Arm Boot Chain and Secure Boot Training: U-Boot, TF-A

From the ROM code to the kernel: the boot chain of an Arm board, integrated, ported and secured stage by stage.

4 days	Intermediate to advanced	On site or remote	English or French
Duration	Level	Format	Language

Overview

This course follows the boot chain of an Arm board from the ROM code to the kernel, integrates it on a new board and secures it from end to end. It covers the first loaders, the secure world with Trusted Firmware-A and OP-TEE, the bootloader that starts Linux, the image formats that carry them, the port to custom hardware, and a full day on secure boot: authentication by the ROM code, signed images and a locked-down bootloader.

The examples use U-Boot, TF-A and OP-TEE on i.MX8M Mini, STM32MP15 and TI AM64x SoCs, and the concepts apply to other bootloaders as well. For x86 platforms, the x86 boot chain and secure boot course covers UEFI, the boot loaders and unified kernel images.

What You Will Learn

- Explain who loads what, from where and at which exception level on a 64-bit Arm SoC
- Integrate TF-A, OP-TEE and U-Boot into a working boot chain
- Assemble a complete boot image at the offsets the ROM code expects
- Port the boot chain to a custom board
- Secure the chain: ROM authentication, signed FIT images, FIP trusted boot and a locked-down bootloader
- Debug a board that does not boot

Who should attend

BSP engineers who bring up new boards, and developers who maintain the boot firmware of a product or need to secure it.

Prerequisites

Good knowledge of embedded Linux and of C. Notions of Arm processors help.

Program

The first three days follow the chain from the ROM code to a ported board, the fourth one secures it.

Day 1

- The boot chain of an embedded Linux system: ROM code and the successive stages
- The Arm secure world: TrustZone, Trusted Firmware-A and OP-TEE
- Getting and building U-Boot
- The bootloader shell: inspecting the board, storage, network and scripting

Day 2

- The environment: variables, storage and access from Linux
- Booting Linux by hand: what the kernel expects and the device tree it receives
- The init sequence of the bootloader and its relocation
- The SPL: role, constraints and how it finds the next stage

Day 3

- The device tree in the bootloader and the fixups applied to the kernel device tree
- The driver model
- Assembling boot images with binman
- Adding support for a new board, from the first instructions to a finished port
- Standard boot and UEFI boot

Day 4

- Secure boot principles and the authentication of the first loader by the ROM code, with keys in fuses
- FIT images and verified boot with signed configurations
- FIP images and the trusted board boot of TF-A
- Locking the bootloader down: environment, console and boot commands
- Secure boot in a Yocto build, and its pitfalls
- Debugging a board that does not boot

Labs and Hardware

Labs run on QEMU machines and on the boards of our lab: Arm 32-bit: STM32MP157C-DK2, BeagleBone Black; Arm 64-bit: Raspberry Pi 4 and 5, NXP i.MX8M Mini, Khadas VIM3; x86: QEMU x86, Generic x86_64 machines, Intel development kits. Other boards, or the hardware of the customer, can be used on request.

Sessions take place on site, in France, Belgium, Switzerland, Germany and beyond, with the hardware provided by LABCSMART, or remotely. Remote attendees work on our build server, each with their own SSH account and workspace, and run the labs on QEMU machines and on real boards. The exercises are drawn from our own client projects. Courses are given in English or French.

Optional Modules

These modules extend the course for teams that develop the bootloader itself.

- Writing commands, logging and coding conventions (half a day)
- Debugging with a probe, the test framework and contributing upstream (1 day)

Who Teaches This Course

The course is designed and given by the founder of LABCSMART, an embedded Linux and Linux kernel engineer with more than eleven years of industrial experience. He contributes upstream to the Linux kernel for Renesas RZ SoCs, in the PCIe, audio, Ethernet, thermal, DMA and clock subsystems, and wrote two books published by Packt, *Linux Device Driver Development, Second Edition* (2022) and *Mastering Linux Device Driver Development* (2021), presented on our publications page.

Plan a session for your team. Every course is made of independent modules. A team can drop the modules it does not need and add others in their place, and we also build courses on demand around a given SoC, board or project. Contact: contact@labcsmart.com, https://labcsmart.com/company/contact/. Course page: https://labcsmart.com/training/embedded-linux-boot-chain/