FOSDEM

IOT BZH

Lessons learned from deploying boot security features on embedded Linux systems

Brussels - 1st & 2nd February, 2025

Johann Gautier
Linux Embedded Engineer

Valentin Geffroy
Linux Embedded Engineer

IoT.bzh at a glance

Our location
Brittany

European **C**yber**S**ecurity **O**rganisation:
Cyber Valleys mapping

30 years of embedded OS
Wind River (1990) - Intel (2009) - IoT.bzh (2015)

VxWorks®
WIND RIVER

Open Source contributions

TIZEN
SAMSUNG

OS open source, Samsung TVs
Intel Vannes (2011-2015)

AUTOMOTIVE GRADE LINUX
THE LINUX FOUNDATION

Open Source OS for Toyota, Suzuki, Subaru
IoT.bzh: +50% technical contributions 2016-2020

Our product
redpesk®: SaaS platform (or On Prem) Linux for industrial IoT (auto, mil-aero, energy...)

redpesk®

Some partners

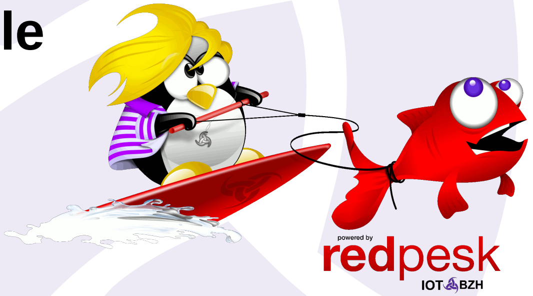
Valeo
IDEMIA
RENESAS
SAFRAN
STELLANTIS
TotalEnergies
THALES
BENETEAU
PIRIOU
THE LINUX FOUNDATION

Cybersecurity in embedded context

- **Surface of attack**
  - Bypassing security rules (gain elevation access for critical features)
  - Entry point for hackers: debug ports, unsafe authentication...
  - Software vulnerabilities: user librairies, main OS packages (CVEs)
- **Already effective rules, a *lot more* are coming!**
  - CYBER RESILIENCE ACT: penalities for manufacturers who have not notify the relevant authorities about exploitable and vulnerabilities
  - Specific automotive standards: ISO/SAE 21434 (Road Vehicles), ETSI EN 303 645 (IoT Devices), ISA/IEC 62443 (Industrial Automation)...
  - All **these rules are (or will be) mandatory** for embedded market

Addressing Risks

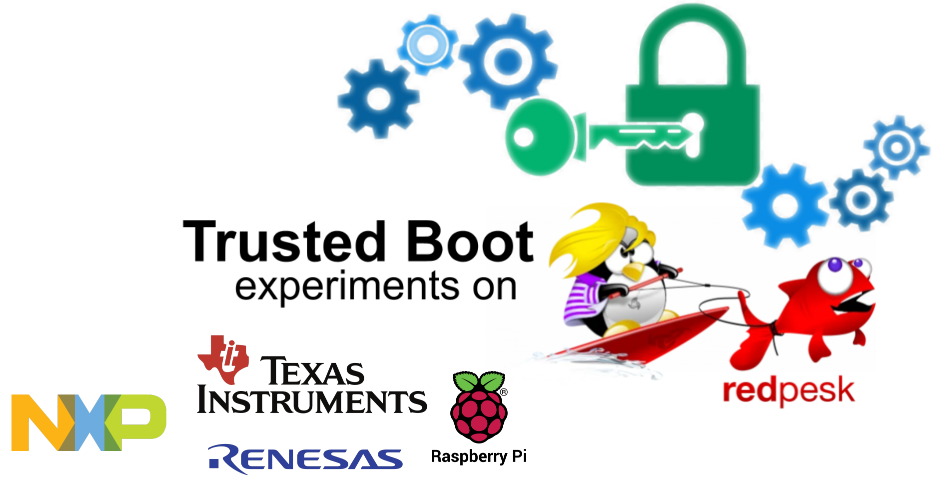
- **Assert run the right code with the right permission**
  - **Secure boot + TPM + Fuse master key**
  - **Check signature on all installed software component**
  - **Encrypt everything that should be (access code, data partition…)**
  - Systemic activation of MAC+DAC+Namespace+Cgroups...
- **Full supply chain control from source code to executable**
  - Build under CI/CD factory
  - Automatise SBOM, CVEs, test report, ...
  - Secure the OTA
  - Organize the system to be auditable (log generation, binary reproducibility, ...)

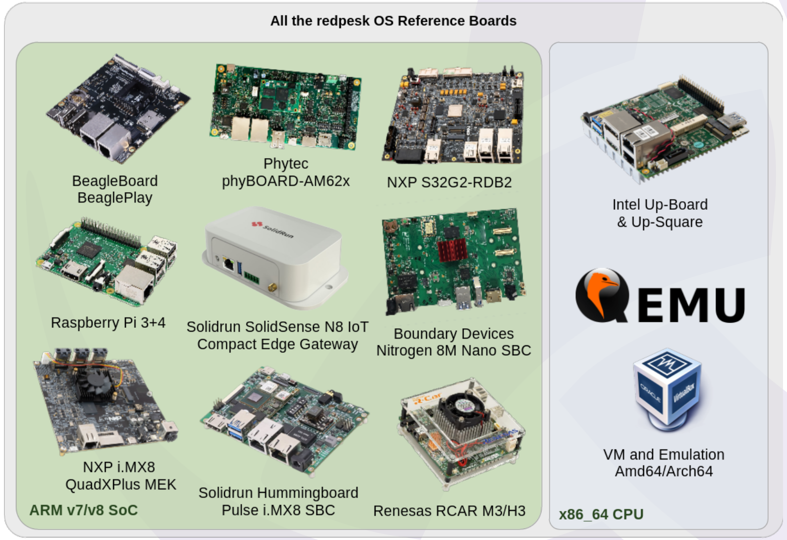


What means boot integrity for us?

General statement for our Linux-based images

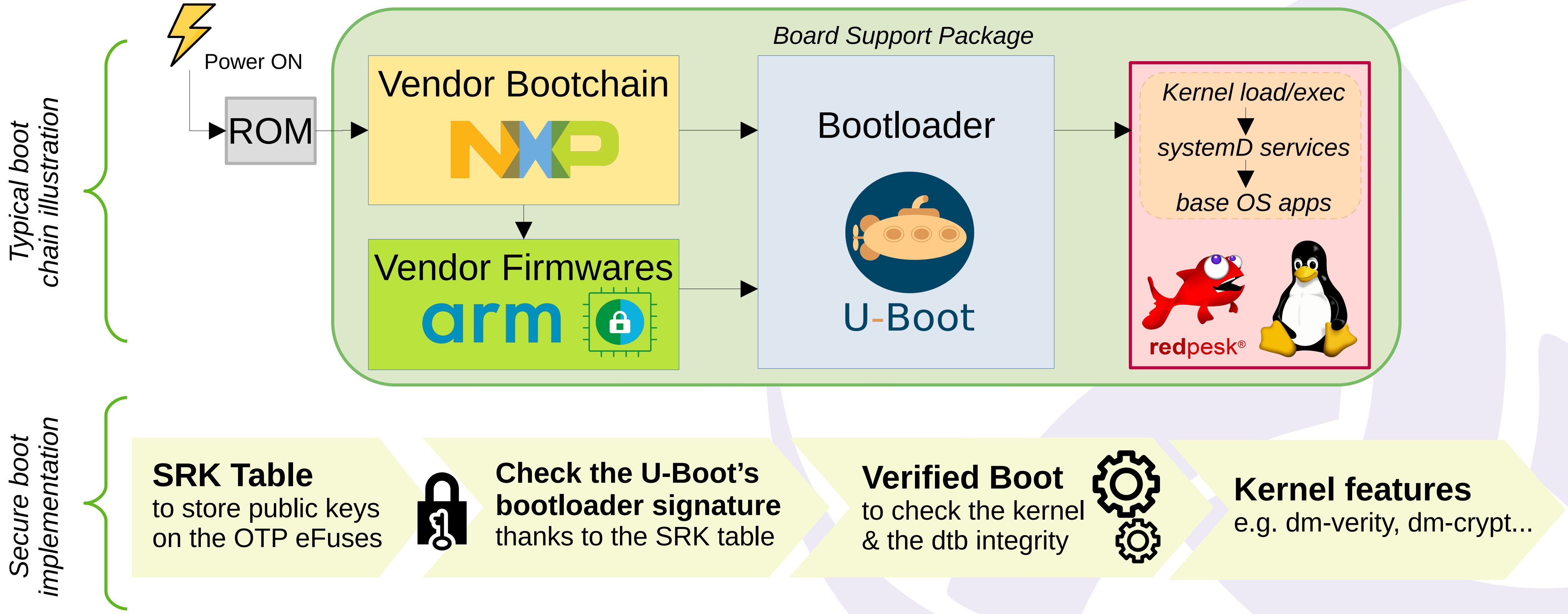
- Available on Intel x86 (64 bits) & ARM aarch (32/64 bits) but others arch too
- Different implementations depending on the embedded board vendor
- **Goal:** each bootflow step is guaranteed and must verify the next in integrity or by a signature process





Securizing Linux bootflow

NXP board (ARM SoC) simplified example



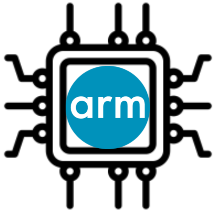
Securizing Linux bootflow

Hands on real production case

Hands on real production case

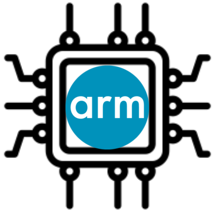
One of our restricted embedded platform

- On top of Secure Boot we address challenges:
  - Full Disk Encryption (FDE)
  - Integrity Check (IC)
  - Heavy hardware constraints (cost)



Platform 1

CPU: 32 bits 1-cores
Freq: @1Ghz
RAM: 1GB
NAND: 512MB



Platform 2

CPU: 64 bits 2-cores
Freq: @1.6Ghz
RAM: 256MB
NAND: 512MB

  - Legal constraints: boot critical services in less than 30 seconds!
  - Legacy constraints: Linux Kernel 3.18 or 4.14 (imposed by SoC vendor)
- Already complex without security
- *Very interesting* challenge with security

Hands on real production case

Lessons learned for Full Disk Encryption (FDE)

- Encryption for *each* board (secrets are stored in HSM/TPM)
- Runtime encryption is required (not possible at build)
- Encryption overhead:
  - At first boot (initial encrypting operation)
  - At runtime (between 20-40% IO throughput)
  - At update (partitions to encrypt again)
- All hardware acceleration must be activated (kernelspace)
- Memory overhead: *dm-crypt* does the job with 15MB

Hands on real production case

Lessons learned for Integrity Check (IC)

- In our case, *dm-verity* costs too much (not respecting our constraints)
- How to do IC without *dm-verity*?
- At boot, when verifying read-only partitions (checksum)...
  - ... boot time is not respecting our constraints :’(
- At runtime, the IC must be done on decrypted data
- The systematic data partition decryption adds an additionnal cost
- **Optimizations are highly required** (IO to optimize, things to do checksums on data partitions because the time is important)

Lessons learned about boot security features

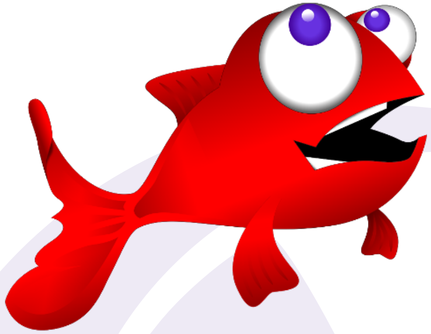
A conclusion for our Linux-based embedded platforms

- Need (a huge need!!) to enforce embedded systems
- Laws, rules and standards are evolving in this way
- Different implementations (SoC vendor)
- Security costs time and performances
- ... so optimization is required!
- Our work is still in progress

Adding secure boot features on restricted platform is CHALLENGING!

For more details,

- **red**pesk®
  - Website: https://redpesk.bzh/
  - Documentation: https://docs.redpesk.bzh/
  - Sources: https://github.com/redpesk/readme
  - Secure Boot: experiments on boards
- **IoT.bzh**
  - Website: https://iot.bzh/
  - Publications: https://iot.bzh/en/publications
- **Community Support**
  - Matrix.org: +redpesk:matrix.org



redpesk®

Q&A

Lorient Harbour, South Brittany, France

redpesk® embedded software for IoT

redpesk OS

- *LTS* version based on RHEL *devel* version based on CentOS Stream
- Support cross-build or emulated-build
- BSP (Board Support Package) allowing to support various embedded boards
- Based on RPM packages
- Enriched by µservices & security frameworks

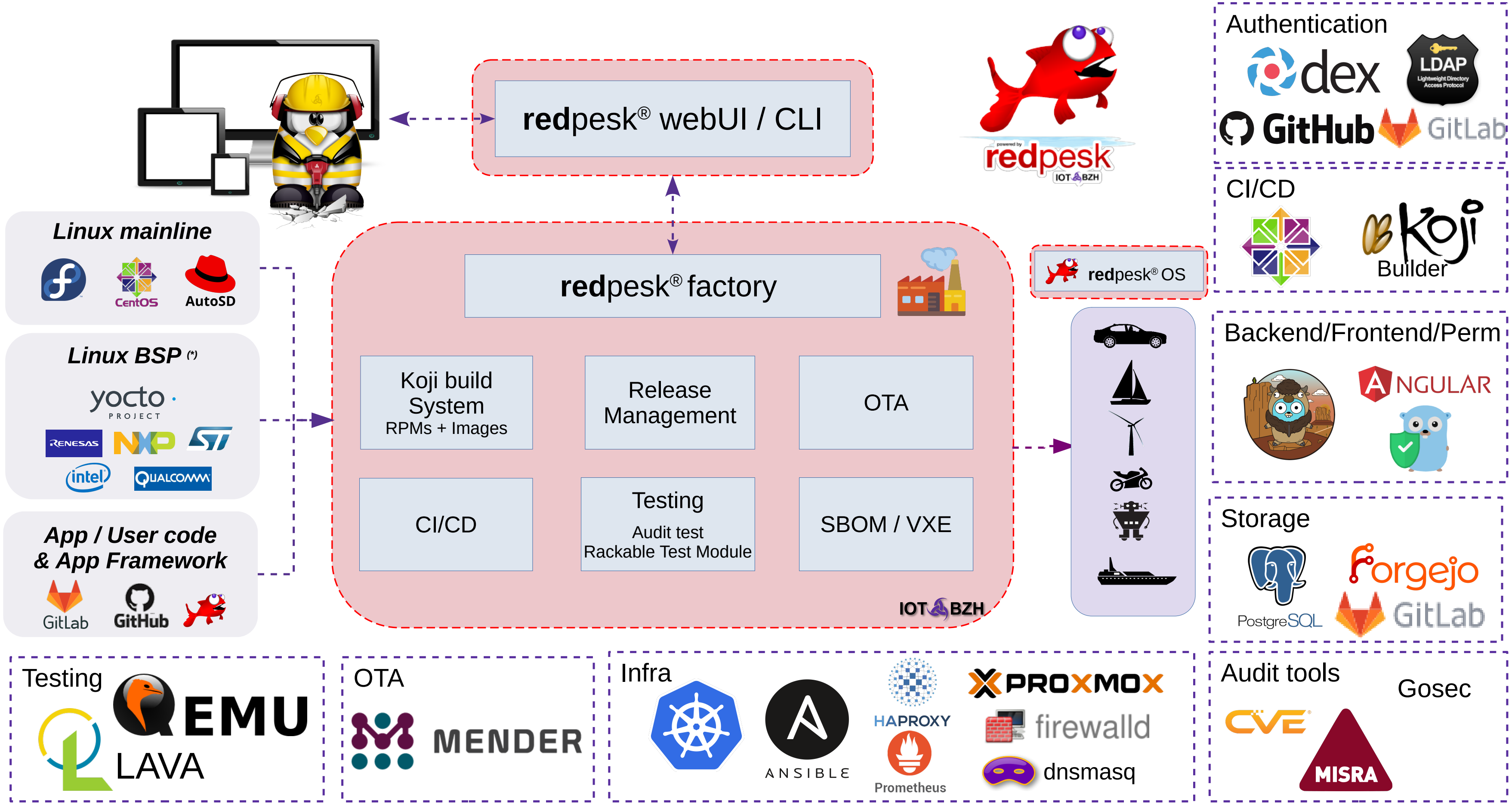
Sources available at
https://github.com/redpesk

redpesk Factory

- Ease development and integration workflows in cross environment
- Design for developers, integrator, QA engineers, delivery managers
- CI/CD: automatic rebuild, testing
- Based on Koji (Fedora build system) with extensions to support cross-building

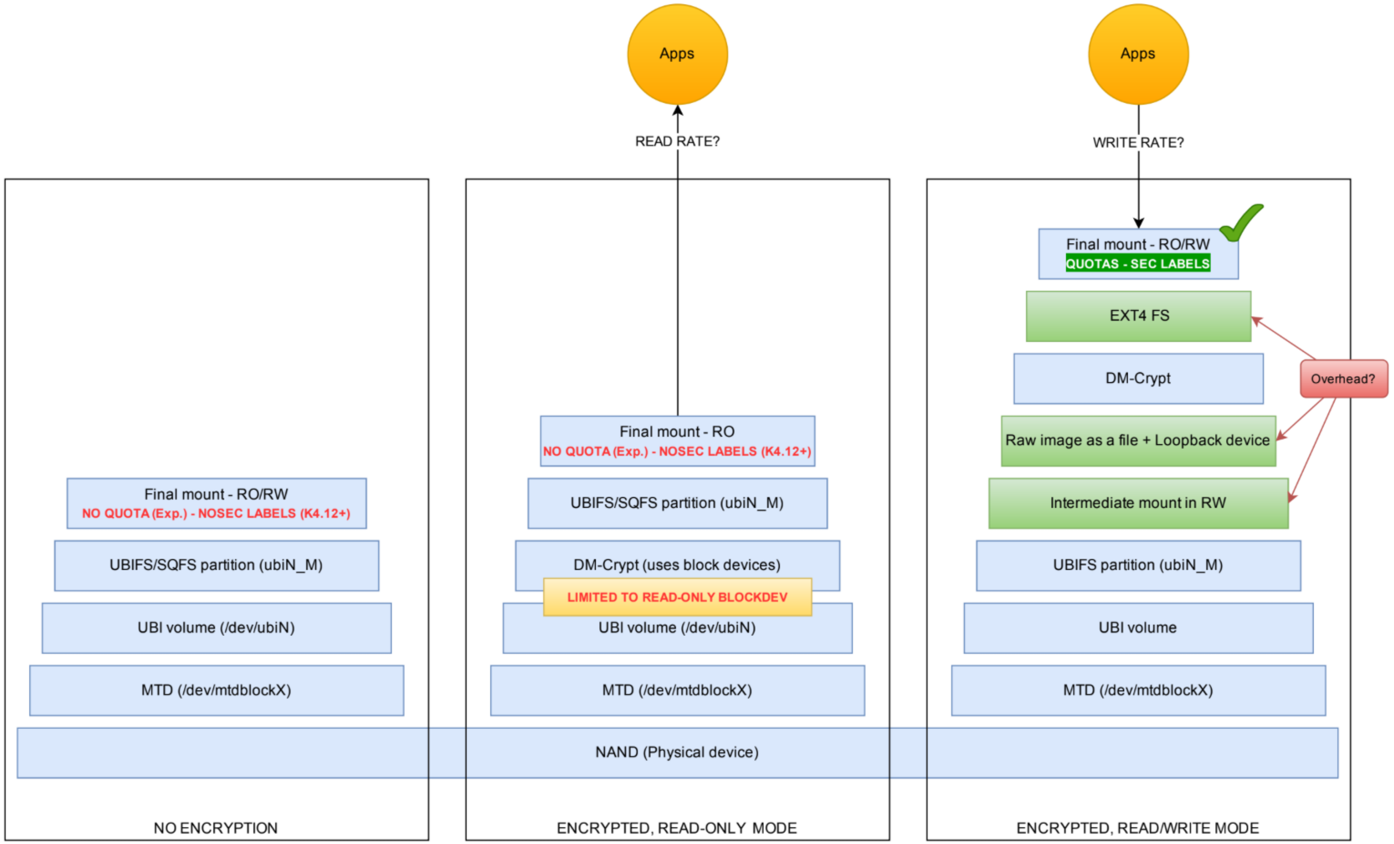
Community edition
https://community-app.redpesk.bzh

redpesk® factory based on proven tools



Hands on real production case

Lessons learned for Full Disk Encryption (FDE)



Hands on real production case

Lessons learned for Integrity Check (IC)