YGREKY

CRA Compliance for Embedded Systems:

A Practical Look from the Yocto Project World

Marta Rybczynska

Who is Marta Rybczynska?

YGREKY

- PhD in Telecommunications
  - Network security/anonymity systems
- Open source/embedded developer/architect
  - 20+ years in open source
  - Contributions to the Linux kernel, various RTOSes
  - Co-maintainer of meta-security YP layer
  - Yocto Project security team and Open Embedded Technical Steering Committee member
    - But opinions here are my own
- Founder/CEO of Ygreky
  - Consulting (processes, architecture, audits)
  - Teaching (“Embedded Security”, webinars)
  - Current contribution to CRA-related standardisation
  - Conference keynotes

What is Yocto Project?

- A (Linux) distribution creation framework
  - Mostly used in embedded systems
  - Easy to extend and modify by hardware vendors
  - The “de-facto” standard for building custom Linux distributions
- Open source
  - A mix of licences, mostly MIT & GPL 2.0
  - A Linux Foundation project
- Usage
  - Automotive, set top boxes, robotics, routers, home appliances…
  - You (very likely) own devices with firmware built using the Yocto Project

Yocto Project and the CRA

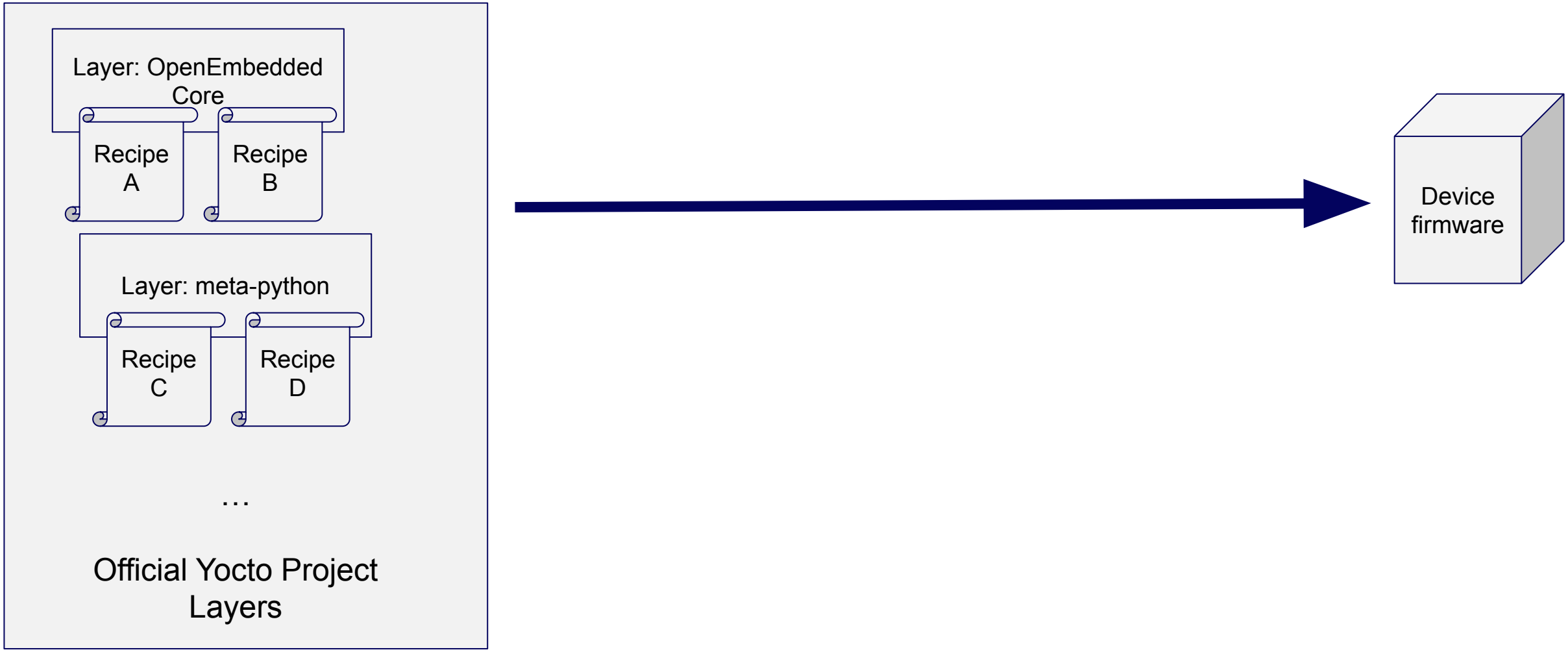
YGREKY

- Yocto Project is not a product with a manufacturer
  - Up to my best knowledge, a stewarded project
  - So, little obligations
- But users of the YP are (usually) manufacturers
  - Full CRA compliance required from them
  - Some products fall into important, or critical categories
  - In practice, YP core developers work on product code too

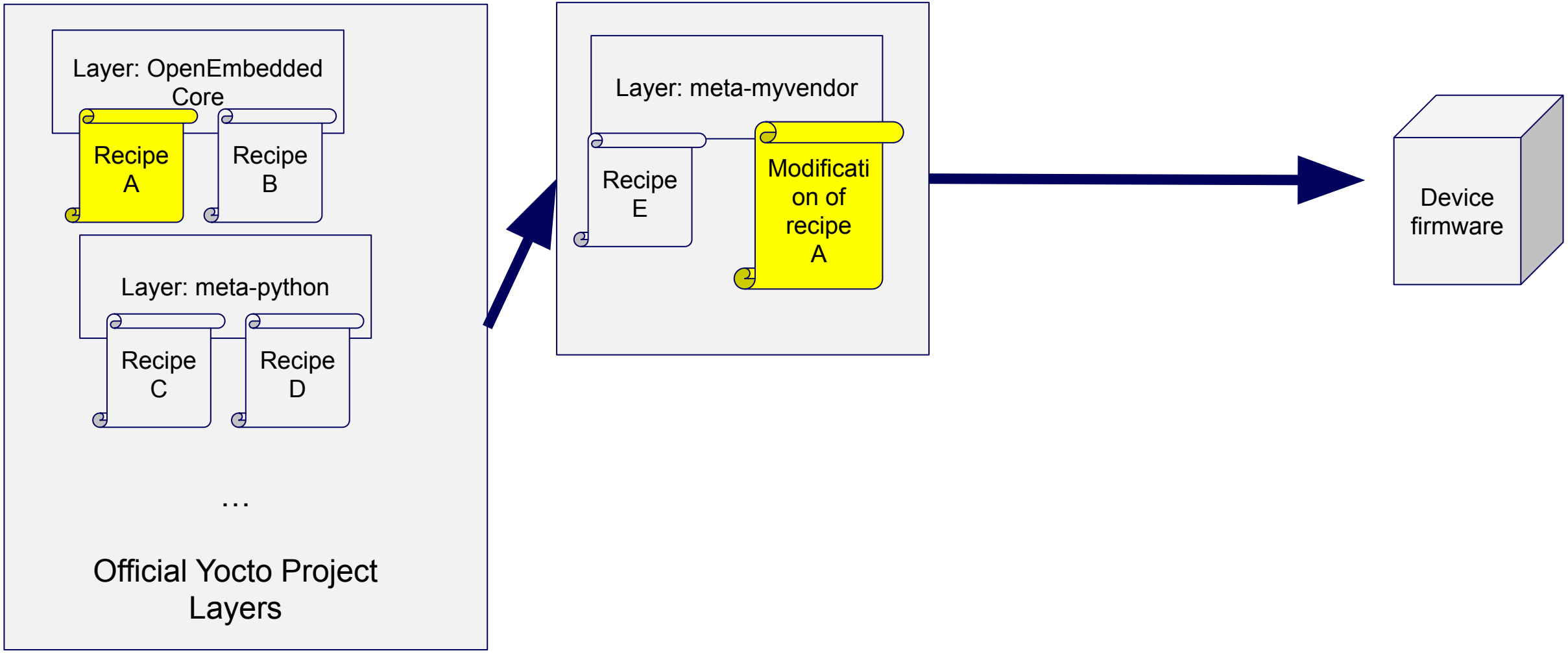
Recipes and layers

- A recipe
  - Instructions to build one software package
  - Points to source code
  - Runs the package's native build system
  - All technologies supported: from assembly and C, to Java, Go, Rust and Python
- A layer (meta-*)
  - A collection of recipes with a common purpose
  - Examples:
    - A layer to support specific hardware
    - A layer around a specific functionality, eg. meta-security

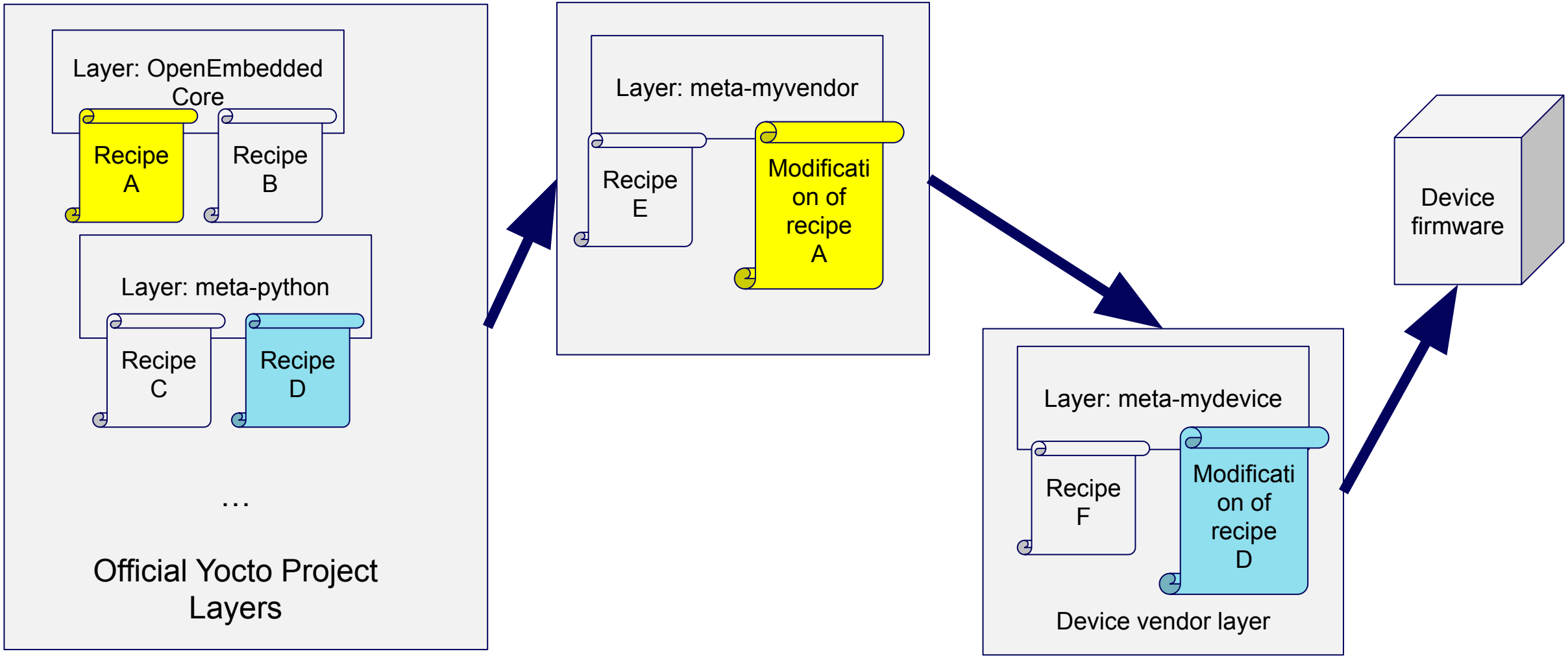
Yocto Project ecosystem example



Yocto Project ecosystem example



Yocto Project ecosystem example



A recipe with vulnerability fixes

```
SUMMARY = "GRUB2 is the next-generation GRand Unified Bootloader"

DESCRIPTION = "GRUB2 is the next generaion of a GPLed bootloader \
intended to unify bootloading across x86 operating systems. In \
addition to loading the Linux kernel, it implements the Multiboot \
standard, which allows for flexible loading of multiple boot images."

HOMEPAGE = "http://www.gnu.org/software/grub/"
SECTION = "bootloaders"

LICENSE = "GPL-3.0-only"
LIC_FILES_CHKSUM = "file://COPYING;md5=d32239bcb673463ab874e80d47fae504"

CVE_PRODUCT = "grub2"

SRC_URI = "${GNU_MIRROR}/grub/grub-${PV}.tar.gz \
           file://autogen.sh-exclude-pc.patch \
           file://grub-module-explicitly-keeps-symbole-.module_license.patch \
           file://0001-grub.d-10_linux.in-add-oe-s-kernel-name.patch \
           file://0001-RISC-V-Restore-the-typcast-to-long.patch \
           file://0001-misc-Implement-grub_strlcpy.patch \
           file://CVE-2024-45781.patch \
           file://CVE-2024-45782_CVE-2024-56737.patch \
           file://CVE-2024-45780.patch \
           file://CVE-2024-45783.patch \
           file://CVE-2025-0624.patch \
           file://CVE-2024-45774.patch \
           file://CVE-2024-45775.patch \
           file://CVE-2025-0622-01.patch \
           file://CVE-2025-0622-02.patch \
           file://CVE-2025-0622-03.patch \
           file://CVE-2024-45776.patch \
           file://CVE-2024-45777.patch \
           file://CVE-2025-0690.patch \
           file://CVE-2025-1118.patch \
           file://CVE-2024-45778_CVE-2024-45779.patch \
           file://CVE-2025-0677_CVE-2025-0684_CVE-2025-0685_CVE-2025-0686_CVE-2025-0689.patch \
           file://CVE-2025-0678_CVE-2025-1125.patch \
"
```

YGREKY

The official YP grub2 recipe https://git.openembedded.org/openembedded-core/tree/meta/recipes-bsp/grub/grub2.inc

hash 1fe39a59d2d9dc6909ba88bfceaf6fd4222c13d2

For grub2 version 2.12

What does Yocto Project offer for CRA compliance? (1)

Directly visible from the sources

YGREKY

- SBOM generation - in any layer setup
  - Build-time SBOM, detailed
  - SPDX3, possible SPDX2.2 (default in the current LTS)
- Provenance information
  - All download locations and patches are traced
- Known vulnerability checking
  - Cve-check class, working with any layer setup
  - With annotations in recipes (status, package name fixes)
- Security tools packaged
  - Layers and recipes exist for secure boot, security tools, even intrusion detection systems

What does Yocto Project offer for CRA compliance? (2)

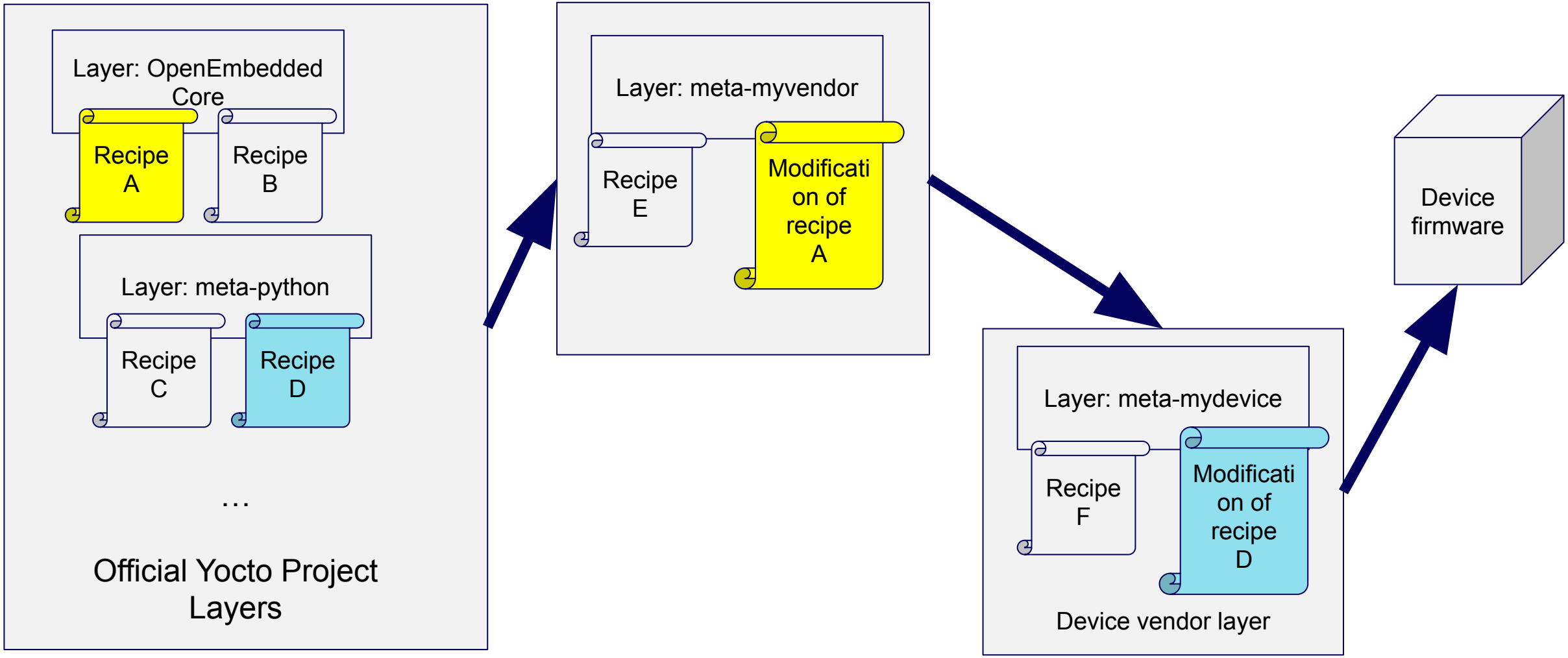
Process and organization

- Long Term Support version and process
  - 4 years support for security fixes
  - Process of accepting fixes/updates into the LTS
- Testing
  - Continuous and release testing, on multiple architectures
  - Frequently showing upstream bugs
  - Regular communication with upstream projects
- Vulnerability reporting and management
  - Security team at the Yocto Project
  - Management of security fixes (tagged patches)

Challenges

Yocto Project ecosystem example

YGREKY



Challenges (1)
Software names and versions

- What is the product name and version of the package from recipe A? And with the change?
  - The current answer: “it is both A”, same version
  - If patches remove security vulnerabilities, annotations by hand
  - What if there are hundreds of vendor patches?
    - Can be even more complex, include patches from yet another vendor
    - Difficult match with the way vulnerability databases are done

Challenges (2)
Configuration and processes

- Secure by default configuration
  - Current policy: minimize the changes from upstream
    - Issue: what if upstream has unsecure configuration by default?
    - In fact, quite common
  - Multiple configurations possible
    - YP is a tool to build distributions, not a distributions
    - Use cases from development board to industrial -> different needs
    - Complicated to come with a configuration people agree on
  - Vendor changes
    - Ecosystem vendors are shipping hardened configurations -> with their own fixes

Challenges (3)
Choice of solutions

YGREKY

- Example: There is no update system included by default
  - Instead: multiple integrations in meta layers
    - Result: work on the manufacturer’s side
    - But: can adapt to update by USB or over the air
- Consequence: hard to write a common risk analysis

Challenges (4)
Vendors not up to date

- Vendors are years (literally) behind upstream
  - Device vendors depend on BSP vendors
  - Consequence: no security updates
  - Consequence: important changes deployed in production years after
- It is slowly changing with the CRA
  - December 2027 is “tomorrow”

Wrapping-up

CRA and embedded
Uneven results

- What works well
  - SBOM
  - Metadata (locations, patches)
  - Even better is the vendor is close to upstream
- What requires work (2027 and beyond)
  - Years of past fragmentation in security solutions
  - Secure by default and choice of software packages
    - Outdated packages, or unsecure by default
    - Upstream problem, but visible at the integration
  - Hardware diversity
    - Example: Until recently, secure boot was a per-chip solution

Questions?