

● PRESENTED AT EMBEDDED WORLD NORTH AMERICA 2026

CRA: Crying Regarding Audits

Cyber Resilience Act guidance for embedded products,
firmware teams and Zephyr RTOS

Embedded Intelligence

Arkadiusz Grzelka · arivEmb · arivemb.com

arivEmb

Presented at EWNA 2026

BSP, firmware and platform enablement from first boot to production.

arivemb.com

Embedded engineering

We develop software for embedded systems - and build the tools we develop it with.

UPSTREAM

Zephyr, upstream

We contribute to mainline Zephyr RTOS.

WORKED EXAMPLE

zdx

Worked example: a controlled Zephyr engineering workflow connecting source, build evidence, review and hardware validation.

BSP / PLATFORM ENABLEMENT · ZEPHYR / FREERTOS · EMBEDDED LINUX / YOCTO · DEVICE SECURITY

Cyber Resilience Act

Regulation (EU) 2024/2847. A product law, like CE marking: cybersecurity becomes a condition for selling in the EU.

THE CRA IN ONE SENTENCE

If you sell a device with software in the EU, you must:

- 01 ship it with no known exploitable vulnerabilities,
- 02 fix vulnerabilities through a generally 5+ year support period,
- 03 report actively exploited vulnerabilities within 24 hours,
- 04 and be able to prove all of the above.

10 Dec 2024

Regulation in force,
nothing owed yet

11 Sep 2026

Report exploited vulnerabilities and incidents -
also for devices already sold

11 Dec 2027

New products: secure by design,
SBOM, security updates, CE mark

● WHY A US COMPANY CARES

The CRA follows the product, not the company

- 01** **Selling in the EU** triggers the obligations - a non-EU manufacturer carries them directly.
- 02** **Art. 18 allows an authorised representative; core manufacturer obligations remain with you.**
- 03** **Your EU importers and distributors** can't sell a non-compliant device from Dec 2027 (Arts 19-20).

**The pressure does not arrive as an EU inspector.
It arrives as your distributor's compliance questionnaire.**

SAME BUILDING BLOCKS IN THE US



FDA §524B

Qualifying cyber devices: premarket submissions include plans, processes, patches and SBOM information.



FCC Cyber Trust Mark

Voluntary label for consumer wireless IoT, based on NIST IR 8425 - incl. secure updates.



Federal procurement

NIST minimums for agency-owned IoT; SBOMs if the agency's contract asks.



California SB-327

Reasonable security; unique or user-set credentials where authentication occurs outside a LAN.

SBOM · vulnerability handling · secure updates · no default credentials

Your product: **a BLE IoT device running Zephyr RTOS, sold in the EU.**

EXAMPLE SCENARIO



From: security researcher Received: Fri 23:00

Hi, I can take over your device over BLE without pairing.
PoC attached. It's being used in the wild.

The clock has started: early warning due in 24 hours.

WHAT YOU NEED TO ANSWER - FAST

01 Is it real, and is it being exploited?

02 Which products and firmware versions are affected?

03 Is the bug in our code, Zephyr, or a vendor SDK?

04 How fast can a signed fix reach the field?

Same habits: fewer incidents, faster answers

BEFORE AN INCIDENT

Make incidents rare - next 4 slides

- Shrink the attack surface
- Attack it yourself first
- Ship it: SBOM, signing, updates
- Watch: scan what you ship



same
tools

WHEN ONE HAPPENS ANYWAY

Answer fast - the 4 slides after

- The first hours: do and report
- Reproduce, fix, ship
- Key or password leaked? Rotate it - fast
- Close the loop

THREE RULES THAT COME BACK LATER

RULE 1 - EVIDENCE IS A BUILD ARTIFACT

Let CI produce the proof

SBOMs, test and scan results come out of every build - not rebuilt by hand on Friday night.

RULE 2 - NOTHING FLOATS

Pin every version

Every dependency, container and tool at a fixed version, so any release can be rebuilt years later.

RULE 3 - UNKNOWN IS NOT GREEN

Can't check = failed

An empty scan report only counts if the scanner could actually see your code.

Shrink the attack surface, pin the supply chain

ENTRY POINTS - WHERE ATTACKER INPUT ARRIVES

BLE

DFU / SMP

Shell / UART

USB

Debug port



WHAT THEY REACH

Parsers

Settings

Crypto

DEVELOPER HABIT

- List every interface an attacker can reach before coding it
- Keep each interface in its own module, so it can be fully tested
- Keep test code isolated; prioritise boundaries, negative cases and attacker-controlled paths
- Pin every module to a fixed revision
- Never commit a secret

TOOLS

Kconfig

Zephyr snippets

Twister

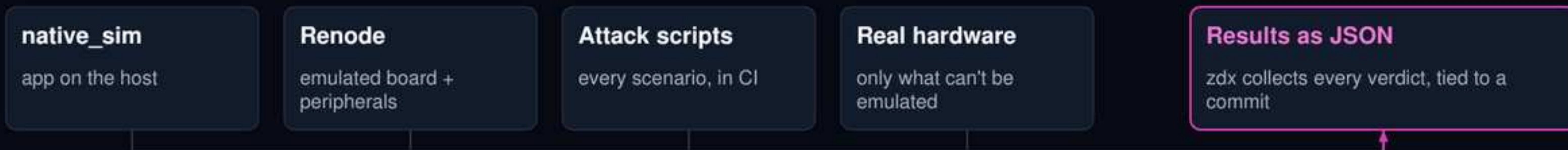
pre-commit hooks

ZDX

- A catalogue of entry points and what they reach, for every review
- A CI check fails if any dependency tracks a branch instead of a fixed revision
- An AI agent reviews vendor SDK security advisories - a review older than 90 days is flagged as stale

Attack it yourself first

Hardware - and access to it - is the bottleneck. Emulate it, and run every attack scenario in CI.



DEVELOPER HABIT

- Emulate as much as you can - hardware and access to it are the bottleneck
- Run every attack scenario as part of the CI release
- A static analyser on every PR
- Attack anything on the attack surface - and never let its author sign it off

CodeChecker - open-source static analysis: finds bugs without running the code. Built into Zephyr.

TOOLS

Twister

native_sim

Renode

CodeChecker

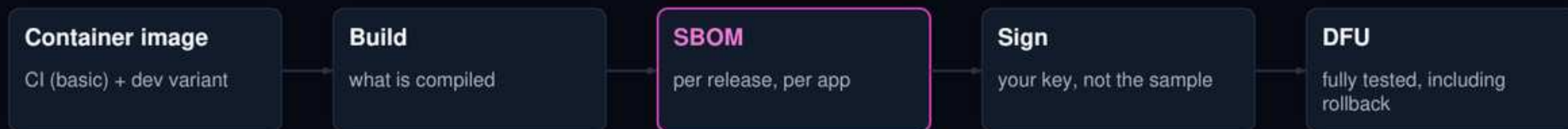
Custom attack scripts

ZDX

- One off-hardware test ladder, the same locally and in CI
- AI-assisted review that traces attacker input to dangerous code
- Independent red team: AI agents + hardcoded attack scenarios over DFU, shell, BLE

Ship it: know what's inside, sign it, keep it updatable

Fully automated release: CI builds every artifact. People make mistakes - a pipeline repeats exactly.



DEVELOPER HABIT

- Containerize the build: a basic CI variant and a dev variant with extra tools
- No manual release steps: CI generates signed images, SBOM, test reports
- SBOM per release and per app - generated by CI, stored with the other release artifacts
- Sign with your own key
- Test the full DFU process end to end - including rollback

TOOLS

Docker

GitHub / GitLab CI

west spdx

MCUboot + SMP

SPDX - open standard SBOM format (ISO/IEC 5962).

ZDX

- Predefined Docker images: CI and develop variants
- SBOM per app: what was compiled, not what was checked out
- Attached to every shipped version by CI
- Fails if firmware is signed with an SDK sample key
- The full DFU process tested like any other code: upload, test, confirm, revert

Scan what you ship - does it find the real vulnerabilities?

CVE

Public ID of one known vulnerability.

CVE-2025-10456 (Zephyr BLE issue; fixed for 4.2.0 and the maintained 3.7 branch)

CPE

Standard product + version name.

cpe:2.3:o:zephyrproject:zephyr:4.1.0

EUVD · NVD

Databases linking CVEs to products.

EUVD: EU, run by ENISA · NVD: US, run by NIST

HOW THE SCANNER MATCHES CVEs

The databases link each CVE to CPEs and version ranges. The scanner checks whether the CPE in your SBOM falls in range.

WHY IT CAN MISS - AND STILL SAY "CLEAN"

- Your SBOM has no CPE - e.g. Zephyr built from a commit, branch or vendor fork
- The CVE has no CPE yet - the databases are behind on adding them
- The names don't match - "zephyrproject-rtos" vs "zephyrproject"

DEVELOPER HABIT

- Scan on every build and on a schedule
- First check what the scan could see

TOOLS

west spdx (SBOM)

Grype (scanner)

Offline vulnerability DB

An empty report proves nothing.

ZDX

- Refuses a CVE DB older than 7 days
- Scans only the compiled image
- "Could not check" is never "clean"
- Lets you add Zephyr's CPE

The first hours: what to do, and what to report

Is it real, what is affected, and who do we have to tell?

DURING THE INCIDENT

- 01 Is it real - and actively exploited?
- 02 Acknowledge the reporter
- 03 Open a private tracking issue
- 04 List affected versions from the SBOMs
- 05 If exploited: early warning within 24 h

BEFORE ANY INCIDENT

Decide who reports, who talks to users and who fixes - and write it down in a runbook.

Actively exploited vulnerability

Someone is using a bug in your product

24 h

early warning

72 h

details + mitigations

14 days

final report, after the fix

Severe incident

e.g. your build or signing system was breached

24 h

early warning

72 h

details + mitigations

1 month

final report

Report via ENISA's single reporting platform (web portal). Tell affected users without undue delay.

Reproduce, fix, ship

Can we reproduce it, does the fix break anything, how fast does it reach devices?



**A rushed fix that bricks devices
is incident number two.**

Key or password leaked? Rotate it - fast

CASE 1

Credential in the repo

- Revoke and replace the secret - deleting it from the repo doesn't un-leak it
- Per-device secret: one leak = one device
- Shared secret: one leak = the whole fleet, fixable only by updating every unit

CASE 2 · ALMOST HOPELESS

Signing key leaked

- An attacker can sign firmware with the leaked key - and devices may accept it as yours
- Upstream MCUboot has no portable, universal field-revocation mechanism for a compromised root signing key
- Recovery depends on the product architecture and hardware - design rotation and recovery before release

**Every other incident is fixed by shipping an update.
This one breaks the mechanism you would ship it with.**

Close the loop - so it never happens again

AFTER THE FIX

- 01 Keep the exploit tests in every CI release**
forever - the same bug can never come back unnoticed
- 02 Third-party bug? Notify the component maintainer**
Share the issue, fix and supporting information where applicable - CRA Art. 13(6)
- 03 Publish an advisory for your users**
what is affected, and what to update to
- 04 Send the CRA final report**
within 14 days of the fix being available
- 05 Record which products are not affected (VEX)**
so the next scan does not raise it again

TOOLS

[Twister](#)[OpenVEX](#)

WHAT NO TOOL DOES FOR YOU

- Decide that the product conforms (CE)
- File the CRA reports
- Run the security inbox
- Talk to affected users

**Tools produce the evidence.
People make the decisions.**

● STOP CRYING

Five things to do on Monday

- 1 Pin every dependency to a fixed revision.
- 2 Generate the SBOM from the build; keep it for every release.
- 3 Scan it in CI - and make sure the scanner can see every component.
- 4 Replace the sample signing key and plan how you would rotate it.
- 5 Write the incident runbook: who reports within 24 h, who fixes, who tells users.

NO CI · AN AFTERNOON

NO CI · AN AFTERNOON

**You cannot prevent every incident.
You can prevent the crying.**

arivemb.com
Questions?

The End

Sources · revised 4 Oct 2026 · Technical guidance, not legal advice.

CRA

[Regulation \(EU\) 2024/2847](#)
[EC: CRA summary](#)
[EC: CRA reporting obligations](#)
[ENISA single reporting platform](#)
[EU Vulnerability Database \(EUVD\)](#)
[Zephyr Vulnerability Alert Registry](#)
[Zephyr Project: CRA guidance](#)

US

[FDA §524B guidance](#)
[U.S. Cyber Trust Mark \(FCC\)](#)
[NIST consumer IoT \(IR 8425\)](#)
[California SB-327](#)

TOOLS

[Twister](#)
[Zephyr snippets](#)
[Renode](#)
[Zephyr SCA \(CodeChecker\)](#)
[west spdx](#)
[MCUboot](#)
[Grype](#)
[OpenVEX](#)