

Manuel Traut

✉ manuel.traut@mt.com

Ⓜ [@manut@moutains.social](https://manut@moutains.social)



Booting an Embedded System Like a PC

All Systems Go!, Berlin, DE, September 25-26 2024

METTLER TOLEDO





Embedded Linux Developer, METTLER TOLEDO

- Concept of our Debian based Embedded Linux distribution
- GitLab CI Pipelines
- Member of our Open Source Focus Group



Laboratory Weighing

Laboratory Balances, Analytical, Precision and Micro & Ultra Micro Balances, Powder & Liquid Dosing, Moisture Analyzers, Test Weights



Industrial Weighing Scales and Systems

Bench Scales, Portable Scales, Floor Scales, Counting and Mixing Scales, Terminals, Weigh Modules, Load Cells, SQC, Software

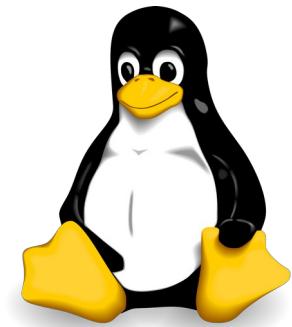
Debian User and Contributor

- User since 2006, contributor since 2022
- Package maintainer of libusbgx
This library is used in some of our scales
- Package maintainer of cozy (audiobook player)
Using it on my PinePhone running Mobian
- Toradex Colibri i.MX6DL Evalboard support in u-boot
It is used for our demos and for integrationtests
- Speaker at Debconf22
Build Debian based Embedded Images in the Cloud



Linux Kernel

- User since ~2000
- Sporadic contributor since 2019
- Maintained PineTab2 Devicetree in 6.9
- Some oneliners



1	Motivation
2	System Architecture
3	Firmware
4	Operating System
5	Conclusion
6	
7	
8	
9	

Further Improve Security of our Debian based Embedded Linux Platform

- Protect our intellectual property
- Only allow **trusted devices** to access cloud services
- Install trusted software **upgrades** (avoid downgrades)
- Allow users to exchange / add software on an **unlocked device**



Similar Requirements are True and Solved for PCs

- ~~Protect our intellectual property~~ Protect users private data



- Only allow trusted ~~devices~~ users to access cloud services



TPM-2 backend, PCR sealing

- Install trusted software upgrades (~~avoid downgrades~~)

[● ◀] **systemd** - sysupdate from signed source, dm-verity

- Allow users to exchange / add software on an unlocked device

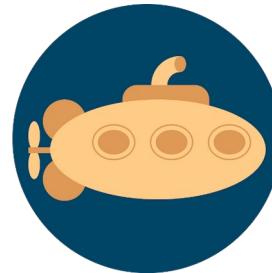


SecureBoot (TM)



...Booting!

- Our devices boot with “Das U-Boot” / distroboot



- A PC boots with UEFI and includes a TPM 2.0



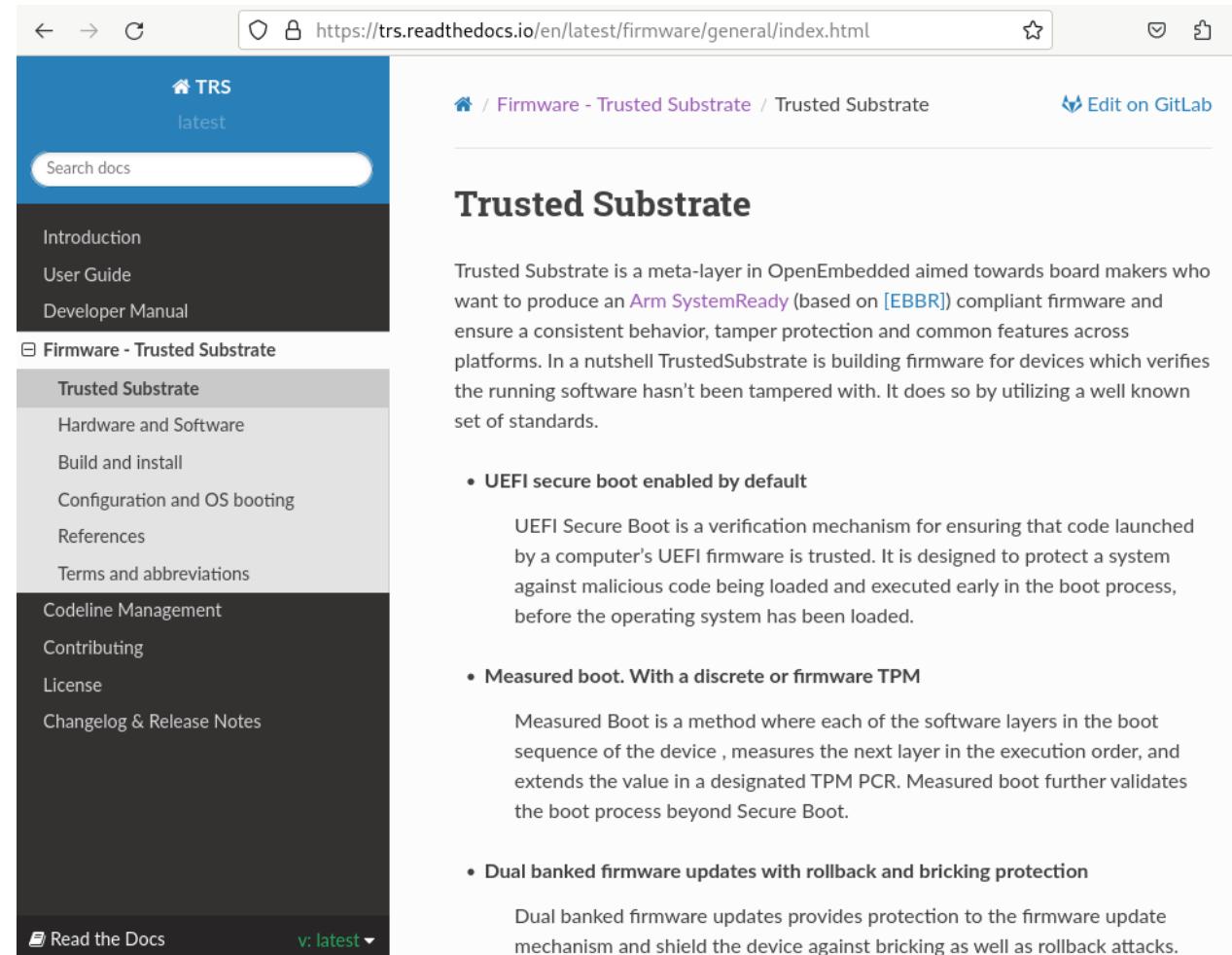
- All needed components to implement UEFI Boot with a TPM2.0 in firmware are available e.g. for Freescale i.MX8MM SoC

- It is “just” an **integration task**

- **“arm SystemReady IR”** is a specification that describes booting via EFI with TPM support

- **Linaro Trusted Substrate Firmware**
Implements this for different platforms

- Documentation is available here:
<https://trs.readthedocs.io/>



The screenshot shows a web browser displaying the Linaro Trusted Substrate Firmware documentation. The URL in the address bar is <https://trs.readthedocs.io/en/latest/firmware/general/index.html>. The page title is "Trusted Substrate". The left sidebar contains a navigation menu with links to "Introduction", "User Guide", "Developer Manual", "Firmware - Trusted Substrate", "Trusted Substrate", "Hardware and Software", "Build and install", "Configuration and OS booting", "References", "Terms and abbreviations", "Codeline Management", "Contributing", "License", and "Changelog & Release Notes". The main content area starts with a section titled "Trusted Substrate" which defines it as a meta-layer in OpenEmbedded aimed towards board makers. It mentions "UEFI secure boot enabled by default", "Measured boot. With a discrete or firmware TPM", and "Dual banked firmware updates with rollback and bricking protection". The footer of the page includes the "Read the Docs" logo and a "v: latest" dropdown.

Trusted Substrate is a meta-layer in OpenEmbedded aimed towards board makers who want to produce an [Arm SystemReady](#) (based on [EBBR](#)) compliant firmware and ensure a consistent behavior, tamper protection and common features across platforms. In a nutshell TrustedSubstrate is building firmware for devices which verifies the running software hasn't been tampered with. It does so by utilizing a well known set of standards.

- **UEFI secure boot enabled by default**

UEFI Secure Boot is a verification mechanism for ensuring that code launched by a computer's UEFI firmware is trusted. It is designed to protect a system against malicious code being loaded and executed early in the boot process, before the operating system has been loaded.

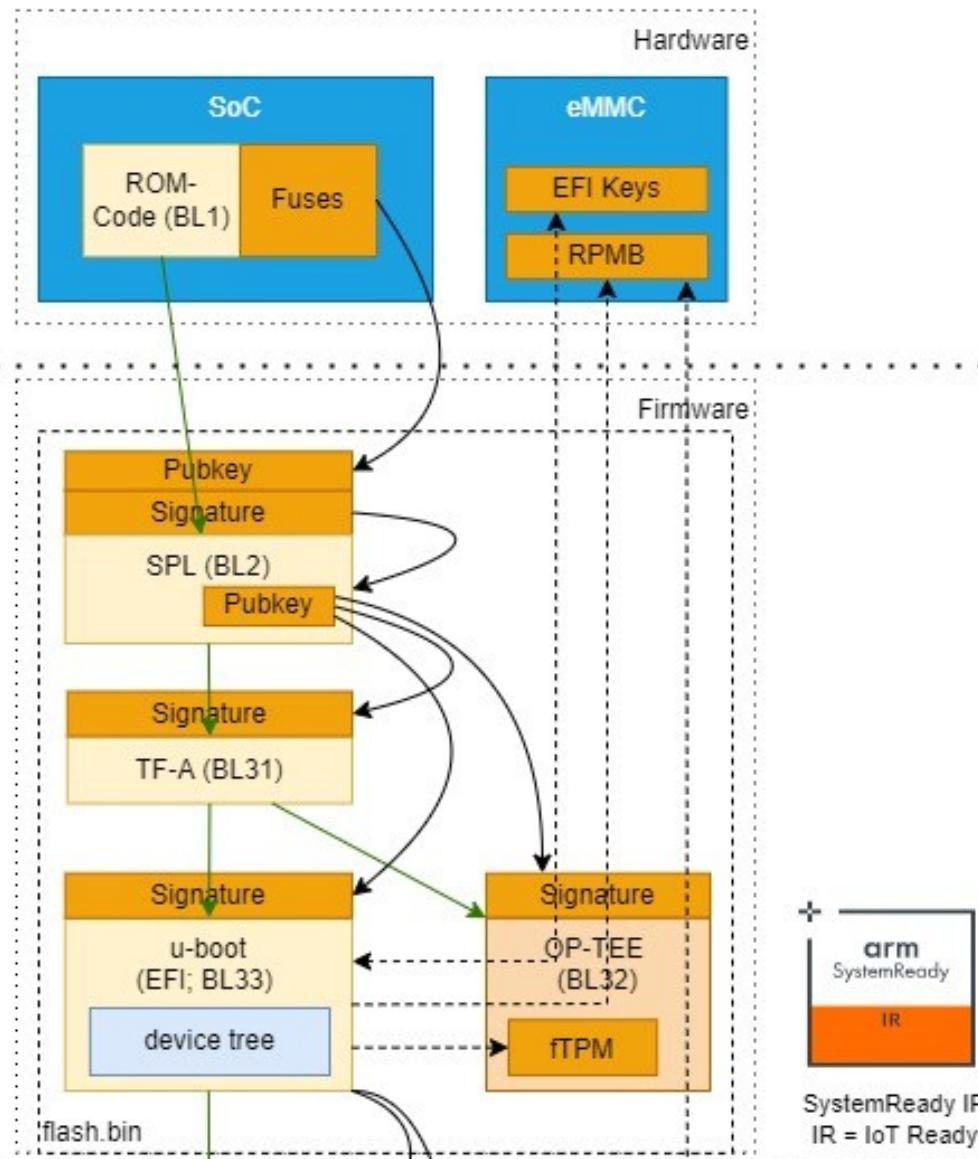
- **Measured boot. With a discrete or firmware TPM**

Measured Boot is a method where each of the software layers in the boot sequence of the device, measures the next layer in the execution order, and extends the value in a designated TPM PCR. Measured boot further validates the boot process beyond Secure Boot.

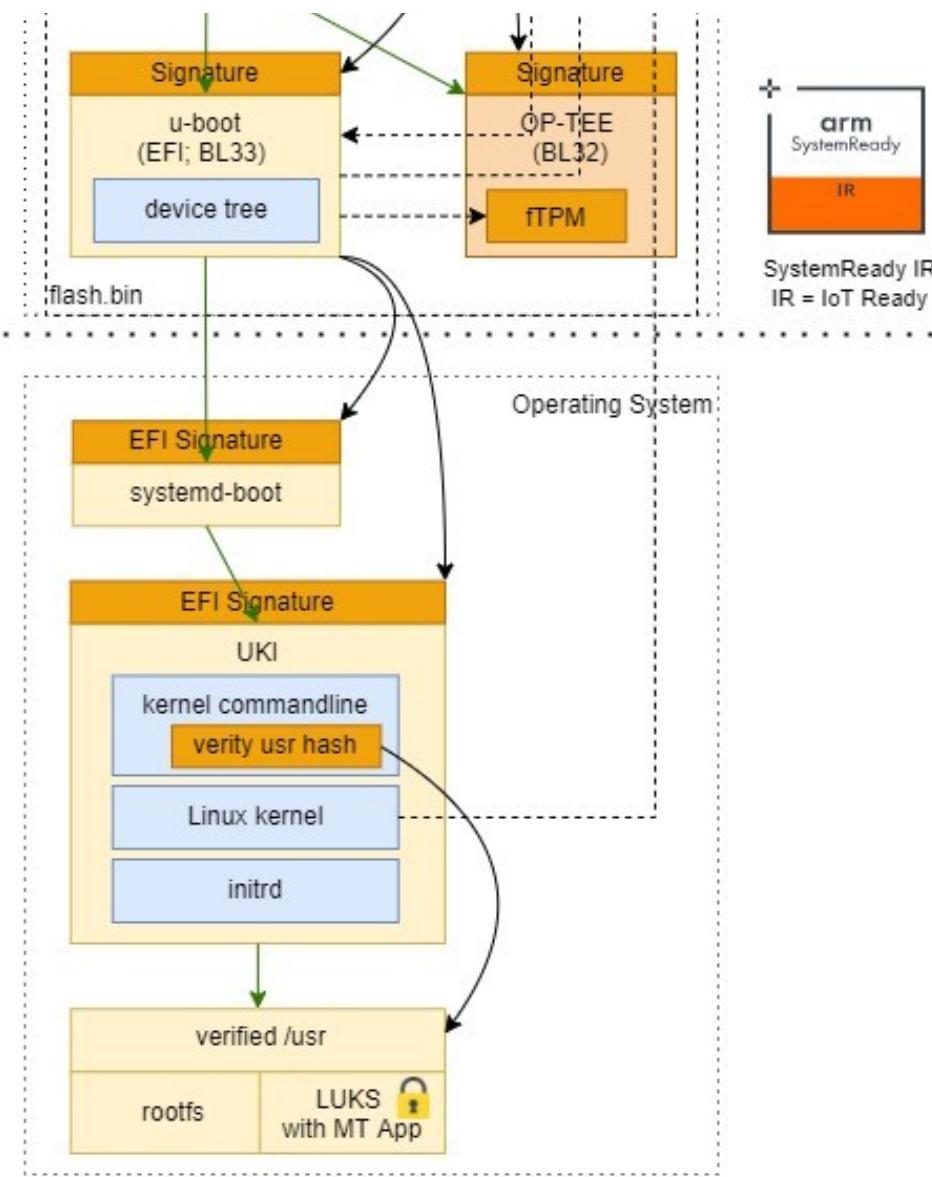
- **Dual banked firmware updates with rollback and bricking protection**

Dual banked firmware updates provides protection to the firmware update mechanism and shield the device against bricking as well as rollback attacks.

System Architecture

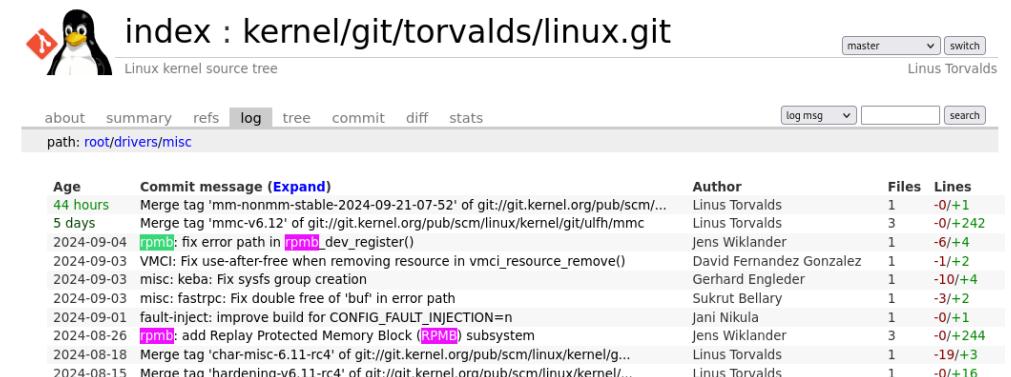


System Architecture



OP-TEE

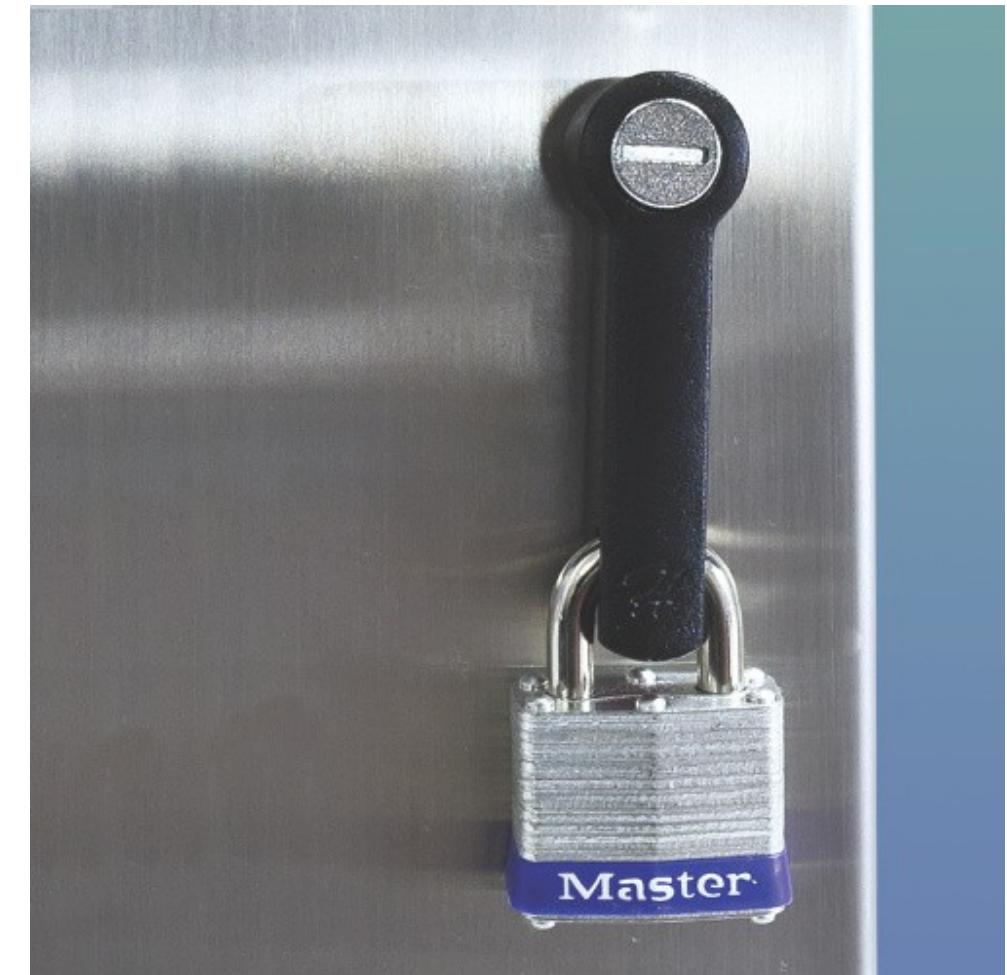
- **fTPM sample application as OP-TEE Early TA (compiled-in)**
 - fTPM driver is in **mainline** U-Boot and mainline Linux
 - From a user perspective no difference to a hardware TPM
 - OP-TEE OS, PR for fTPM as TA from Jens Wiklander
https://github.com/OP-TEE/optee_os/pull/7054
- Non-volatile Storage for fTPM is the **RPMB** block of the **eMMC**
 - eMMC is not available in ARM TrustZone
 - OP-TEE calls non-secure world to handle RPMB requests
 - U-Boot includes code to handle this RPMB requests
 - Linux will have an RPMB subsystem (merged for 6.12):
<https://lore.kernel.org/lkml/20240814153558.708365-1-jens.wiklander@linaro.org/>
 - Currently tee-suplicant needs to do this in userspace



RPMB Key

- RPMB Key is a **Symmetric Key** that is needed to write to the eMMC RPMB block
- RPMB Key can be configured/written to the eMMC only once
- **Key shall not be compiled into software** to avoid extracting/reverse engineering it
- On i.MX8 the RPMB Key can be **derived** from an i.MX **Uniq-Device-Id** that is only readable in TrustZone

The diagram illustrates the RPMB key management process. It shows two main components: the eMMC Device and the Host SoC. The eMMC Device contains a SHA-256 block (labeled 'Secret Key') and an RPMB block. A Counter is also present. The Host SoC contains a SHA-256 block (labeled 'Secret Key') and a Counter. A 'Message' is sent from the Host SoC to the eMMC Device. The eMMC Device performs a MAC operation using its internal SHA-256 block and Counter. The Host SoC also performs a MAC operation using its internal SHA-256 block and Counter. The results of these MAC operations are compared to verify the integrity of the message.



Das U-Boot

- **EFI support is mainline**

- **SecureBoot support is mainline**

- EFI variables are stored in OP-TEE
- EDK II StandAloneMM is linked to OP-TEE

- **EFI Capsule Upgrade support is mainline**

- **Documentation is excellent**

<https://docs.u-boot.org/en/latest/develop/uefi/uefi.html>



The screenshot shows a web browser displaying the official U-Boot documentation at <https://docs.u-boot.org/en/latest/develop/uefi/uefi.html>. The page has a blue header with the U-Boot logo and navigation links for "Build U-Boot", "Learn about U-Boot", and "Use U-Boot". A sidebar on the left is titled "Develop U-Boot" and contains sections for "General", "Implementation" (which is expanded to show "Directory hierarchy", "Blob Lists - bloblist", "Standard Boot", "Continuous Integration testing", "Implementing shell commands", and "U-Boot configuration node"), and "Read the Docs". The main content area is titled "UEFI on U-Boot" and describes the Unified Extensible Firmware Interface Specification (UEFI) as the default for booting on AArch64 and x86 systems, providing a stable API for interacting with drivers and applications. It also mentions the Linux kernel and boot loaders like GRUB or the FreeBSD loader can be executed. Below this, a section titled "Development target" discusses the implementation of UEFI in U-Boot, mentioning requirements from EBBR and SBBRS. A note states that a full-blown UEFI implementation would contradict the U-Boot design principle "keep it small". Finally, a section titled "Building U-Boot for UEFI" explains that UEFI support is for little-endian systems and provides configuration options: `CONFIG_CMD_BOOTEFI=y` and `CONFIG_EFI_LOADER=y`.

UEFI on U-Boot

The Unified Extensible Firmware Interface Specification (UEFI) [1] has become the default for booting on AArch64 and x86 systems. It provides a stable API for the interaction of drivers and applications with the firmware. The API comprises access to block storage, network, and console to name a few. The Linux kernel and boot loaders like GRUB or the FreeBSD loader can be executed.

Development target

The implementation of UEFI in U-Boot strives to reach the requirements described in the “Embedded Base Boot Requirements (EBBR) Specification - Release v2.1.0” [2]. The “Server Base Boot Requirements System Software on ARM Platforms” [3] describes a superset of the EBBR specification and may be used as further reference.

A full blown UEFI implementation would contradict the U-Boot design principle “keep it small”.

Building U-Boot for UEFI

The UEFI standard supports only little-endian systems. The UEFI support can be activated for ARM and x86 by specifying:

```
CONFIG_CMD_BOOTEFI=y  
CONFIG_EFI_LOADER=y
```

Use systemd and mkosi

- Use TPM 2.0 PCR measurements to verify system integrity
systemd-pcrlock
- Deploy LUKS Key to TPM
systemd-cryptenroll, systemd-cryptsetup
- Manage credentials used by services
systemd-creds
- Build and sign OS images with dm-verity and UKIs
mkosi
- Applying A/B OS upgrades generated by mkosi
systemd-sysupdate, systemd-boot, bootctl
- Creating / resizing partitions by deploying an OS update
systemd-repart



We are able to...

✓ Protect IP

Stored in LUKS volumes backed by TPM
LUKS key sealed to PCR of SecureBootMode

✓ Protect certificates used for authentication stored in TPM, sealed to PCR of SecureBootMode

✓ Deploy OS upgrades

File based verification before deployment is missing

✓ Boot latest OS

Bootcounting/fallback is missing

✓ Boot verified operating systems only with EFI SecureBoot enabled

✓ Unlock devices by deploying new EFI Keys

And there are Even More Benefits...

- ✓ The same OS Image boots on many devices
- ✓ Install any Linux distribution



systemd-sysupgrade

```
root@iris:~# /lib/systemd/systemd-sysupdate list
Automatically discovered root block device '/dev/mmcblk1'.
Automatically discovered root block device '/dev/mmcblk1'.
Discovering installed instances...
Discovering available instances...
Determining installed update sets...
Determining available update sets...
  VERSION INSTALLED AVAILABLE
ASSESSMENT
• 1.1      ✓
current
root@iris:~# /lib/systemd/systemd-sysupdate check-new
```

CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.8 | VT102 | Offline | ttyUSB4

Fedora Installer

- File based signature verification in systemd-sysupdate or systemd-boot bootcount logic implementation in RAUC
- Implement FAT “move” in U-Boot (needed by systemd-boot bootcounter)
- Learn from **Enterprise Linux admins**
 - How to
 - Provision new systems?
 - Deploy, update/revoke keys and certificates?
 - How to **maintain** the firmware components?
 - SBOM
 - Security Tracking
 - Updates
- Try EFI Capsule Upgrade
- Use efifb for bootlogo, plymouth



- **RPMB Kernel Patches** “Tested-by”
 - RPMB data was not page-aligned
(Problems with SDHCI controller, DMA, ..)
- **systemd-sysupdate**
 - Report MatchPattern assumption to user
- **mkosi bugreport**
 - ‘login’ is no longer essential in Debian
 - This needed to be mentioned in documentation

Backlog

- **USB Gadget support in u-boot/SPL** for CI UDC Controller
- Minor patches here and there...



- All needed software components are available
- Integration is not an easy task
 - If something does not work,
the complete stack needs to be understood
- Well established tooling can be used
- Well specified and established interface
between firmware and operating system
 - The **operating system** and
upgrade implementation is
not device specific



To Rebuild the Demos

▪ Hardware

- <https://www.toradex.com/products/carrier-board/verdin-development-board-kit>
- <https://www.toradex.com/computer-on-modules/verdin-arm-family/nxp-imx-8m-mini-nano#buynow>

▪ Software

- Build the firmware

<https://github.com/mt-at-mt/build-efi-firmware>

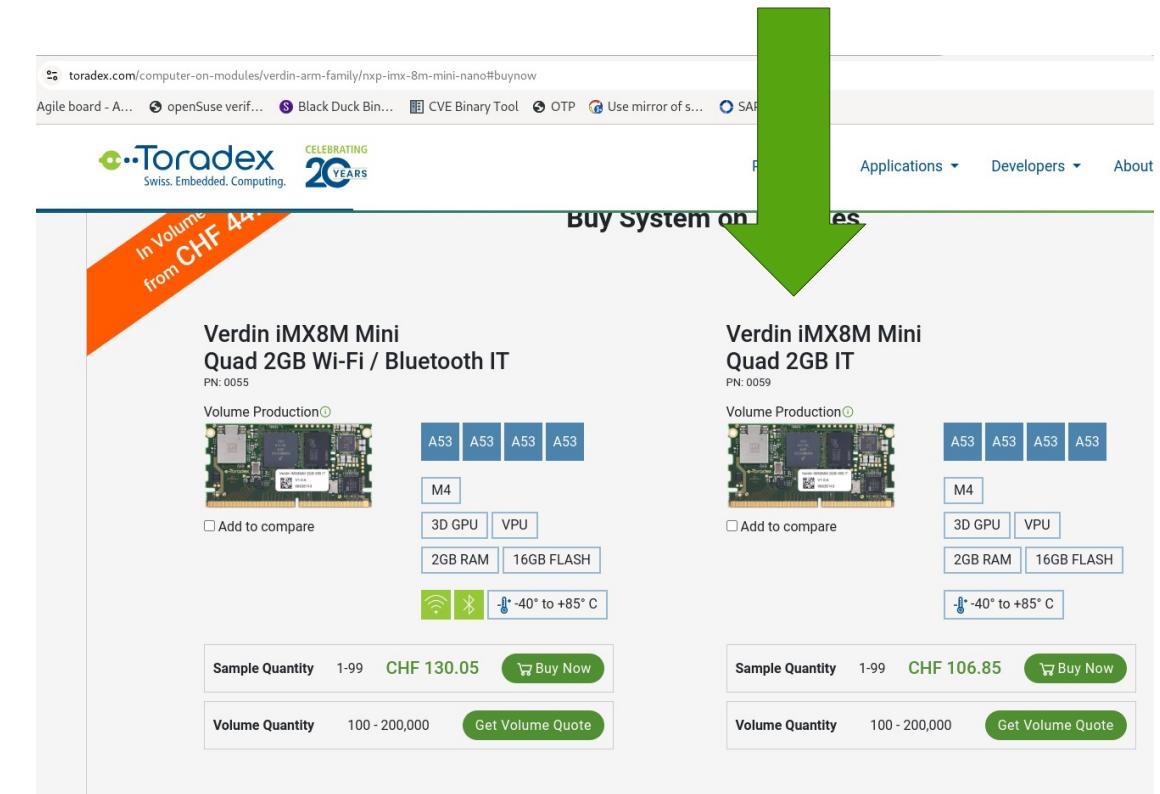
'./build.sh' clones all needed repos and builds the firmware

- Fedora Installer (dd'ed to an SDCard)

<https://dl.fedoraproject.org/pub/archive/fedora/linux/releases/38/Server/aarch64/iso/Fedora-Server-netinst-aarch64-38-1.6.iso>

- OS Image for the systemd-sysupgrade Demo

<https://github.com/mt-at-mt/os-image>



Thanks

- **All Systems Go!** For organizing this event and for the invitation
- **So many people, projects and communities made this a success**, especially:



[● ◀]
systemd



 debian



▪ METTLER TOLEDO

- For sponsoring my visit of this years All Systems Go! conference
- Burak Gerz for the systemd-sysupdate evaluation and testing
- Tobias Graemer for documentation and graphics

Current Open Positions

- **Embedded Software Engineer – Linux, Zürich / Switzerland**
Debian, systemd, u-boot, Kernel, PREEMPT_RT, ...
- **Senior DevSecOps Engineer, Karnataka, Tamil Nadu / India**
On prem k8s, Linux, Windows, maven, Gradle, MsBuild, NuGet

Join us now: <https://careers.mt.com/>

**One Team
Many Passions**

