LINUTRONIX

2026-07-07

CRA-Anforderungen mit ETSI-Standards für Embedded Linux implementieren

Tobias Deiminger, Simone Weiß

Hinweis

Dieser Vortrag soll helfen…

- Den Charakter von IEC 62443, ETSI 304–x und dem CRA zu verstehen
- Das Potenzial von Open-Source-Software für Zertifizierung zu erkunden
- Unsicherheit in handhabbare erste Schritte umzuwandeln

Dieser Vortrag wird nicht…

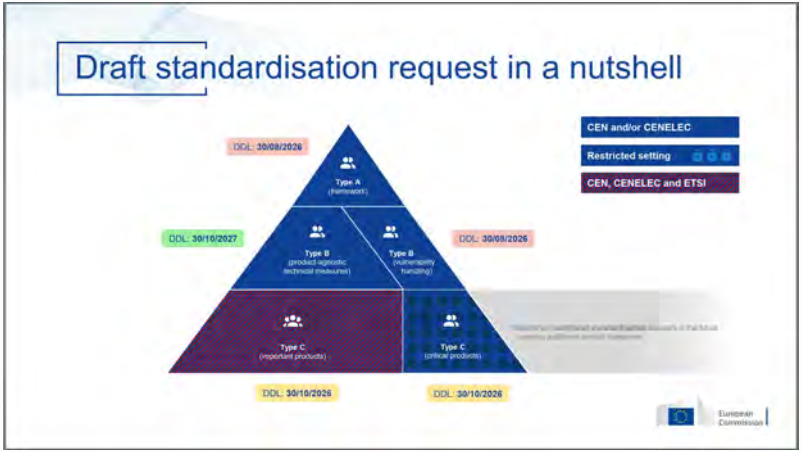
- Rechtsberatung bieten
- Technische Details oder formale Ausnahmen aufschlüsseln
- Eine vollständige Anleitung zur Zertifizierung bieten

LINUTRONIX

Agenda

1. **CRA – Standardisierung und Produktklassen** — M/606, harmonisierte Normen, Produktklassen
2. **Open Source unter dem CRA** — Linux Kernel, Produktbewertung, Konformitätsbewertungsverfahren
3. **Praxisbeispiel: Linux-BSP zertifizieren** — Von IEC 62443 zu EN 304 623/626
4. **Die 6 Schritte + Restrisiko: EN 304 623/626 anwenden** — Scope, Use Cases, Requirements, Traceability

CRA – Standardisierungsauftrag M/606



Quelle: European Commission – Draft standardisation request in a nutshell

CRA – Harmonisierte Normen

M/606: 41 harmonisierte Normen – 15 horizontal + 26 vertikal

Typ A – Framework — *CEN/CENELEC* – **08/2026**
EN 40000-1-1 (Vocabulary), EN 40000-1-2 (Cyber Resilience Principles)

Typ B – Vulnerability Handling — *CEN, CENELEC, ETSI* – **08/2026**
EN 40000-1-3

Typ B – Produktübergr. techn. Maßnahmen — *CEN/CENELEC* – **10/2027**
EN 40000-1-4 (Generic Security Req.), TR 40000-1-5 (Threats & Security Obj.)

Typ C – important & critical Produkte — *ETSI* – **10/2026**
EN 304 617-642: 18 vertikale Standards (u. a. **623 Boot Manager**, **626 OS**, 617 Browser, 620 VPN, 627 Router, 636 Firewalls)

Weitere: EN IEC 62443-4-1/-4-2 (Industrie), EN 50764-50766 (Smart Cards, μC)

Quelle: OpenSSF CRA Standards Map

LINUTRONIX

CRA – Produktklassen & Zeitplan

Default
Self-Assessment
z. B. Sensoren, Festplatten, Drucker, SPS

→

Important
Standards oder Dritte
z. B. VPNs, Firewalls, Router, OS, Bootloader

→

Critical
Pflichtzertifizierung
z. B. Smart Cards, Secure Elements, Smart Meter

Sep 2026: Meldepflicht Schwachstellen → **Dez 2027:** Volle Anwendung

ETSI EN 304-x: Harmonisierte vertikale Standards für den CRA

Was sind sie?

- Entwürfe harmonisierter Standards zur Ergänzung des CRA
- Übersetzen CRA-Anforderungen in produktspezifische Sicherheitsspezifikationen

Aktueller Status

- Noch in der Entwurfs- und Konsultationsphase
- Praktische Erfahrung noch gering

EN 304 623 – Boot Manager & **EN 304 626** – Betriebssysteme

Vergleich: ETSI EN 304 623/626 vs. IEC 62443-4-x

ETSI EN 304 623/626

- Produktspezifisch
- Boot Manager & Betriebssysteme
- Abgeleitet aus CRA-Anforderungen
- Noch im Entwurf

IEC 62443-4-x

- Domänenspezifisch
- Industrielle Automatisierung
- Etablierter Industriestandard
- Ausgereift & aktiv zertifiziert

Trotz unterschiedlichem Scope: erhebliche praktische Überschneidungen!

LINUTRONIX

Harmonisierte Normen: IT- und OT-Varianten

ETSI IT Standards	CENELEC OT Standards based on IEC 62443
EN 304 623 – Boot Manager	**EN 50770-2** – Netzwerkmanagement
EN 304 626 – Betriebssysteme	**EN 50770-3** – Netzwerkschnittstellen
EN 304 627 – Router/Switches	**EN 50770-5** – Router, Modems, Switches
EN 304 620 – VPN	**EN 50770-4** – VPN
EN 304 636 – Firewalls	**EN 50770-1** – Firewalls, IDS/IPS
EN 304 617 – Browser	**EN 50770-6** – SIEM
…	

OT-Varianten basieren explizit auf IEC 62443 – bestehende 62443-Arbeit zahlt sich aus!

LINUTRONIX

Wie gut ist eigentlich Open Source? – Beispiel Linux Kernel

Weltraum
Mars-Helikopter Ingenuity,

SpaceX, Starlink

Automotive
Automotive Grade Linux:

Tesla, VW, BMW, Toyota

Infrastruktur
96% der Top-1M-Server,

100% der Top-500-Supercomputer

Medizintechnik
Bildgebung, Monitoring,

Infusionssysteme

Industrie
SPS, Robotik, SCADA,

Embedded Controller

Maritim & Verteidigung
U-Boot-Sonar (US Navy),

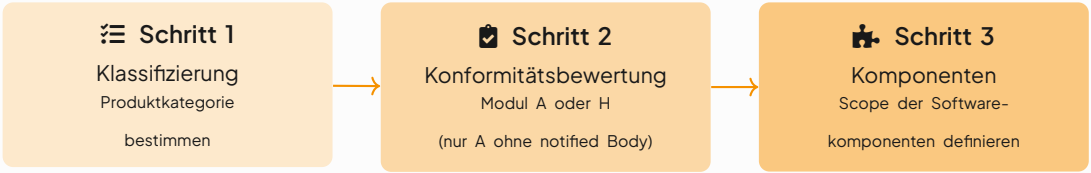
Drohnen, Radar

Open Source ist die Basis kritischer Systeme weltweit

LINUTRONIX

Open Source ist gut – bauen wir ein Produkt!

Was müssen wir tun, um ein CRA-konformes Embedded-Linux auf den Markt zu bringen?



CRA-Klassifizierung nach 2025/2392 Anhang I

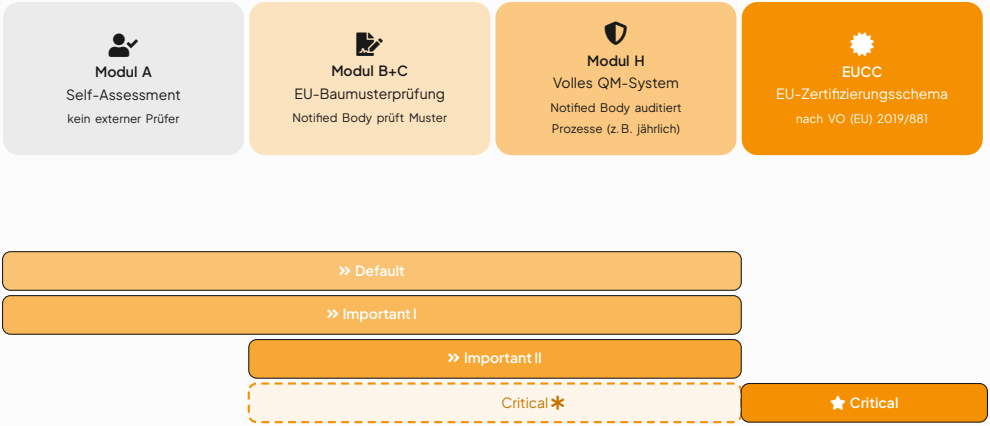
Produktsicht
Aus Nutzersicht ist das Produkt ein **Betriebssystem** (Operating System)

Bootloader-Merkmale
Immutable System als Hauptfeature – diese Fähigkeit wird zu wesentlichen Teilen vom Bootloader bereitgestellt → indirekt auch Merkmale eines **Boot Managers**

Einordnung
IMPORTANT PRODUCT Class I – Kategorie *Operating System* und Kategorie *Boot Managers*

Ref: Hauptverordnung 2024/2847 Anhang III, Class I Nr. 8 (Boot Managers) & Nr. 11 (Operating systems); Durchführungsverordnung 2025/2392 Anhang I

CRA-Konformitätsbewertungsverfahren im Überblick



ℹ Modul A: bei *Default* auch ohne harmonisierten Standard (“Freestyle”), bei *Important I* nur mit harmonisiertem Standard – gestrichelt = Fallback für *Critical*, falls kein EUCC-Schema verfügbar ist (Art. 32(4)).

Warum Modul A und nicht Modul H?

Modul A – Self-Assessment

- Ermöglicht durch Anwendung harmonisierter Standards (EN 304 623/626)
- Eigenverantwortliche Konformitätsbewertung
- Kein externer Prüfer erforderlich

Modul H – Drittprüfung

- Wäre möglich, aber:
- EU-weit noch keine Notified Bodies gelistet (NANDO-Datenbank)
- Kein externer Leitfaden verfügbar

Praxisbeispiel: Linutronix IGLOS

- Kuratierte Debian-Distribution für **resiliente Embedded-Systeme**
- Entwickelt für Konformität mit dem **Cyber Resilience Act**
- **Immutable** Root-Dateisystem, **verschlüsselte** Nutzerdaten - manipulationssicher per Design
- **Secure Boot** – verifizierte Vertrauenskette
- Automatisierte **Software-Updates**
- Sicherheits-Konformität nachgewiesen durch **Drittprüfung** nach IEC 62443

IGLOS by LINUTRONIX

INDUSTRIAL GRADE LINUX OPERATING SYSTEM

Industrial IT Security
TÜV SÜD
IACS Component
Secure Product Development Lifecycle assessed & monitored according to IEC 62443-4-1
Technical Security Requirements assessed according to IEC 62443-4-2

IEC 62443 als fertig verfügbare Annährung an CRA

Zertifizierungen

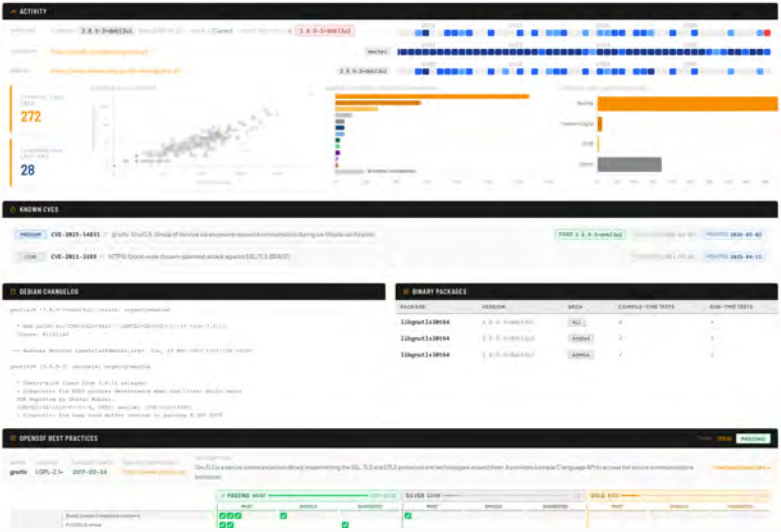
- **62443-4-1**: Secure Development Lifecycle (2023)
- **62443-4-2**: Component Security Requirements (2024)
- Ziel: State-of-the-Art Absicherung des Betriebssystems, Vorbereitung auf CRA

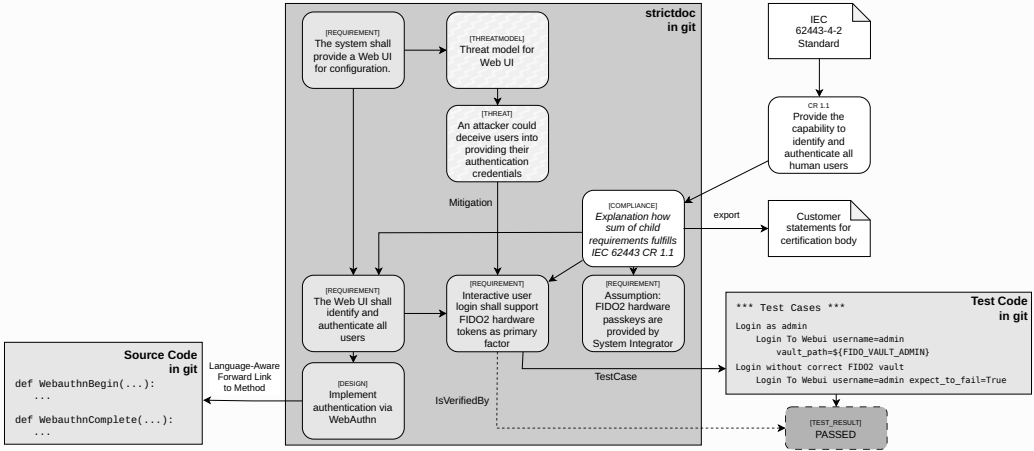
Herausforderung

- 62443 adressiert fertige OT-Komponenten, nicht wiederverwendbare Linux-Plattformen.
- Nutzerrollen, Deployment-Kontext müssen angenommen werden.
- Referenzapplikation als Stellvertreter nötig.

Solide Basis. Aber für ein generisches OS müssen viele Annahmen getroffen werden.


IGLOS
SECURE BEACON
by LINUTRONIX
IGLOS
by LINUTRONIX
INDUSTRAL
GRADE
LINUX
OPERATING
SYSTEM
RS-485
Status / Reset
Power Supply


ACTIVITY
272
28
KNOWN CVES
CVE-2025-14831
CVE-2011-3389
DEBIAN CHANGELOG
BINARY PACKAGES
OPENSSF BEST PRACTICES
PASSING
gnutls
GPL-2.0+
2017-02-14

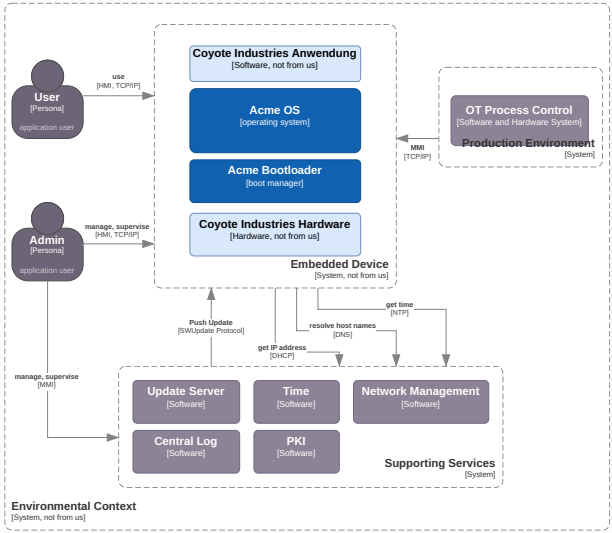

strictdoc in git
[REQUIREMENT] The system shall provide a Web UI for configuration.
[THREATMODEL] Threat model for Web UI
[THREAT] An attacker could deceive users into providing their authentication credentials
Mitigation
IEC 62443-4-2 Standard
CR 1.1 Provide the capability to identify and authenticate all human users
[COMPLIANCE] Explanation how sum of child requirements fulfills IEC 62443 CR 1.1
export
Customer statements for certification body
[REQUIREMENT] The Web UI shall identify and authenticate all users
[REQUIREMENT] Interactive user login shall support FIDO2 hardware tokens as primary factor
[REQUIREMENT] Assumption: FIDO2 hardware passkeys are provided by System Integrator
[DESIGN] Implement authentication via WebAuthn
IsVerifiedBy
TestCase
Source Code in git
def WebauthnBegin(...):
...
def WebauthnComplete(...):
...
Language-Aware Forward Link to Method
Test Code in git
*** Test Cases ***
Login as admin
Login To Webui username=admin
vault_path=${FIDO_VAULT_ADMIN}
Login without correct FIDO2 vault
Login To Webui username=admin expect_to_fail=True
[TEST_RESULT] PASSED

Und jetzt für den CRA

Wie viel der 62443-Arbeit ist für den CRA wiederverwendbar?

Trainings-Experiment: Entwicklung des fiktiven Betriebssystems **Acme OS** mit CRA typischer Konformitätsbewertung.

Schritte 1-6: Von Produkt-Klassifizierung über Risikoanalyse hin zu Konformitäts-Evidenzen


Environmental Context
[System, not from us]
User
[Persona]
application user
use
[HMI, TCP/IP]
Admin
[Persona]
application user
manage, supervise
[HMI, TCP/IP]
Coyote Industries Anwendung
[Software, not from us]
Acme OS
[operating system]
Acme Bootloader
[boot manager]
Coyote Industries Hardware
[Hardware, not from us]
Embedded Device
[System, not from us]
OT Process Control
[Software and Hardware System]
Production Environment
[System]
MMI
[TCP/IP]
Push Update
[SWUpdate Protocol]
get IP address
[DHCP]
resolve host names
[DNS]
get time
[NTP]
manage, supervise
[MMI]
Update Server
[Software]
Time
[Software]
Network Management
[Software]
Central Log
[Software]
PKI
[Software]
Supporting Services
[System]

LINUTRONIX

Von der Produktidee zur Konformität…

Schritt 1: Klassifizierung des Produkts

Ausgangspunkt

- Wiederverwendbare Linux-Plattform für industrielle Embedded-Geräte
- Zielplattform: ARM mit TrustZone
- EU-Vertrieb → **CRA**

CRA-Einordnung ??

Schritt 1: Klassifizierung des Produkts

Ausgangspunkt

- Wiederverwendbare Linux-Plattform für industrielle Embedded-Geräte
- Zielplattform: ARM mit TrustZone
- EU-Vertrieb → **CRA**

CRA-Einordnung nach 2025/2392 Anhang I

- **Important Product Class I**
- Kategorie: *Operating system*
- Kategorie: *Boot managers*

Modul A: Self-Assessment via harmonisierte Normen (2024/2847 Anh. VIII)

Vorgehen: ETSI EN 304 623 (Boot Manager) und EN 304 626 (Betriebssystem)

Schritt 1: Komponenten im Scope

Boot Manager

- ?
- ?
- ?

Betriebssystem

- ?
- ?
- ?

In Scope:

Nicht im Scope:

- ?

Schritt 1: Komponenten im Scope

Boot Manager
- ARM Trusted Firmware-A
- U-Boot (inkl. UEFI Emulation)
- shim, systemd-boot

Betriebssystem
- Kernel: Linux, OP-TEE
- Krypto: OpenSSL
- Auth: shadow, PAM
- Privileged: systemd, D-Bus
- Updates: swupdate
- Logging: linux-audit, journald

In Scope: Komponenten, die unmittelbar zu den deklarierten Produkteigenschaften beitragen.

Nicht im Scope:
- ROM (z. B. NXP HAB4) wird extern als Teil des MCU/MPU-Produkts bewertet
- Debian-minbase-Rest: Due Diligence (SBOM, Vulnerability-Monitoring, Patch-Management)

Scope-Definitionen direkt aus ETSI EN 304 623/626 übernehmen

Schritt 2: Produktbeschreibung - Intended Use

ETSI liefert Checkliste: Product Functions, Product Architecture, Trust Domains.

Beispiel: Hat das Produkt per-process CPU/Memory-Limits? Nicht als gegeben ansehen – ausdrücklich benennen, sonst fehlt es in Risikoanalyse und Mitigations.

Bestimmungsgemäße Verwendung

- Basis-OS für industrielle Embedded-Geräte
- Lieferung an OEMs zur Integration
- Inbetriebnahme durch Entwickler und Integratoren

Vorhersehbarer Fehlgebrauch

- Betrieb mit unsicheren Testschlüsseln
- Einsatz ohne Integrator-Konfiguration
- Vendor-Apps außerhalb des Immutable System

ETSI-Terminologie übernehmen, Konzepte abgleichen

LINUTRONIX

Schritt 2: Produktbeschreibung – Eigenschaften schärfen

Capabilities bestimmen Angriffsfläche und geben Hinweis auf Risiko und benötigtes Security Profil.

Security Capabilities (Beispiele)

Bestimmen Use Case & Sicherheitsprofil

- Secure Boot
- Software Update
- Hardware-Security

Functional Capabilities (Beispiele)

Vergrößern die Angriffsfläche

- Persistenter Speicher
- Benutzerkonten
- Integration mit Remote-Diensten

EN 304 623 (Boot Manager): Capabilities – Use Case (Tab. 4.6.4–1) – Security Profile

EN 304 626 (OS): Capabilities steuern Applicability einzelner Mitigations (z. B. Update-Fähigkeit)

Capabilities vor der Risikoanalyse festlegen – gilt für Boot Manager und Betriebssystem

Schritt 3: Risikoanalyse. Zwei Normen – zwei Prozesswege.

EN 304 623 Boot Manager

1. Produkteigenschaften in Use Case überführen (nach Tab. 4.6.4–1; z. B. UC-HW)
2. Use Case ergibt Security Profile (z. B. Profil HIGH)
3. Benötigte Mitigations aus Profiltabelle (Clause 5.3)

EN 304 626 Betriebssystem

1. Risikofaktoren bewerten (Risiko durch mehrere Benutzer, Risiko durch sensible Daten, …) nach Clause 4 + Annex B. Use Cases sind keine alleingültige Abkürzung.
2. Security Profile = Menge der Risikofaktorwerte. Abkürzung via Use Case: B.5–Mapping RF-Werte ➞ Profil.
3. Benötigte Mitigations aus Profiltabelle (Clause 6.3)

Gemeinsames Ziel: Benötigte Mitigations feststellen. Nebenprodukt: Threat-Model.

Hersteller **muss** Risikoanalyse durchführen. Hersteller **kann** Annex B verwenden.

Schritt 4: Anforderungen / Mitigations festlegen

Boot Manager: Für alle Requirments in Abschnitt 5

1. Applicabilty-Kriterium 1: Auswahl nach Security Profil (z.B. "Applies to MEDIUM or HIGH")
2. Applicabilty-Kriterium 2: Auswahl nach Capability (z.B. "Applies to boot managers with update capability")
3. Mitigation-Set = alle Anforderungen, für die beides zutrifft

Betriebssystem: Herleitung aus Risikofaktoren via Threats

1. Für alle anwendbaren Threats die Mitigationen nehmen, die zum Absenken des Threat-Risiko auf Low benötigt werden.
2. Mitigation-Set = alle Anforderungen die von allen Threats genannt werden)
3. ergibt entweder standard Security Profile oder benutzerdefiniertes Security Profile

Sowohl anwendbare (Stubs) und nicht anwendbare (Begründung) dokumentieren.

Security Profile ist normativ – Annex B ist ein mögicher Rechenweg.

LINUTRONIX

Schritt 4: Anforderungen / Mitigations festlegen - Beispiele

Für jedes Requirement: Wird es aufgrund Security Profile oder Capabilities benötigt?
Applicability **ja/nein** mit Begründung dokumentieren. Insgesamt ~100 Mitigations pro Standard – hier nur Beispiele.

Boot Manager Beispiele (bei Acme OS 71 von 99)

- **REQ-BM-INT-002** – Integritätsschutz
  Applicable: Profil MEDIUM/HIGH
- **REQ-BM-SU-003** – Sichere Updates
  Applicable: Update-Capability

OS Beispiele (bei Acme OS 49 von 74)

- **MI-MMAC** – Speicherzugangskontrolle
  Applicable: Has user accounts
- **MI-SCHM** – Sichere Updates (Medium)
  Applicable: Has update capability

Praxistipp: Traceability Matrix mit StrictDoc → hEN Compliance-Dokument

Schritt 5: Anforderungen implementieren

1. **Jede benötigte ETSI-Anforderung** in die Traceability-Matrix aufnehmen
2. **Auch nicht benötigte Anforderungen** – mit Begründung, warum nicht anwendbar oder nicht benötigt
3. **Für jede benötigte Anforderung:** auf die tatsächlich implementierenden Produktanforderungen verlinken
4. **Konformitätsstatement:** in Prosa beschreiben, wie die Gesamtheit der verlinkten Produktanforderungen die ETSI-Anforderung erfüllt

Werkzeug: StrictDoc

Traceability-Matrix, Links und Konformitätsstatements direkt im Dokument – nachvollziehbar, versioniert, maschinenlesbar.

Vollständigkeit: jede Anforderung muss eine Entscheidung haben – implementiert oder begründet ausgeschlossen

Schritt 6: Konformitätsbewertung

- Assessment-Kapitel definieren pro Anforderungen Prüfziel, Aktivitäten, Pass / Fail Bedingung und Beleg.
- Test mit Konformitäts-Test-Suite oder produktspezifischen Systemtests.
  - Manche Anforderungen lassen sich besser auf Systemebene testen
  - Manche Anforderungen lassen sich nur rekursiv pro integrierter Komponente testen (Linux hat gehärtete Datenstrukturen, OpenSSL hat gehärtete Datenstrukturen, etc., …)
  - Manche Anforderungen sind reine Dokumentaitonsartefakte (sollte die Ausnahme sein, CRA will Verhaltensprüfung)
- Manche Bewertungen lassen Interpretationsspielraum
  z. B. ob die Art und Abdeckung durch statische Code-Analyse ausreichend ist.

Sollte für Marktaufsichtsbehörde nachvollziehbar und reproduzierbar sein

Letzter Schritt: Restrisiko dokumentieren

Was ist Restrisiko?

Bedrohung anwendbar. Mitigation implementiert. Assessment: PASS.
Aber: Risiko durch Produkt allein nicht eliminiert.

Beispiel 1

Microarchitectural Side Channels lassen sich nicht vollständig in Software mitigieren.

Beispiel 2

Eine bekannte Schwachstelle ist nicht gepatched. ”Compensating control” dokumentieren.

Restrisiko am besten je nicht-vollständig mitigiertem Threat aufschreiben.

Was ist kein Restrisiko

⚠ Verwechslung 1

Fehlende Mitigation ohne externe Kompensations-maßnahme

- Mitigation nachrüsten
- Alternative wählen (wo Profil “or” anbietet)
- Nachweis über andere Methode erbringen

⚠ Verwechslung 2

Deployment-Kontext als Restrisiko formulieren

“*Das OS wird ohne Schlüsseleinbrennung betrieben*” ist kein Restrisiko des OS-Herstellers.
Es ist eine verletzte **Assumption** (Annex B.3).
Zuständig: der Integrator.

Norm-Begriff: **Assumption**, nicht Residual Risk.

Compliance-Lücke und Restrisiko sind getrennte Konzepte

Fazit

- IEC 62443 Prozessschritte und Artefakte sehr ähnlich
- ETSI hEN: Anforderungen viel produktspezifischer, besser geeignet für Komponenten „out-of-context“
- Kritischer Pfad bei OS aus OSS Komponenten: Testabdeckung und Testnachweis

Anhang: Normativ geforderte Artefakte

Artefakt	Normative Quelle	Inhalt / Pflicht
Cybersecurity-Risikoanalyse	CRA Art. 13(2)–(4); EN 304 623 Kl. 5.2.1	Analyse von Assets, Betriebsumgebung, Bedrohungen; Aufnahme in Techn. Dokumentation (CRA Art. 31)
SBOM	CRA Anh. I, Teil II, (1)	Maschinenlesbares Format, mindestens Top-Level-Abhängigkeiten
Bedrohungsanalyse	CRA Art. 13(3); EN 304 623 Kl. 5.2.2	Bedrohungen aus Annex B für konkreten Use Case + Betriebsumgebung prüfen – “shall consider the threats... applicable to the declared use case”
Security Profile + Begründung	EN 304 626, Kl. 6.3.1	Auswahl aus Alternativen (“or”-Mitigationen) dokumentieren – als Konformitätsnachweis
Restrisiko	EN 304 626, Kl. 6.1.3.4	Für jede nicht vollständig mitigierte Bedrohung begleitend dokumentieren

Anhang: Mögliche Prüfbelege

1. **Test Protokolle**
   - Test Reports. Empfehlung für Systemtests: Robot Framework, LAVA, QEMU, manueller Testsessions
   - Protokolle manueller Testsessions
   - Logdateien
   - Ergebnis von Scannern (z.B. Vulnerabilty Scanner)

2. **Dokumente und Dateien direkt verlinken**
   - Quellcode, formale Spezifikation
   - Handbücher für Benutzer und Integratoren
   - Konfigurationsdateien verlinken, z.B. /boot/config
   - Bedrohungsmodell + Risikoanalyse als Prozess-Evidenz

3. **Prozessartefakte**
   - Checklisten
   - Reviews in GitLab, github, Mailingliste
   - Kommentare im Issue-Tracker

3. **Externe Nachweise**
   - Externe dynamische Tests: z.B. Kernel CI, syzcaller, ... z.B. syzbot coverage, smatch code analysis patches
   - Externe Handbücher: z.B. man pages, Hardening Guidelines, ...
   - Gültigkeit unklar. Zukunft: CRA Art. 25 FOSS-Attestierungsprogramme. Siehe z.B. Open Regulatory Compliance Working Group - CRA Attestations.

Einschätzung: Marktaufsichtsbehörde kann im Zweifelsfall alle Artefakte inklusive Source-Code verlangen.

StrictDoc verbindet hEN Requirements, Compliance, Produktanforderungen und Source Code.

Was, wenn ich gar kein eigenes Betriebssystem baue?

Art. 13(5) CRA: Hersteller müssen "*die gebotene Sorgfalt walten lassen*", wenn sie von Dritten bezogene Komponenten integrieren, sodass diese die Cybersicherheit des Produkts nicht beeinträchtigen.

- Verantwortung bleibt beim Hersteller – auch bei fertiger Distribution oder fertigem BSP
- Nachweis kommt entweder vom OS-Lieferanten oder muss selbst erbracht werden

→ freiwillige Attestierungsprogramme (Art. 25 CRA) könnten Nachweise für FOSS künftig erleichtern

Kein eigenes OS zu bauen befreit nicht von der Sorgfaltspflicht – nur die Nachweisquelle ändert sich

LINUTRONIX

2026-07-07

Thank you for your attention

Presented to you by:

Tobias Deiminger, Simone Weiß