Embedded Linux Security

Online seminar

Course duration

4 half days – 16 hours

Language

Materials	English
Oral Lecture	English French

Trainer

One of the following engineers

- Mathieu Dubois-Briand
- Olivier Benjamin

Contact

training@bootlin.com

+33 484 258 097

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, over video-conference. Participants can ask questions at any time.
- Practical demonstrations done by the trainer, based on practical labs, over video-conference. Participants can ask questions at any time. Optionally, participants who have access to the hardware accessories can reproduce the practical labs by themselves.
- Instant messaging for questions between sessions (replies under 24h, outside of week-ends and bank holidays).
- 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

Mandatory equipment:

- Computer with the operating system of your choice, with the Google Chrome or Chromium browser for videoconferencing.
- Webcam and microphone (preferably from an audio headset).
- High speed access to the Internet.

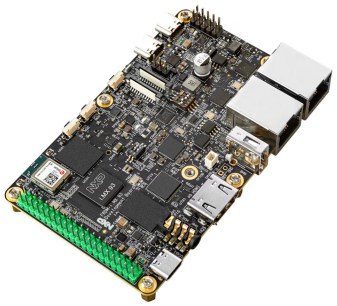
Optionnally, if the participants want to be able to reproduce the practical labs by themselves, they must separately purchase the hardware platform and accessories, and must have a PC computer with a native installation of Ubuntu Linux 24.04.

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

Half day 1

Lecture	Fundamental concepts	▪ Threat modeling ▪ Cryptography basics: public/private key ▪ Understanding a TLS handshake from start to finish ▪ Understanding Public Key Infrastructures
Demo	Setting up a small PKI	▪ Using OpenSSL to generate a key hierarchy ▪ Experimenting with key revocation ▪ Performance comparison of public/private key encryption
Lecture	Harware-enforced security barriers	▪ General security concepts: kernel/user split, CPL, memory protections ▪ Common mitigations: ASLR, NX/DEP, SMAP/SMEP, RELRO ▪ ARM features: Exception levels ▪ ARM features: Secure world, TF-A, OP-TEE ▪ Understanding a full ARMv8 bootflow, including secure world ▪ Understanding Trusted Applications
Demo	Exploring the secure world	▪ Building secure world software (TF-A, OP-TEE, TA) ▪ Adding logs to observe Exception level transitions ▪ Adding logs to observe world transitions ▪ Writing a small Trusted Application in OP-TEE ▪ Interacting with our Trusted Application from userland

Half day 2

Lecture	Secure boot: part 1	▪ Understanding the secure boot chain concept ▪ Secure boot implementation examples: x86 and ARM SoCs ▪ Secure boot: threat model ▪ Be able to design and implement a secure boot chain
Demo	Secure boot on i.MX93: ROM stage	▪ Building all software components for secure boot (AHAB) on i.MX93 ▪ Creating a signed AHAB container using SPSDK ▪ Flashing the fuses on the i.MX93 to enable secure boot (AHAB) ▪ Testing that secure boot (AHAB) is correctly enabled
Lecture	Secure boot: part 2	▪ Deep dive into the U-Boot components: TPL/SPL, U-Boot proper, FIT images ▪ Hardware description: Device Tree ▪ Enabling signature verification in the U-Boot bootloader ▪ RootFS verification: *dm-verity* and implications for system design
Demo	Secure boot on i.MX93: boot-loader stage	▪ Integrating the public key into the DTB ▪ Adding signature to the kernel FIT image ▪ Configuring U-Boot to enforce signature verification ▪ (optional) Configuring SPL to also enforce signature verification

Half day 3

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

Demo	Secure key management	▪ Provisioning keys into the i.MX93 ELE ▪ Using OP-TEE as software HSM ▪ (optional) Filesystem encryption using the ELE key ▪ (optional) Signing U-Boot FITs with HSM integration over PKCS#11
Lecture	Userland security measures	▪ Access Control paradigms: MAC/DAC ▪ Linux capabilities: usage and examples ▪ Process isolation: Namespaces, Cgroups, SECCOMP ▪ Linux Security Modules: SELinux, AppArmor ▪ Application hardening via *systemd*
Demo	Restricting userland applications	▪ Preventing a simple application from executing shellcode using SECCOMP ▪ Manipulating SELinux contexts ▪ Using systemd to restrict a daemon's access to resources

Half day 4

Lecture	Maintenance, regulations and compliance	▪ Introduction to vulnerability frameworks: CVE, CWE, CVSS ▪ A walkthrough of the Cyber Resilience Act (CRA) ▪ Defining an upgrade strategy, based on the release and maintenance cycles of important open-source projects ▪ Generating a Software Bill of Materials ▪ Analyzing a Software Bill of Materials
Demo	Getting familiar with SBoMs and CVEs	▪ Generating an SBoM for a Yocto distribution ▪ Analyzing the generated SBoM using `sbom-cve-check` ▪ Patching a CVE present in the SBoM ▪ Differential analysis
Lecture	Secure updates	▪ A/B updates ▪ Presentation of SWUpdate ▪ Presentation of RAUC ▪ Bootloader integration ▪ Consequences on the secure boot chain
Demo	Setting up RAUC for secure updates	▪ RAUC configuration on the target ▪ Building, shipping and installing a RAUC bundle ▪ Bundle signature verification using PKCS#11 ▪ (optional) RAUC bundle encryption ▪ (bonus) HSM integration