Security-by-Design

Wie der CRA Embedded-Linux und Systems/Security-Engineering zusammenbringt

Durch den Cyber Resilience Act (CRA) wird ein gutes Systems- und Security-Engineering auch auf Ebene des Embedded-Linux-Systems obligatorisch. Das erscheint auf den ersten Blick möglicherweise sehr aufwändig. Im Endeffekt sorgt es dafür, dass Linux-basierte Systeme nicht nur sicherer, sondern auch effizienter und über den Lebenszyklus kostengünstiger (weiter-)entwickelt und betrieben werden können.

Von Heike Jordan,
emlix

Der Cyber Resilience Act (CRA) tritt im Dezember 2027 in Kraft. Gemessen an der Disruption, die er mit sich bringt, ist die Zeit eher knapp bemessen. Gleichzeitig sind noch nicht alle Regularien für die Umsetzung ausformuliert. Zwei Dinge werden jedoch in jedem Fall zu berücksichtigen sein:

- die grundsätzliche Forderung nach einem Security-by-Design-Ansatz
- die Forderung nach einer Lifecycle Maintenance, die sicherstellt, dass der einmal erreichte Security-Level über den Lebenszyklus aufrechterhalten wird.

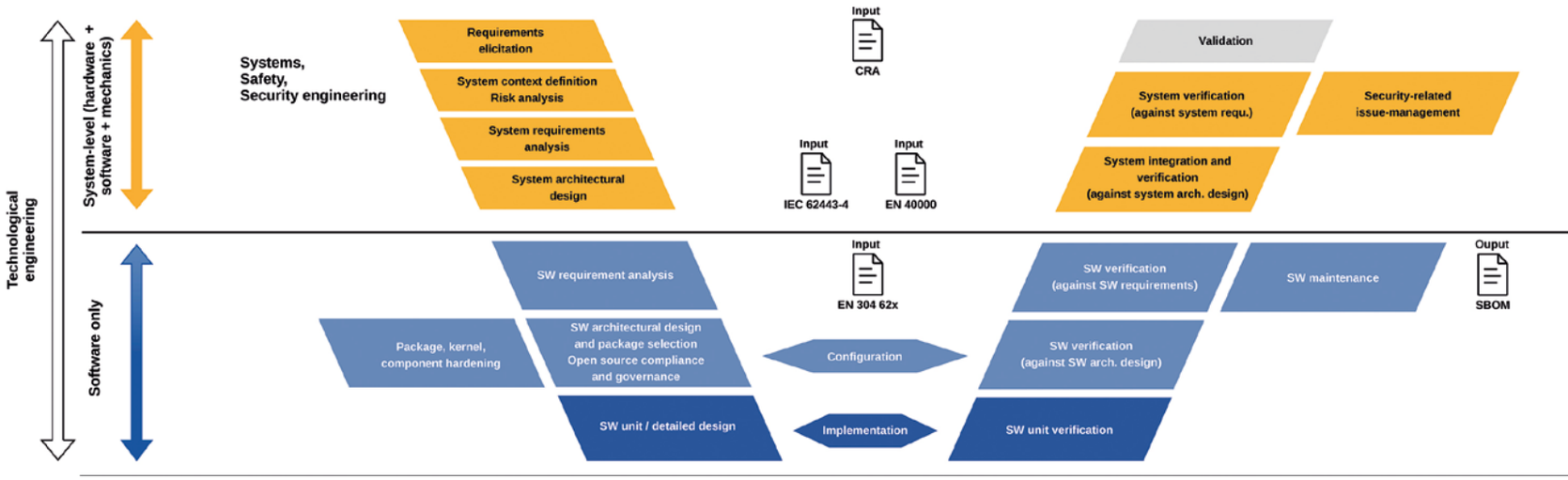
Beides wird auch in bereits existierenden industriellen Security-Normen gefordert, an die der CRA voraussichtlich die Umsetzung delegiert, beispielsweise die IEC 62443.

Satz (1) in Anhang 1 des CRA fordert, dass angesichts spezifischer Risiken ein angemessenes Cybersicherheitsniveau gewährleistet werden muss. Umgekehrt bedeutet dies, dass es nicht die eine generische CRA-Umsetzung geben kann, die eine abzuhakende Checkliste liefert.

Selbst Secure Boot zur Verhinderung der Ausführung von Schadsoftware ist, anders als es bisweilen dargestellt wird, vom CRA nicht zwingend gefordert, wenn er auch in den meisten Fällen Teil des Security-Konzeptes sein wird. Ob das der Fall ist, muss sich jedoch aus einer produktspezifischen Risiko-Analyse und -Bewertung ergeben, die standardmäßig die zu schützenden Assets und die Angriffsvektoren ermittelt und das damit einhergehende Risiko bewertet, um zu geeigneten Mitigationen zu kommen – oder auch zu der gut begründeten Entscheidung, dass gegen ein spezifisches Risiko keine Maßnahmen ergriffen werden müssen.

Damit fordert der CRA an sich einen klassischen Systems-Engineering-Ansatz. Und hier kommen wir zu einer Besonderheit der Nutzung von Embedded-Betriebssystemen im Allgemeinen und von Embedded Linux im Besonderen. Bisher wurde ein Embedded-Linux-System (ebenso wie andere Full-featured-Betriebssysteme) vom Engineering in der Produktentwicklung häufig eher stiefmütterlich behandelt.

Nicht selten kam es als Bestandteil der Hardware-Plattform in die Produktentwicklung und wurde – durchaus auch über den gesamten Produktlebenszyklus – nur wenig angepasst. Es wurde in vielen Fällen als eine geschlossene Komponente oder quasi zur Hardware gehörende Firmware betrachtet. Im Ergebnis verfügen einige bereits seit Jahren im Feld befindliche Systeme weder über eine nachvollziehbare Software Bill of Ma-



Embedded-Linux Security-by-Design im V-Modell (Bild: emlix)

terial (SBOM) noch wurden sicherheits- oder funktionsrelevante Software-Updates eingespielt. Die Hersteller und Inverkehrbringer wissen über die Betriebssystemebene bisweilen nicht viel mehr, als dass es sich eben um ein Linux-System handelt.

Hier verändert sich aktuell sehr viel. Nach wie vor gibt es jedoch den Wunsch, das Embedded-Linux-System idealerweise »CRA-ready«vom Hardware-Hersteller zu beziehen und an diesen auch die Lifecycle Maintenance zu delegieren. Das ist insofern nachvollziehbar, dass sich einiges auf den ersten Blick standardisiert vorbereiten lässt: beispielsweise eine belastbare SBOM für das Embedded Linux Board Support Package, aktuelle Komponenten-Versionen – idealerweise die aktuellste LTS-Version –, ein vorbereiteter Secure-Boot-Mechanismus und ein Template-artig vorbereitetes Update-Konzept basierend auf Standardfunktionen.

Dies ersetzt jedoch nicht das Security Engineering spezifisch für das jeweilige Produkt und seinen Einsatzkontext sowie die Adaption der vorbereiteten Mechanismen. Das lässt sich am Beispiel eines Secure-Update-Konzeptes gut illustrieren. Natürlich drängt sich eine Signatur-Prüfung der im Update aufzuspielenden Software auf, ebenso wie ein A/B-Pendel-Update. Dieses berücksichtigt aber nicht, durch wen (Rollenkonzept), in welcher Umgebung (gesicherte Umgebung, frei zugänglich), über welche Infrastruktur (OTA, USB-Stick,...) und in welcher Phase des Produktlebenszyklus (Update bei Erstinbetriebnahme, reguläres Update im Feld) das Update erfolgt. Und es berücksichtigt in keinem Fall, welche Assets dabei in jedem Fall gegen welche Angriffsvektoren geschützt werden müssen. Um angemessen auf ein Risiko reagieren zu können, muss dieses jedoch bekannt und es muss in seiner Kritikalität bewertet sein (siehe den o.g. Satz (1) in Anhang 1 des CRA).

Die eingangs erwähnte Risikoanalyse findet dabei nicht auf der Ebene des Betriebssystems statt, sondern auf der des Gesamtsystems. Die daraus ermittelten Anforderungen an das System in Bezug auf die Cybersicherheit müssen im Gesamtsystem umgesetzt werden. Dies ist ein weiterer Aspekt, warum es für ein Embedded-Linux-System nicht die eine »richtige«CRA-Umsetzung geben kann: Ein Security-Risiko kann auf unterschiedlichen Ebenen mitigiert werden (Defence-in-Depth-Konzept). Zum Beispiel kann die Absicherung des USB-Interface dadurch erfolgen, dass diese deaktiviert wird.

In aller Regel gilt zumindest für einige Security Requirements, dass sie nicht isoliert auf einer »Ebene«des Software-Stacks erfüllt werden können. Die Maßnahmen erstrecken sich beispielsweise auf Applikations- und Betriebssystemebene. Auch hier ist ein Secure-Update-Konzept ein gutes Beispiel.

Der Security-by-Design-Ansatz ist aber bei weitem nicht nur Fluch. Er kann durchaus auch Segen sein. Je nach Produkt, Anwendungs- und Integrationskontext und diversen weiteren »Umweltfaktoren«kann die Risikoanalyse auch zu einer validen Argumentation führen, warum bestimmte Maßnahmen, die regelmäßig als »Must have«im Kontext des CRA aufgerufen werden, nicht ergriffen werden müssen. In letzter Konsequenz gilt das, wie eingangs erwähnt, auch für einen Secure-Boot-Mechanismus.

Für viele industrielle Komponenten und Geräte, die nicht in kritischen Kontexten zum Einsatz kommen, ergibt sich am Ende ein reduziertes Set an Mitigationen, abgestimmt auf konkrete Use Cases. Die Investition in eine Risikoanalyse lohnt sich aber nicht nur technisch, sie macht sich auch wirtschaftlich bezahlt. Dies gilt insbesondere mit Blick auf die Lifecycle Maintenance, die gemäß CRA obligatorisch ist. Bei den bei Embedded-Geräten gerne auch oberhalb von zehn Jahren liegenden Produktlebenszeiten sinken bei reduzierter und »aufgeräumter«Betriebssystemebene die Gesamtkosten (TCO) erheblich.

Aus einem konsequenten Security Engineering ergibt sich grundsätzlich ein gehärtetes System: Nur was per Requirement gefordert wird, soll im System enthalten sein. Dies gilt für Hardware-Schnittstellen (zum Beispiel USB) ebenso wie für den Software-Stack und hier die Komponenten, aus denen sich die Betriebssystemebene zusammensetzt. Mit dieser Härtung besteht das Gesamtsystem aber auch aus Sicht der Maintenance aus einem Minimum an zu wartenden »Bestandteilen".

Mit Blick auf ein Embedded-Linux-System sorgt die bewusste Entscheidung für Komponenten und ihre Versionen ebenso wie die sichere Konfiguration dieser Softwarepakete für eine deutliche Reduktion des Mainte-

nance-Aufwandes. Die nachvollziehbare SBOM ist nicht nur eine Forderung im Sinne der Open Source Compliance, sondern auch eine nahezu obligatorische Forderung der allermeisten Unternehmen, die die Systeme in den Verkehr bringen. Die SBOM ist zusätzlich das Herzstück der Open Source Governance mit dem Ziel, hinsichtlich Risiken und Maintenance für die verwendete Software Verantwortung übernehmen zu können (dies fordert im Übrigen auch die NIS2). Wurde dies alles in der Entwicklung berücksichtigt, lassen sich die Sorgfaltspflichten (übrigens auch im Sinne des aktualisierten Produkthaftungsgesetzes) deutlich effizienter erfüllen. Für ein Embedded-Linux System heißt das, dass für die SBOM regelmäßig alle relevanten potenziellen Quellen für gemeldete Security Issues geprüft werden müssen. Das ist in erster Linie die NVD-Database, hinzu kommen aber auch noch diverse andere Quellen, die für einen guten Zugang zu neu gefundenen Schwachstellen sorgen.

Weitgehend automatisieren lässt sich der Filterprozess der gefundenen Security-Probleme gegen die SBOM, die Konfiguration des Systems und einige andere Parameter. Tatsächlich tiefgehendes Linux- und Kernel Know-how verlangt vor allen Dingen das normativ geforderte Expert Assessment, das wiederum hoch-spezifisch für das konkrete System und seinen Einsatzkontext ist. Dieser Aufwand reduziert sich quasi linear mit der Zahl der tatsächlich zu überwachenden Software-Komponenten.

Fazit: Der CRA sorgt dafür, dass Linux-basierte Systeme nicht nur sicherer, sondern auch effizienter und über den Lebenszyklus kostengünstiger (weiter-)entwickelt und betrieben werden können. (ku)