees.
 - For onsite trainings:
 - ▶ An installation and test manual is provided to allow preinstallation of the needed software.
 - ▶ The trainer come with target boards if needed during the practical activities (and bring them back at the end of the course).
- Downloadable preconfigured virtual machine for post-course practical activities
- At the start of each session the trainer will interact with the trainees to ensure the course fits their expectations and correct if needed

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Course Outline

First Day

Linux overview

- Linux history
- Linux architecture and modularity
- Linux system components
- The various licenses used by Linux (GPL, LGPL, etc)

Boot chain

- Low-level boot
 - Boot on NOR
 - Boot on NAND
 - Boot on SD/MMC/eMMC
 - Multistage Boot
 - Why do we need a trusted boot chain
- Security Concerns
 - Confidentiality and Integrity
 - Tampering Prevention
 - Compliance and Certification

Exercise: Boot the platform with the prebuilt image

Secure Boot

- Secure Boot concept
 - The chain of trust
 - Complete secure boot process
- Key Management
 - Introduction to key management
 - Cryptographic algorithms and key types
 - Key storage options: Hardware-based and software-based
 - Key management processes: Generation and revocation of keys
 - ARM-based platforms hardware features overview
 - Secure Monitor
 - Secure World
 - Trusted Execution Environment
 - Secure Boot on RISC-V and X86_64
 - Cryptographic Accelerators
- Software Solutions
 - Open source
 - Proprietary

Exercise: Generate keys that are going to be used for platform encryption

First and Second Stage Bootloaders

- U-Boot
 - Capabilities and features
 - Configuration, customization, and compilation
 - U-Boot SPL as First-Stage Boot Loader (SSBL)
 - Role of u-boot in the trusted boot chain
 - How U-Boot verifies the authenticity of the images it loads

- Configuration options for securing the boot process
- Interaction with the secure world and Trusted Execution Environment
- Signing U-boot
- Arm Trusted Firmware (ATF)
 - Overview and features
 - ATF Boot flow
 - Services
 - Build and deploy
- Other platform specific components

Exercise: Build and boot the platform with U-boot as FSBL and SSBL

Exercise: Build and Boot the platform with ATF as FSBL and U-boot as SSBL

Secured Linux Image

- Introduction to Linux kernel
 - Source code
 - Configuration
 - Compilation
- FIT (Flattened Image Tree) Image
 - What is FIT and why is it used
 - Advantages of using FIT image
 - Configuration
 - Building a Secure FIT Image
- Kernel Configuration for a Secure Linux Platform
 - Configuration options for secure boot in the Linux kernel
 - Access Control Configuration overview

Exercise: Create a secured FIT Linux image

Second Day

Security Considerations when Creating a Root Filesystem

- Tips for hardening and securing a rootfs
 - Minimizing the rootfs
 - Strong authentication
 - Keep software updated
 - Using initramfs
- Read-only root filesystem
 - Introduction to read-only root filesystem
 - Purpose and benefits
 - Overview of the different solutions available
 - SquashFS
 - CramFS: Small memory footprint
 - OverlayFS-based read-only root filesystem
 - UnionFS-based read-only root filesystem
- Considerations when choosing a read-only root filesystem solution
 - Evaluation based on use case, security, performance, and compatibility
- Encrypting Update Images
 - Securely update Linux platform using Mender

Exercise: Create a read-only file system using SquashFS

Open Portable Trusted Execution Environment (OP-TEE)

- Introduction to OP-TEE
- Key Features
- Hardware, software, and firmware requirements
- Architecture of OP-TEE

- Components, modules, and communication channels
- Use Cases
 - Secure storage
 - Secure communication
 - Secure execution of applications
- OP-TEE build and deployment
 - Setting up the environment
 - Configuration of OP-TEE
 - Compilation of OP-TEE
- Comparison to other TEE solutions
- Trusted Applications (TA) on OP-TEE
 - The role of a TA in a secure system
 - Writing a Trusted Application
 - Loading and executing a Trusted Application within the OP-TEE runtime
 - Debugging and testing Trusted Applications
 - Communication between Trusted Applications and normal world applications
 - Best practices for creating secure Trusted Applications

Exercise: Build and install OP-TEE

Exercise: Write a TA application that communicates with a normal world application