
Beyond the Release
Managing Long-Term Risk and Compliance in Embedded Linux with Yocto
INNOVATE. INTEGRATE. EXCEED.
inovex

inovex

anna-lena.marx@inovex.de

+49 1523 / 31 81 26 0

@anna-lena-marx-embedded

@Allegra42

Hello,

I’m Anna-Lena Marx

Senior Embedded Dev / Tech Lead Embedded Linux, Karlsruhe (Germany)

- with inovex since 2015
- has a Master’s degree in Embedded Systems
- ~~studies~~ studied Electrical Engineering as a hobby
- Yocto user since ~2017? Pre-LTS times!

#embeddedsystems #yocto #linux #kernel #zephyr #aosp

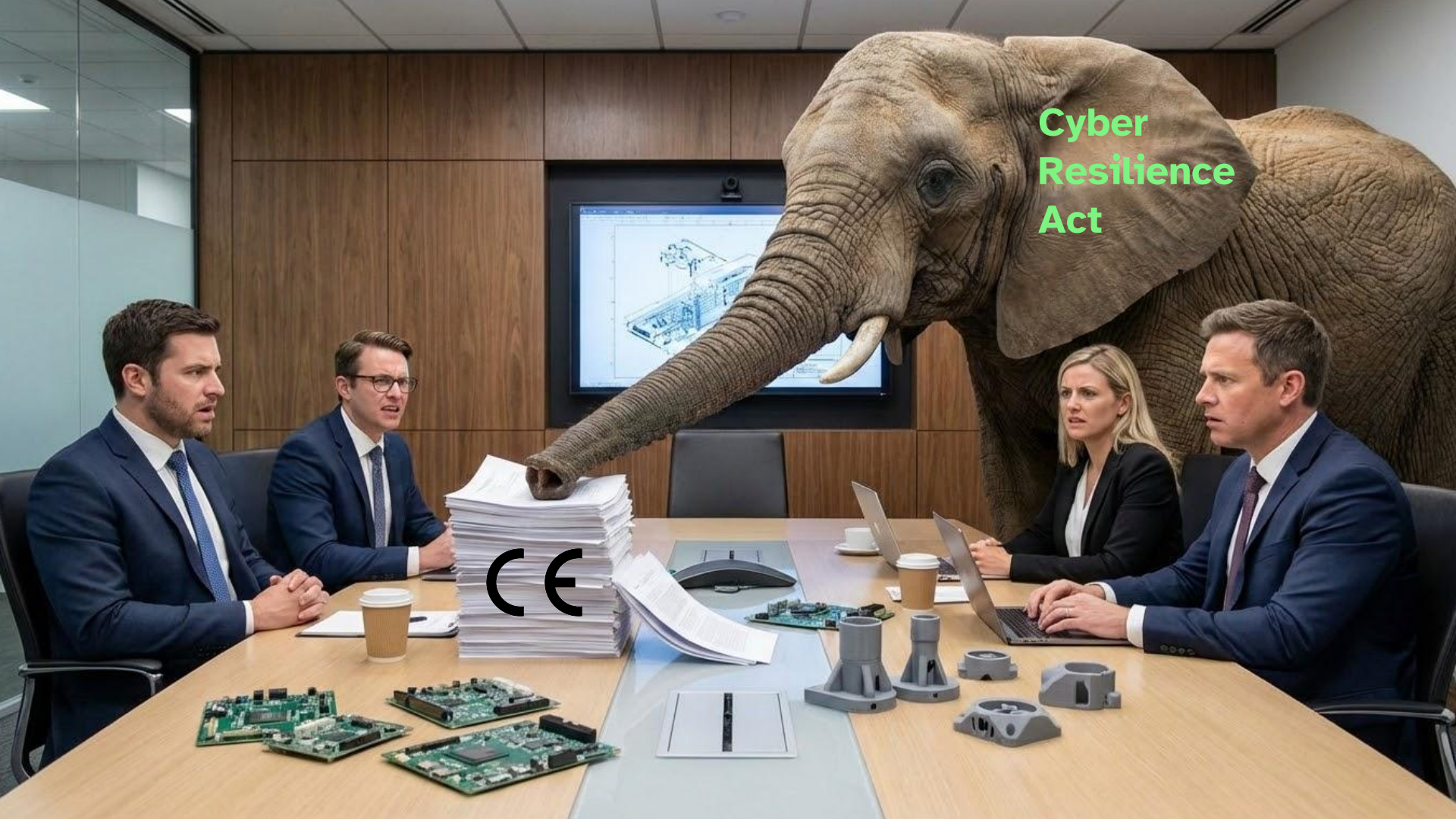
inovex

Yocto has always been built to support products in the long run ...

inovex

... but let me introduce

the new elephant in the room


Cyber
Resilience
Act
CE

Software can no longer be treated as a necessary ~~evil~~ part of the product

- **Security as a Core System Architecture Guideline**
  instead of just don’t populating the debug UART
- **Continuous Vulnerability Management**
  instead of maybe fix critical bugs
- **Continuous Liability** (minimum 5 years, rather longer)
  instead of fire and forget

The importance of Software (Firmware) increases from a regulatory perspective! And it should do the same within product development.

CRA

ANNEX I

Part I

Essential Cybersecurity Requirements

- secure standard configuration
- no known exploitable vulnerabilities
- security updates
- access control
- minimizing the attack surface
- ...

Part II

Vulnerability Handling Requirements

- SBoM
- security tests, automated scans, pentests
- continuous monitoring (e.g. CVE databases)
- security updates ➔ secure distribution
- ...

CRA

ANNEX I

Part I

Essential Cybersecurity Requirements

- secure standard configuration
- no known exploitable vulnerabilities
- security updates
- access control
- minimizing the attack surface
- ...

yocto PROJECT

Part II

Vulnerability Handling Requirements

- SBoM
- security tests, automated scans, pentests
- continuous monitoring (e.g. CVE databases)
- security updates ➔ secure distribution
- ...

CRA

ANNEX I

Part I

Essential Cybersecurity Requirements

- secure standard configuration
- no known exploitable vulnerabilities
- security updates
- access control
- minimizing the attack surface
- ...

Part II

Vulnerability Handling Requirements

- SBoM
- security tests, automated scans, pentests
- continuous monitoring (e.g. CVE databases)
- security updates ➔ secure distribution
- ...

Board Support Package

Hardware

inovex

The **Layer Bloat** Pitfall

Vendor **BSPs Layers** often include

- demo applications
- (proprietary) additional drivers or binary blobs
- predefined distros and images
- and even **more**!

The Risk:

➔ applications installed from “nowhere” → maybe hard to control

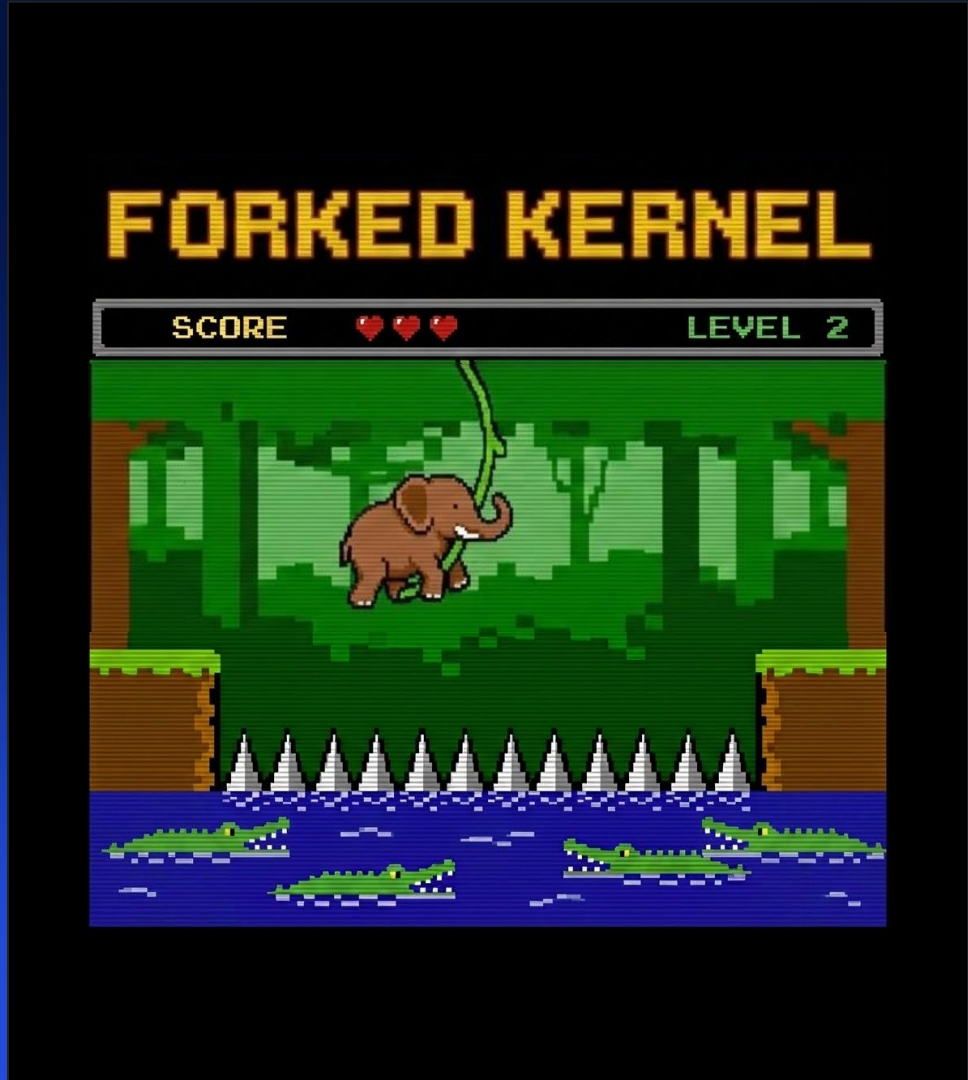
➔ bloated system

➔ your system becomes a black box

Understanding your system and what’s included for what reason in depth is essential for understanding and limiting risks.

inovex

The **Forked Kernel** Pitfall



Vendor BSPs often use **forked kernels**

The Reason:

- to showcase unique components and features
  - not yet upstream
  - containing proprietary binary blobs
- to ease their own development and maintenance workflows

The Risk:

➔ upstream security patches are not pulled in time

➔ diverged codebases → manually applying patches fails

➔ additional effort for backporting critical fixes

Evaluating a BSP's kernel maintenance strategy in the past gives you a crucial insight into your future product maintenance strategy.

inovex

Yocto as **the Compliance Engine**

CRA

ANNEX I

Part I

Essential Cybersecurity Requirements

- secure standard configuration
- no known exploitable vulnerabilities
- security updates
- access control
- minimizing the attack surface
- ...

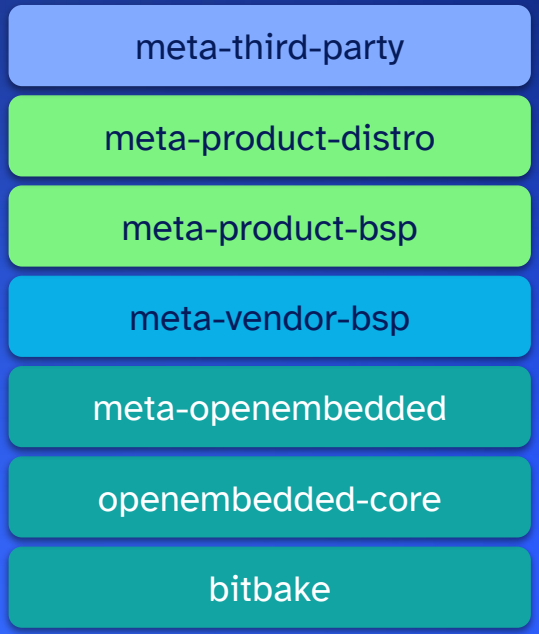
yocto PROJECT

Part II

Vulnerability Handling Requirements

- SBoM
- security tests, automated scans, pentests
- continuous monitoring (e.g. CVE databases)
- security updates ➔ secure distribution
- ...

Owning the System



- The system is always **built from source**
  ➔ full-stack patch-ability
- **archiver.bbclass** can create source code archives for each released version
  ➔ fix issues, independent from upstream code archives
- Yocto as a system concept and Bitbake as a build system ensure **reproducible builds**
  ➔ gain independence from single persons and golden images

Taking ownership for your specific system and ensuring source code access, full-stack patchability and reproducibility is a cornerstone of a compliant, maintainable system.

Security Built-In

- Remove unneeded **IMAGE_FEATURES** and **DISTRO_FEATURES**
- Enable **security flags**
- Minimizing the system by **explicitly installing packages**
- **Limit** service and user **rights**
- Remove unnecessary or potentially dangerous kernel configurations
  →see
  https://github.com/a13xp0p0v/kernel-hardening-checker

See `meta-security` for more advanced mechanisms!

→
```
IMAGE_FEATURES:remove = “”
DISTRO_FEATURES:remove = “”
```

→
```
require security_flags.inc
```

→
```
IMAGE_INSTALL = “openssh myapp”
```

→
```
<recipename>.bbappend
```

→
```
<kernelrecipename>.bbappend
```

```
FILESEXTRAPATHS:prepend := "${THISDIR}/files:"
SRC_URI += "\
      file://hardening.cfg \
 "
```

Compliance and Software Traceability by Design

- Yocto already offers deeply **integrated license management**
  ➔ license.bbclass
  ➔ *Software built in languages like Go and Rust with integrated package management needs additional attention for proper license management in Yocto!*

➔ `INCOMPATIBLE_LICENSE = “GPL-3.0* LGPL-3.0*”`

- SBoM generation is built-in and enabled
  ➔ create-spdx.bbclass

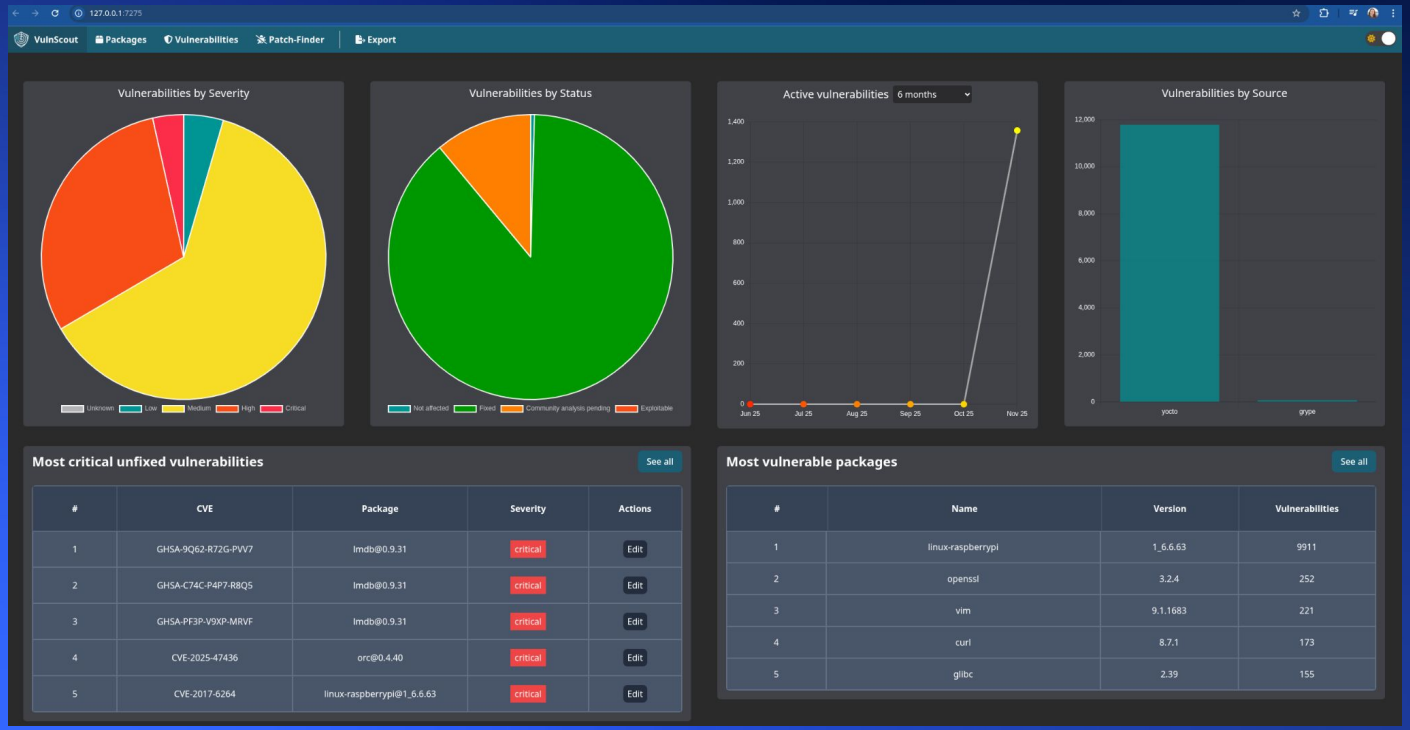
➔ ~~`INHERIT += "create-spdx"`~~ → now enabled by default

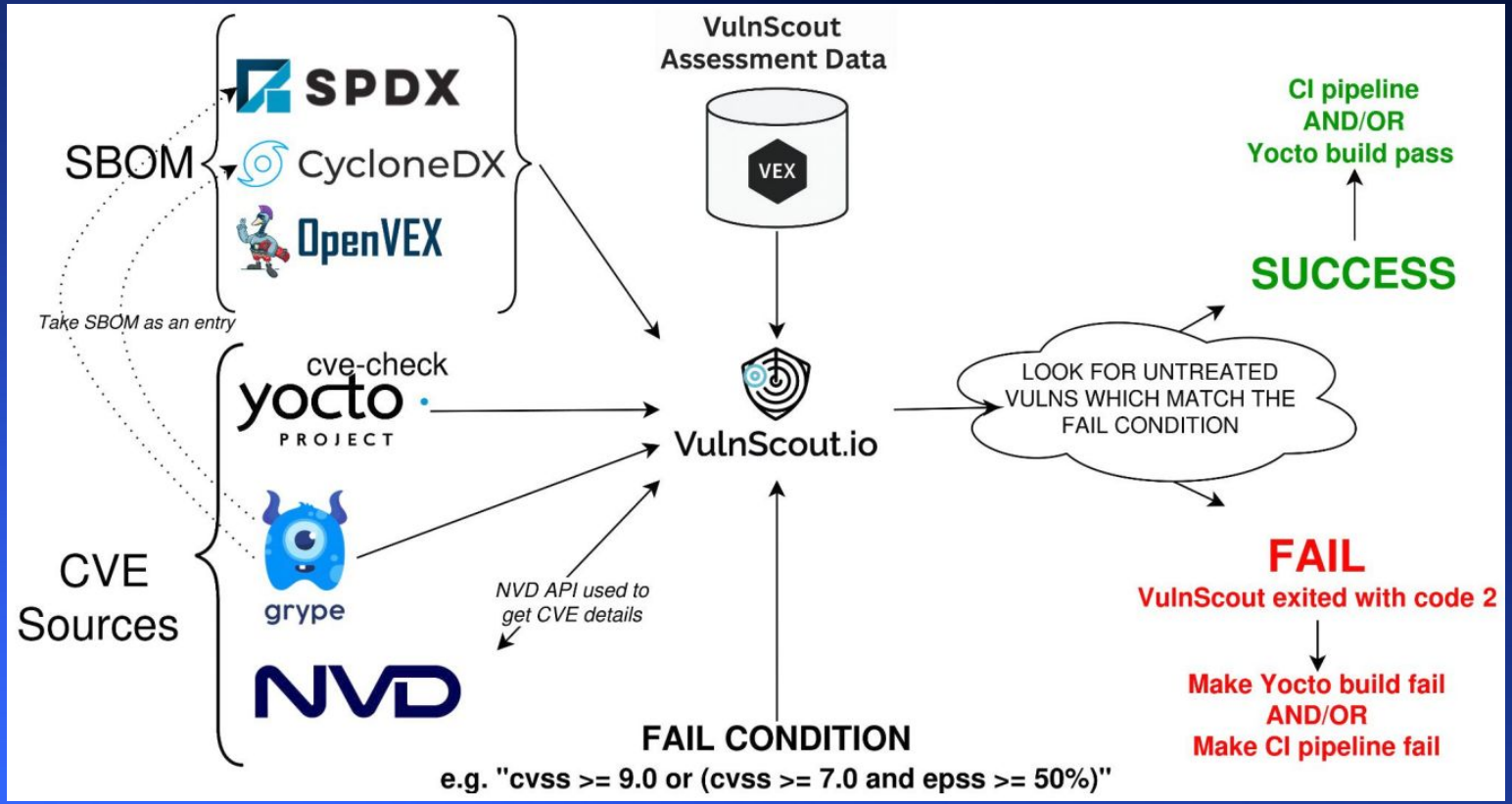
- Basic CVE checking is already included
  ➔ sbom-cve-check.bbclass

➔ `OE_FRAGMENTS += "core/yocto/sbom-cve-check"`

Compliance and Software Traceability by Design

CVE Monitoring: VulnScout





https://pretalx.com/media/yocto-project-summit-2025-12/submissions/9JWKV7/resources/_uTXsryR.pdf

Updates

MENDER

RAUC

OSTree / meta-updater / aktualizr

systemd-sysupdate (future)

- **Image-based Updates** over Package-based ones
  → ensure clear and reproducible image states
  → reduce the combinatorial exploitation of system states in the field
- **Separate Functional and Security Updates**
  → CRA recommended
- **Sign the Update Artifacts**
  → only allow trusted updates

Test the process!

- → power disconnects
- → roll-back behaviour
- → offline devices
- → expired signing certificates
- → fall-back mechanism, e.g. USB updates
- → **get creative!**

https://wiki.yoctoproject.org/wiki/System_Update

The Path Forward

- Fearless long-term maintenance starts already by selecting a hardware platform
- Plan for your internal and regulatory requirements before writing the first line of code
- Integrate an update client, test it well!
- **Take ownership for your system**
  Well developed software is a chance for your product, not a nagging problem!

Starting with Yocto is complex, but it pays off in the long run!

Thank you!

Anna-Lena Marx

Tech Lead Embedded Linux Systems

+49 1523 / 3181260

anna-lena.marx@inovex.de

inovex