# Embedded Linux Security

## Course duration

**3** days – 24 hours

## Language

| | |
|---|---|
| Materials | English |
| Oral Lecture | English<br>French |

## Trainer

One of the following engineers

- Mathieu Dubois-Briand
- Olivier Benjamin

## Contact


training@bootlin.com

+33 484 258 097


bootlin

bootlin.com

Onsite training

## Audience

Companies and engineers who design, develop and produce embedded Linux systems that have cyber-security requirements.

## Training objectives

- Be able to understand the security and isolation mechanisms of modern embedded Linux systems: non-executable memory, address space randomization, privilege levels, mandatory and discretionary access control.
- Be able to design and implement a secure boot chain, from the bootloader all the way to userspace, including dm-verity.
- Be able to reason about cryptography, and implement secure key storage, using hardware or software HSM.
- Be able to leverage ARM TrustZone technology (secure world, TF-A, OP-TEE) to isolate sensitive operations and develop Trusted Applications.
- Be able to implement userland security mechanisms: Linux capabilities, namespaces, cgroups, SECCOMP, SELinux, and systemd hardening features.
- Be able to set up filesystem encryption using dm-crypt and integrate it with secure key management through PKCS#11.
- Be able to manage software vulnerabilities using CVE databases, generate and analyze Software Bills of Materials (SBoM), and understand compliance requirements including the Cyber Resilience Act.
- Be able to design and deploy secure update mechanisms using A/B partitioning schemes and tools like RAUC or SWUpdate.
- Be able to understand measured boot concepts and implement platform integrity verification using TPMs and IMA/EVM.

## Prerequisites

- **Knowledge and practice of UNIX or GNU/Linux commands**: participants must be familiar with the Linux command line. Participants lacking experience on this topic should get trained by themselves, for example with our freely available on-line slides.
- **Minimal experience in embedded Linux development**: participants should have a minimal understanding of the architecture of embedded Linux systems: role of the Linux kernel vs. user-space, development of Linux user-space applications in C. Following Bootlin's Embedded Linux course allows to fulfill this pre-requisite.
- **Minimal English language level: B1**, according to the *Common European Framework of References for Languages*, for our sessions in English. See the CEFR grid for self-evaluation.

## Pedagogics

- Lectures delivered by the trainer: 40% of the duration
- Practical labs done by participants: 60% of the duration
- Electronic copies of presentations, lab instructions and data files. They are freely available here.

## Certificate

Only the participants who have attended all training sessions, and who have scored over 50% of correct answers at the final evaluation will receive a training certificate from Bootlin.

## Disabilities

Participants with disabilities who have special needs are invited to contact us at *training@bootlin.com* to discuss adaptations to the training course.

## Required equipement

For on-site session delivered at our customer location, our customer must provide:

- Video projector
- One PC computer on each desk (for one or two persons) with at least 16 GB of RAM, and Ubuntu Linux 24.04 installed in a free partition of at least 30 GB
- Distributions other than Ubuntu Linux 24.04 are not supported, and using Linux in a virtual machine is not supported.
- Unfiltered and fast connection to Internet: at least 50 Mbit/s of download bandwidth, and no filtering of web sites or protocols.
- PC computers with valuable data must be backed up before being used in our sessions.

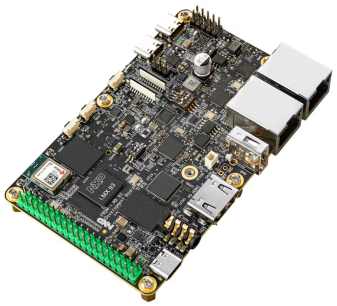
For on-site sessions organized at Bootlin premises, Bootlin provides all the necessary equipment.

## Hardware platform for practical labs

### NXP i.MX93 FRDM

**NXP FRDM-IMX93** development board

- NXP i.MX93 SoC (Dual Cortex-A55 + Cortex-M33)
- 2 GB LPDDR4X, 32 GB eMMC
- Dual Gigabit Ethernet
- USB 2.0 Type-C + USB Type-A
- CAN interface
- MicroSD slot, EEPROM
- Wi-Fi 6 + Bluetooth 5.4 + 802.15.4 (MAYA-W276)
- HDMI output (via LVDS), MIPI DSI and CSI
- Audio jack (MQS), buttons and LEDs
- SWD and UART debug



# Training Schedule

## Day 1 - Morning

| | | |
|---|---|---|
| Lecture | Fundamental concepts | ▪ Threat modeling<br>▪ Cryptography basics: public/private key<br>▪ Understanding a TLS handshake from start to finish<br>▪ Understanding Public Key Infrastructures |
| Lab | Setting up a small PKI | ▪ Using OpenSSL to generate a key hierarchy<br>▪ Experimenting with key revocation<br>▪ Performance comparison of public/private key encryption |
| Lecture | Harware-enforced security barriers | ▪ General security concepts: kernel/user split, CPL, memory protections<br>▪ Common mitigations: ASLR, NX/DEP, SMAP/SMEP, RELRO<br>▪ ARM features: Exception levels<br>▪ ARM features: Secure world, TF-A, OP-TEE<br>▪ Understanding a full ARMv8 bootflow, including secure world<br>▪ Understanding Trusted Applications |

## Day 1 - Afternoon

| | | |
|---|---|---|
| Lab | Exploring the secure world | ▪ Building secure world software (TF-A, OP-TEE, TA)<br>▪ Adding logs to observe Exception level transitions<br>▪ Adding logs to observe world transitions<br>▪ Writing a small Trusted Application in OP-TEE<br>▪ Interacting with our Trusted Application from userland |
| Lecture | Secure boot: part 1 | ▪ Understanding the secure boot chain concept<br>▪ Secure boot implementation examples: x86 and ARM SoCs<br>▪ Secure boot: threat model<br>▪ Be able to design and implement a secure boot chain |
| Lab | Secure boot on i.MX93: ROM stage | ▪ Building all software components for secure boot (AHAB) on i.MX93<br>▪ Creating a signed AHAB container using SPSDK<br>▪ Flashing the fuses on the i.MX93 to enable secure boot (AHAB)<br>▪ Testing that secure boot (AHAB) is correctly enabled |

## Day 2 - Morning

| | | |
|---|---|---|
| Lecture | Secure boot: part 2 | ▪ Deep dive into the U-Boot components: TPL/SPL, U-Boot proper, FIT images<br>▪ Hardware description: Device Tree<br>▪ Enabling signature verification in the U-Boot bootloader<br>▪ RootFS verification: *dm-verity* and implications for system design |
| Lab | Secure boot on i.MX93: boot-loader stage | ▪ Integrating the public key into the DTB<br>▪ Adding signature to the kernel FIT image<br>▪ Configuring U-Boot to enforce signature verification<br>▪ (optional) Configuring SPL to also enforce signature verification |

## Day 2 - Afternoon

| | | |
|---|---|---|
| Lecture | Confidential information | ▪ Filesystem encryption, presentation of dm-crypt/cryptsetup<br>▪ Key management: Hardware Security Module<br>▪ Secure key usage: Introduction to PKCS#11<br>▪ Secure key usage: kernel keyring and trusted keys |

| | | |
|---|---|---|
| Lab | Secure key management | ▪ Provisioning keys into the i.MX93 ELE<br>▪ Using OP-TEE as software HSM<br>▪ (optional) Filesystem encryption using the ELE key<br>▪ (optional) Signing U-Boot FITs with HSM integration over PKCS#11 |
| Lecture | Userland security measures | ▪ Access Control paradigms: MAC/DAC<br>▪ Linux capabilities: usage and examples<br>▪ Process isolation: Namespaces, Cgroups, SECCOMP<br>▪ Linux Security Modules: SELinux, AppArmor<br>▪ Application hardening via *systemd* |

## Day 3 - Morning

| | | |
|---|---|---|
| Lab | Restricting userland applications | ▪ Preventing a simple application from executing shellcode using SECCOMP<br>▪ Manipulating SELinux contexts<br>▪ Using systemd to restrict a daemon's access to resources |
| Lecture | Measured Boot | ▪ Boot event logs<br>▪ TPM's role in platform integrity<br>▪ Introduction to IMA/EVM |
| Lecture | Maintenance, regulations and compliance | ▪ Introduction to vulnerability frameworks: CVE, CWE, CVSS<br>▪ A walkthrough of the Cyber Resilience Act (CRA)<br>▪ Defining an upgrade strategy, based on the release and maintenance cycles of important open-source projects<br>▪ Generating a Software Bill of Materials<br>▪ Analyzing a Software Bill of Materials |
| Lab | Getting familiar with SBoMs and CVEs | ▪ Generating an SBoM for a Yocto distribution<br>▪ Analyzing the generated SBoM using `sbom-cve-check`<br>▪ Patching a CVE present in the SBoM<br>▪ Differential analysis |

## Day 3 - Afternoon

| | | |
|---|---|---|
| Lecture | Secure updates | ▪ A/B updates<br>▪ Presentation of SWUpdate<br>▪ Presentation of RAUC<br>▪ Bootloader integration<br>▪ Consequences on the secure boot chain |
| Lab | Setting up RAUC for secure updates | ▪ RAUC configuration on the target<br>▪ Building, shipping and installing a RAUC bundle<br>▪ Bundle signature verification using PKCS#11<br>▪ (optional) RAUC bundle encryption<br>▪ (bonus) HSM integration |