CYBER RESILIENCE ACT

Reifegradtest für Softwareentwicklung

Der CRA richtet sich an Produkte mit digitalen Elementen und macht sichtbar, wie in deren Lebenszyklus bekannte Prinzipien der Softwareentwicklung tatsächlich umgesetzt werden.

Der Cyber Resilience Act (CRA) [1] ist in den letzten Monaten schnell von einem abstrakten EU-Vorhaben zu einem konkreten Gesprächsthema geworden. Er wurde lange ignoriert, taucht nun aber in immer mehr Management-Meetings, Projektplänen und Angebotsgesprächen auf. Bußgelder, CE-Kennzeichnung, externe Audits und Herstellerhaftung sorgen für spürbare Unruhe. Der Ton schwankt dabei zwischen Alarmismus und Verdrängung. Für viele fühlt sich der CRA an wie ein externer Eingriff, der unerwartet tief in die eigenen, etablierten Abläufe hineinreicht.

Diese Nervosität ist kein Zufall. Der CRA unterscheidet sich in einem maßgeblichen Punkt von früheren regulatorischen Initiativen: Er richtet sich nicht an einzelne Branchen, sondern adressiert horizontal alle Produkte mit digitalen Elementen. Damit fällt ein großer Teil dessen, was heute entwickelt, betrieben und ausgeliefert wird, in seinen Anwendungsbereich. Allein diese Breite sorgt für Verunsicherung, weil klare Abgrenzungen fehlen, an denen man sich orientieren könnte.

Hinzu kommt, dass der CRA erstmals sehr deutlich die Verantwortung beim Hersteller verortet und nicht bei Betreibern, bei Kunden oder irgendwo in der Lieferkette. Für viele Unternehmen ist das eine neue Perspektive. Sicherheit war lange etwas, das man bereitgestellt, empfohlen oder optional ergänzt hat. Der CRA verschiebt diesen Fokus. Er macht Sicherheit zu einer Eigenschaft des Produkts selbst und damit zu etwas, das nachweisbar vorhanden sein muss.

Ein weiterer Grund für die aktuell herrschende Unruhe liegt in der Diskussion um Sanktionen und Durchsetzung. Der Cyber Resilience Act ist ein Rechtsakt mit klaren Konsequenzen. Hohe Bußgelder bei Verstößen werden häufig als Erstes genannt, ebenso die Möglichkeit, Produkte vom Markt zu nehmen oder ihren Vertrieb einzuschränken. Diese Konsequenzen verändern die Wahrnehmung, auch wenn diese Maßnahmen nicht automatisch greifen und an Bedingungen geknüpft sind. Cybersicherheit wird somit zur Voraussetzung für den Zugang zum europäischen Markt. Allein diese Perspektive reicht aus, um insbesondere in Unternehmen, die bislang wenig Berührung mit Produktsicherheitsrecht hatten, Aufmerksamkeit zu erzeugen.

Verstärkt wird die Unsicherheit durch das Thema der externen Bewertung. Der CRA bringt das Konzept der Konformitätsbewertung stärker ins Spiel. Je nach Produktkategorie (siehe Bild 1) kann das bedeuten, dass externe Stellen prüfen, ob ein Produkt die geforderten Eigenschaften erfüllt. Für viele technische Teams ist das ungewohnt. Architekturentscheidungen, Update-Strategien oder der Umgang mit Abhängigkeiten waren bislang interne Angelegenheiten. Nun besteht die Aussicht, dass genau diese Entscheidungen nachvollziehbar begründet werden müssen. Es geht dabei nicht um die Bewertung der Technologie, sondern um die Bewertung der Angemessenheit der Lösung.

In dieser Situation entstehen schnell verkürzte Narrative. Der CRA wird als Bürokratiemonster oder Innovationsbremse bezeichnet. Gleichzeitig kursieren vereinfachende Aussagen, wonach man mit ein paar zusätzlichen Dokumenten oder einem neuen Tool „CRA-ready“ werde. Beides greift zu kurz. Die aktuelle Unruhe ist weniger Ausdruck überzogener Anforderungen als vielmehr ein Hinweis darauf, dass vielen Organisationen noch unklar ist, worauf der Cyber Resilience Act tatsächlich abzielt und welche Fragen er wirklich stellt.

Genau hier setzt die nachfolgende nüchterne Betrachtung an. Es ist entscheidend, zu klären, was der CRA eigentlich ist, welches Problem er lösen soll und welche Erwartungen er an Produkte stellt. Erst vor diesem Hintergrund wird verständlich, warum das Thema nicht nur Juristen und Compliance-Verantwortliche betrifft, sondern tief in unsere technischen Entscheidungen hineinwirkt.

Was der Cyber Resilience Act ist und welches Problem er lösen soll

Der Cyber Resilience Act adressiert ein strukturelles Problem, das sich über Jahre aufgebaut hat. Digitale Funktionen sind heute fester Bestandteil nahezu aller technischen Produkte. Software steuert Maschinen, verbindet Geräte, wertet Daten aus und ermöglicht Fernzugriff und Updates. Gleichzeitig ist genau diese Software in vielen Fällen unzureichend abgesichert, schlecht wartbar oder über ihren Lebenszyklus hinweg sich selbst überlassen. Sicherheitslücken bleiben ungepatcht, Zuständigkeiten sind unklar und Verantwortung wird entlang der Lieferkette weitergereicht.

Der CRA setzt genau an dieser Stelle an. Sein grundlegender Anspruch ist schnell formuliert: Produkte mit digitalen Elementen, die auf dem europäischen Markt bereitgestellt werden, sollen ein angemessenes Maß an Cybersicherheit aufweisen. Und zwar nicht nur zum Zeitpunkt ihrer Markteinführung, sondern über die gesamte Zeit, in der sie ▶

realistisch genutzt werden. Der CRA verfolgt damit einen produktrechtlichen Ansatz. Er behandelt Cybersicherheit als Eigenschaft des Produkts selbst. Genau wie mechanische Sicherheit, elektrische Sicherheit oder funktionale Zuverlässigkeit wird auch Cybersicherheit zu einem Kriterium, das ein Produkt erfüllen muss, um überhaupt in Verkehr gebracht werden zu dürfen.

Der CRA ist dabei bewusst technologieneutral formuliert. Er schreibt keine konkreten Programmiersprachen, Frameworks oder Sicherheitsmechanismen vor. Stattdessen definiert er Schutzziele und Erwartungen an das Ergebnis. Produkte sollen so gestaltet sein, dass sie Risiken angemessen berücksichtigen, Angriffsflächen minimieren, sicher aktualisiert werden können und bekannte Schwachstellen systematisch behandelt werden. Diese Offenheit soll sicherstellen, dass Verantwortung nicht durch formale Erfüllung einzelner Maßnahmen umgangen werden kann. Entscheidend ist nicht, welches Werkzeug eingesetzt wird, sondern ob das Produkt als Ganzes die geforderten Eigenschaften aufweist.

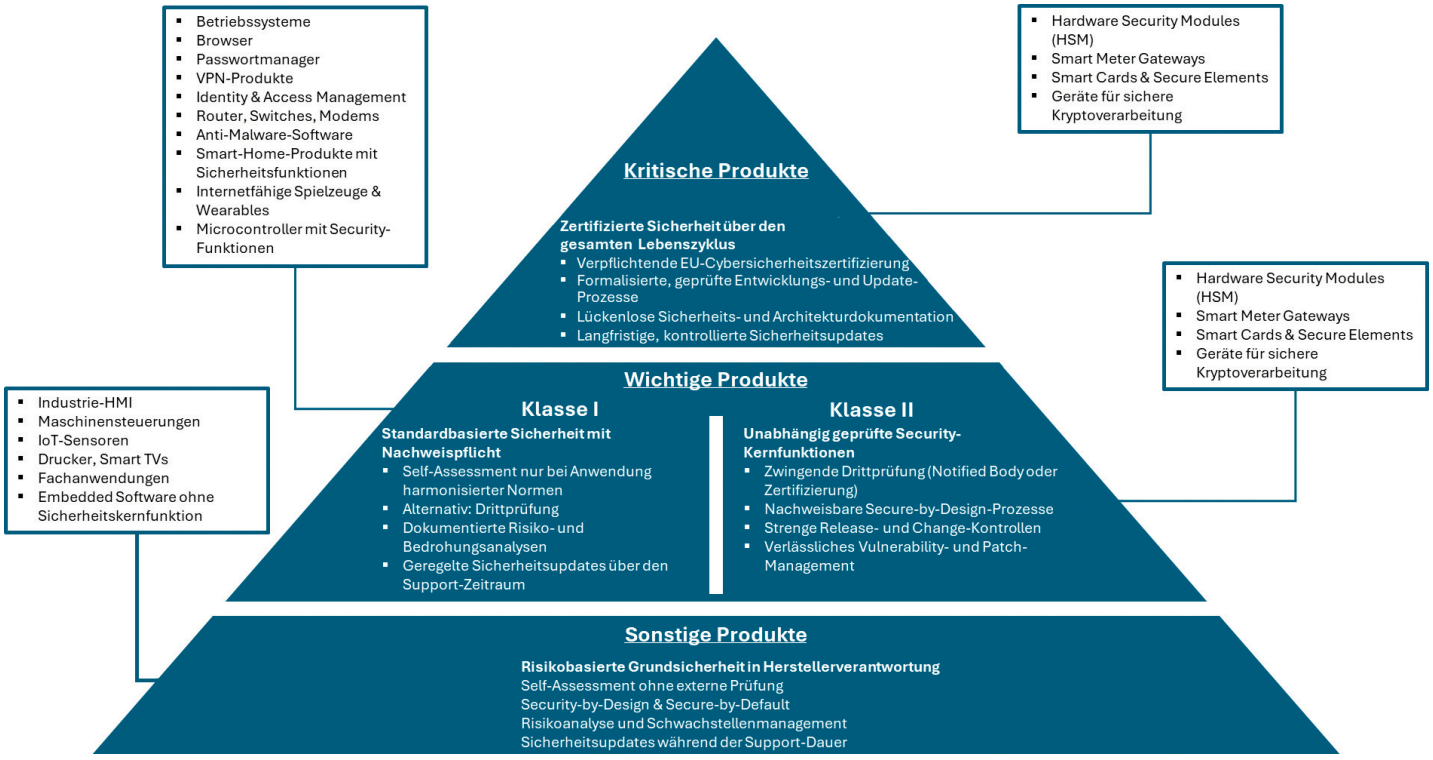
Dabei ist wichtig zu verstehen, dass der CRA keine vollständige Sicherheit verspricht und auch nicht verlangt. Er geht ausdrücklich von einem risikobasierten Ansatz aus. Produkte sollen nicht frei von allen denkbaren Schwachstellen sein, sondern so entwickelt werden, dass Risiken systematisch erkannt, bewertet und angemessen behandelt werden. Diese Angemessenheit ist entscheidend. Sie eröffnet Spielräume, verlangt aber im Gegenzug Begründbarkeit. Entscheidungen müssen nachvollziehbar sein. Annahmen über Nutzung, Einsatzumgebung und Bedrohungen dürfen nicht implizit bleiben oder im Nachgang festgelegt werden. Sie müssen bewusst getroffen werden.

Der CRA zwingt Hersteller dazu, Cybersicherheit als dauerhafte Aufgabe zu verstehen, die Entwicklung, Betrieb und Wartung verbindet. Der Fokus verschiebt sich weg vom einzelnen Release hin zum gesamten Lebenszyklus. Genau dieser Perspektivenwechsel ist es, der viele bestehende Strukturen und Prozesse infrage stellt und für Unsicherheit sorgt, denn er berührt nicht nur Sicherheitsfunktionen, sondern grundlegende Fragen der Produktentwicklung: Wie lange unterstützen wir ein Produkt? Wie gehen wir mit Abhängigkeiten um? Wer entscheidet, welches Risiko akzeptabel ist?

Um diese Fragen einordnen zu können, ist der CRA nicht als Sammlung einzelner Pflichten zu betrachten, sondern als Rahmen, der Verantwortung neu verteilt. Er verlangt nicht, dass alles neu erfunden wird. Er verlangt, dass bekannte Prinzipien ernst genommen, systematisch angewendet und über den gesamten Lebenszyklus hinweg getragen werden. Vor diesem Hintergrund wird auch verständlich, warum der CRA und seine Auswirkungen tief in die technische Praxis hineinreichen.

Was der CRA konkret fordert und warum das vielen Organisationen wehtut

Die grundlegenden Anforderungen des Cyber Resilience Act umfassen knapp zwei DIN-A4-Seiten und finden sich als Anhang I in den von der EU veröffentlichten Regularien [2]. Genau darin liegt eine der Ursachen für die aktuelle Verunsicherung. Der CRA beschreibt keine festen Maßnahmen, die sich einfach abarbeiten lassen, sondern formuliert Erwartungen an das Ergebnis. Produkte mit digitalen Elementen sollen so gestaltet, entwickelt und betrieben werden, dass sie über ihren gesamten Lebenszyklus hinweg ein angemessenes Maß an Cybersicherheit aufweisen. Diese Ergebnisorientierung ist



Klassifizierung von Produkten nach dem CRA und dessen Anforderungen (Bild 1)

für viele ungewohnt, weil sie sich nicht auf einzelne technische Artefakte und Checklisten reduzieren lässt.

Im Kern verlangt der CRA drei Dinge gleichzeitig. Erstens müssen Sicherheitsrisiken systematisch berücksichtigt werden. Risiken dürfen also nicht zufällig oder reaktiv behandelt werden, sondern müssen bewusst identifiziert, bewertet und adressiert werden. Zweitens müssen Produkte so ausgelegt sein, dass sie über einen relevanten Zeitraum sicher betrieben werden können. Dazu gehören insbesondere Update-Fähigkeit, der Umgang mit Schwachstellen und die Möglichkeit, Sicherheitsprobleme auch nach der Markteinführung zu beheben. Drittens müssen diese Eigenschaften nachvollziehbar sein. Hersteller müssen in der Lage sein, darzulegen, wie sie zu ihren Entscheidungen gekommen sind und warum diese Entscheidungen aus ihrer Sicht angemessen sind.

An dieser Stelle entsteht häufig der Eindruck, der CRA verlange vor allem mehr Dokumentation. Tatsächlich spielt Dokumentation eine Rolle, aber nicht als Selbstzweck. Sie ist das Mittel, um technische Entscheidungen nachvollziehbar zu machen. Der CRA geht implizit davon aus, dass Sicherheit nicht zufällig entsteht, sondern das Ergebnis bewusster Gestaltung ist. Wo Entscheidungen bewusst getroffen werden, lassen sie sich erklären. Wo sie sich erklären lassen, können sie geprüft werden. Genau diese Prüfbarkeit ist es, die für viele Teams neu ist und als belastend empfunden wird.

Ein weiterer Punkt, der häufig als schmerzhaft wahrgenommen wird, ist der Fokus auf den Lebenszyklus. Der CRA denkt Produkte nicht als abgeschlossene Projekte, sondern als Systeme, die über Jahre hinweg genutzt werden. Sicherheit endet nicht mit dem Release. Hersteller müssen sicherstellen, dass bekannte Schwachstellen behandelt werden können, dass Updates bereitgestellt werden und dass Nutzer über relevante Sicherheitsaspekte informiert sind. Diese Erwartung kollidiert mit vielen etablierten Entwicklungsmodellen, in denen der Schwerpunkt auf der Markteinführung liegt und Wartung als nachgelagerte Aufgabe betrachtet wird.

Besonders deutlich wird diese Verschiebung beim Thema Schwachstellenmanagement. Der CRA geht davon aus, dass Schwachstellen unvermeidlich sind. Er fordert nicht, dass Produkte fehlerfrei sind, sondern dass Hersteller in der Lage sind, mit Fehlern umzugehen. Dazu gehört, Schwachstellen zu erkennen, ihre Relevanz einzuschätzen und angemessene Maßnahmen zu ergreifen. In manchen Fällen wird eine solche Maßnahme ein Patch sein, in anderen eine Konfigurationsanpassung oder eine Risikobewertung, die begründet, warum keine unmittelbare Maßnahme erforderlich ist. Entscheidend ist nicht die einzelne Reaktion, sondern der dahinterliegende Prozess. Für viele ist das ungewohnt, weil es technische, organisatorische und wirtschaftliche Aspekte miteinander verknüpft. Genau diese Verknüpfung ist es jedoch, die erklärt, warum der CRA so tief in bestehende Strukturen eingreift und warum er als unbequem empfunden wird.

Warum der CRA kein Compliance-Thema ist, sondern ein Entwicklerthema

An diesem Punkt entsteht häufig der Reflex, den CRA als klassisches Compliance-Thema einzuordnen. Die Anforderungen wirken abstrakt, die Begriffe kommen aus dem Produktrecht und die Konsequenzen werden in Form von Bußgeldern, Marktüberwachung oder Konformitätsbewertung beschrieben. Es liegt nahe, das Thema an Rechtsabteilungen oder Informationssicherheitsbeauftragte zu delegieren. Genau hier liegt jedoch ein zentraler Denkfehler. Der CRA ist kein Regelwerk, das sich allein durch organisatorische Maßnahmen erfüllen lässt. Er greift dort ein, wo Produkte technisch gestaltet werden, und damit unweigerlich dort, wo Entwickler Entscheidungen treffen.

Der Grund dafür ist einfach: Software ist kein Beiwerk mehr, sondern integraler Bestandteil des Produkts. Funktionen, Schnittstellen, Update-Mechanismen und Abhängigkeiten bestimmen maßgeblich, wie sicher ein Produkt über seinen Lebenszyklus hinweg betrieben werden kann. Diese Eigenschaften entstehen nicht durch Richtlinien und Prozesshandbücher, sondern im Code, in der Architektur und in den Entwicklungsprozessen.

Für Entwickler bedeutet das, dass Entscheidungen, die bislang als rein technische Details galten, plötzlich eine andere Bedeutung bekommen. Die Wahl einer Architektur beeinflusst, ob Sicherheitsupdates sauber getrennt von Funktionserweiterungen ausgeliefert werden können. Die Struktur eines Build- und Release-Prozesses entscheidet darüber, ob Änderungen nachvollziehbar und reproduzierbar sind. Der Umgang mit Abhängigkeiten bestimmt, wie schnell auf neu entdeckte Schwachstellen reagiert werden kann. All das sind klassische Entwicklerentscheidungen, die im CRA-Kontext zu prüfbaren Eigenschaften eines Produkts werden.

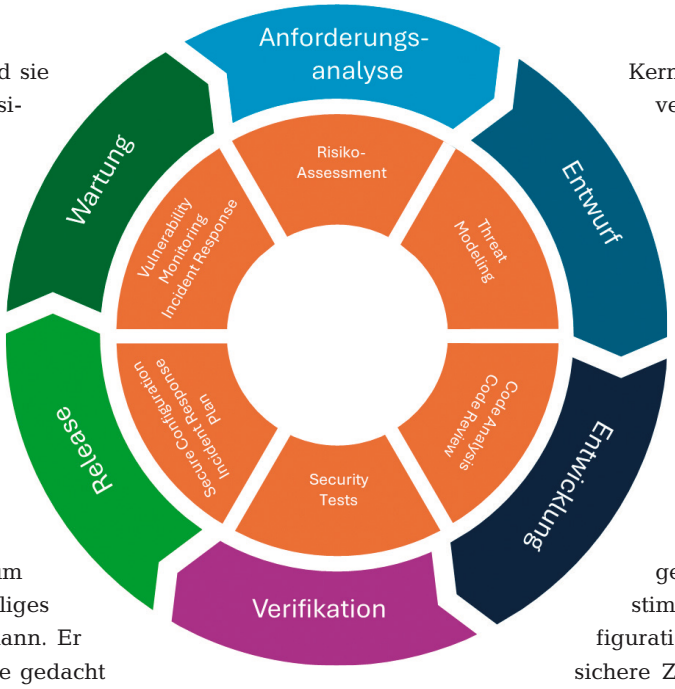
Besonders deutlich wird dieser Perspektivenwechsel beim Thema Update-Fähigkeit. Ob ein Produkt zuverlässig aktualisiert werden kann, ist keine organisatorische Frage, sondern eine technische. Es hängt davon ab, wie Software modularisiert ist, wie Konfigurationen gehandhabt werden und wie robust der Update-Mechanismus gegenüber Fehlern ist. Wenn Updates nur unter idealen Bedingungen funktionieren oder tief in die Funktionalität eingreifen, ist das kein Randproblem, sondern ein Sicherheitsrisiko. Der CRA macht diese Zusammenhänge sichtbar, weil er Update-Fähigkeit als zentrales Element der Produktsicherheit betrachtet.

Ähnlich verhält es sich mit der Transparenz über eingesetzte Komponenten. Ob eine Software Bill of Materials (SBOM) existiert und aktuell gehalten wird, ist keine Formalie. Sie setzt voraus, dass Abhängigkeiten bewusst verwaltet, versioniert und in den Entwicklungsprozess integriert sind. Das ist eine Aufgabe, die tief im technischen Alltag verankert ist. Entwickler entscheiden, welche Bibliotheken genutzt werden, wie sie eingebunden werden und wie Updates eingespielt werden. Diese Entscheidungen lassen sich nicht im Nachhinein durch Dokumentation korrigieren. Sie müssen von Anfang an mitgedacht werden. Das BSI liefert mit der Richtlinie TR-03182 Part 2 eine Anforderungsbeschreibung für SBOMs [4].

Der CRA zwingt Entwickler außerdem dazu, implizite Annahmen explizit zu machen. In zahlreichen Projekten gibt es stillschweigende Voraussetzungen darüber, wie ein Produkt genutzt wird, wer Zugriff hat oder in welcher Umgebung es betrieben wird. Solange diese Annahmen zutreffen, ▶

funktioniert das System. Sobald sie verletzt werden, entstehen Risiken. Der CRA verlangt, dass solche Annahmen bewusst betrachtet werden. Nicht um jedes Risiko auszuschließen, sondern um nachvollziehbar zu entscheiden, welche Risiken akzeptiert werden und welche nicht. Diese Entscheidungen entstehen nicht auf dem Papier, sondern im Zusammenspiel von Architektur, Implementierung und Nutzungskontext.

Damit wird auch klar, warum der CRA nicht durch ein einmaliges Projekt abgearbeitet werden kann. Er verändert die Art, wie Produkte gedacht und weiterentwickelt werden (vergleiche Bild 2). Entwicklern kommt dabei eine Schlüsselrolle zu, weil sie die technische Grundlage schaffen, auf der alle weiteren Maßnahmen aufbauen. Der CRA ist deshalb kein Compliance-Problem, das sich am Rand der Entwicklung lösen lässt. Er ist ein Thema, das den Kern professioneller Softwareentwicklung berührt. Er stellt die Frage, wie belastbar technische Entscheidungen sind, wenn sie über Jahre hinweg Bestand haben müssen. Für Entwickler ist das unbequem, weil es gewohnte Freiheitsgrade einschränkt. Gleichzeitig macht es sichtbar, wo gute Entwicklungspraxis bereits vorhanden ist und wo sie bisher nur implizit gelebt wurde. Genau an dieser Stelle beginnt der eigentliche Mehrwert des CRA für die technische Praxis.



Vom Development Lifecycle zum Secure Development Lifecycle durch zusätzliche Praktiken (Bild 2)

Was viele Entwickler bereits beherrschen: Der solide Kern moderner Softwareentwicklung

Wenn der Blick nun von der regulatorischen Einordnung auf die technische Praxis wechselt, ist es wichtig, zunächst festzuhalten, dass ein erheblicher Teil der CRA-Anforderungen für viele Entwickler nichts grundsätzlich Neues darstellt. Moderne Softwareentwicklung arbeitet seit Jahren mit Prinzipien, die sich nahezu deckungsgleich im CRA wiederfinden. Das ist kein Zufall. Der Act greift bewusst auf etablierte Konzepte zurück, weil sie sich als wirksam erwiesen haben. Für Entwickler bedeutet das: Ein Teil der Anforderungen bestätigt, was ohnehin als guter Standard gilt.

Dazu gehört in erster Linie das Prinzip Security by Design. Sicherheit wird nicht als Zusatz betrachtet, sondern als integraler Bestandteil von Architektur und Implementierung. Entwickler berücksichtigen Angriffsflächen bei der Gestaltung von Schnittstellen, minimieren Komplexität und vermeiden unnötige Funktionen. Entscheidungen über Protokolle, Authentifizierungsmechanismen oder Datenhaltung werden nicht isoliert, sondern im Kontext möglicher Risiken getroffen. Diese Denkweise ist in vielen Teams etabliert, zumindest dort, wo Software nicht als reines Hilfsmittel, sondern als Kernbestandteil eines Produkts verstanden wird.

Eng damit verbunden ist Security by Default. Systeme werden so ausgeliefert, dass sie im Ausgangszustand möglichst geringe Angriffsflächen bieten. Unsichere Voreinstellungen, offene Zugänge oder aktive Debug-Funktionen gelten heute als klare Mängel. Entwickler wissen, dass Sicherheit nicht davon abhängen darf, ob ein Nutzer bestimmte Optionen setzt oder Konfigurationen korrekt vornimmt. Der sichere Zustand muss der Normalzustand sein. Auch diese Anforderung findet sich im CRA wieder, ohne dass sie technisch neu definiert würde.

Ein weiteres Feld, das für viele Entwickler bereits zum Alltag gehört, ist die saubere Trennung von Rollen und Rechten. Authentifizierung und Autorisierung werden nicht mehr als Nebenthemen, sondern als zentrale Sicherheitsmechanismen behandelt. Unterschiedliche Nutzerrollen, begrenzte Privilegien und kontrollierte Zugriffe gehören zum Standardrepertoire moderner Systeme. Zwar zeigen sich in der Praxis immer noch Schwächen, etwa durch übermäßig privilegierte Service-Accounts oder unklare Wartungszugänge, doch das grundlegende Prinzip ist etabliert und verstanden.

Auch Secure Coding Practices im engeren Sinne sind für viele Entwickler selbstverständlich. Dazu zählen etwa die Validierung von Eingaben, der kontrollierte Umgang mit Fehlern und Ausnahmen, das Vermeiden unsicherer Konstrukte sowie der bewusste Einsatz von Bibliotheken und Frameworks. Der CRA verlangt hier keine neuen Methoden. Er setzt vielmehr voraus, dass diese Grundlagen beherrscht werden. Neu ist lediglich, dass Versäumnisse in diesem Bereich nicht mehr nur als Qualitätsprobleme wahrgenommen werden, sondern als potenzielle Sicherheitsmängel mit regulatorischer Relevanz.

Wichtig ist jedoch, diesen Abschnitt nicht als Entwarnung zu missverstehen. Dass viele Prinzipien bekannt sind, bedeutet nicht automatisch, dass sie durchgängig und konsequent umgesetzt werden. In vielen Projekten existieren sie als Leitlinien, werden aber unter Zeitdruck relativiert oder in Randbereichen aufgeweicht. Der CRA macht diese Unterschiede sichtbar. Er bewertet nicht, ob ein Team die richtigen Konzepte kennt, sondern ob das Produkt sie tatsächlich verkörpert. Zwischen Wissen und Umsetzung entsteht hier eine entscheidende Differenz.

Gerade für Entwickler kann dieser Punkt auch entlastend sein. Der CRA fordert keine radikale Umstellung der technischen Grundlagen. Er bestätigt vielmehr, dass etablierte gu-

te Praxis weiterhin trägt. Wer bereits strukturiert entwickelt, Sicherheit frühzeitig berücksichtigt und technische Schulden nicht dauerhaft ignoriert, bringt eine solide Basis mit. Gleichzeitig wird deutlich, dass diese Basis allein nicht ausreicht, um alle Anforderungen abzudecken. Die bekannten Prinzipien bilden den Ausgangspunkt, nicht das Ziel. Genau an dieser Stelle setzt die nächste Ebene an, auf der sich zeigt, wo bekannte Konzepte weitergedacht und konsequenter umgesetzt werden müssen.

Bekannt, aber nicht zu Ende gedacht: Updates, Abhängigkeiten und Verantwortung

Während die Grundlagen moderner sicherer Softwareentwicklung in vielen Teams etabliert sind, zeigt sich auf der nächsten Ebene ein deutliches Spannungsfeld zwischen Anspruch und gelebter Praxis. Die hier verorteten Themen sind den meisten Entwicklern vertraut, werden jedoch häufig nicht systematisch, nicht durchgängig oder nicht über den gesamten Lebenszyklus hinweg umgesetzt. Genau an dieser Stelle entfaltet der CRA seine größte Wirkung, weil er implizite Kompromisse sichtbar macht, die bislang selten hinterfragt wurden.

Ein zentrales Beispiel dafür ist das Patch- und Update-Management. Nahezu jedes Produkt verfügt heute über einen Mechanismus, um Software zu aktualisieren. In der Praxis sind diese Mechanismen jedoch oft fragil, historisch gewachsen oder stark von der Einsatzumgebung abhängig. Updates funktionieren unter idealen Bedingungen, scheitern jedoch bei instabilen Verbindungen, begrenzten Ressourcen oder ungewöhnlichen Konfigurationen. Häufig sind Sicherheitsfixes eng mit Feature-Releases verknüpft, weil der organisatorische oder technische Aufwand für getrennte Update-Pfade als zu hoch empfunden wird. Der CRA stellt diese Praxis infrage, weil er davon ausgeht, dass es möglich sein muss, Sicherheitsupdates unabhängig von funktionalen Weiterentwicklungen bereitzustellen.

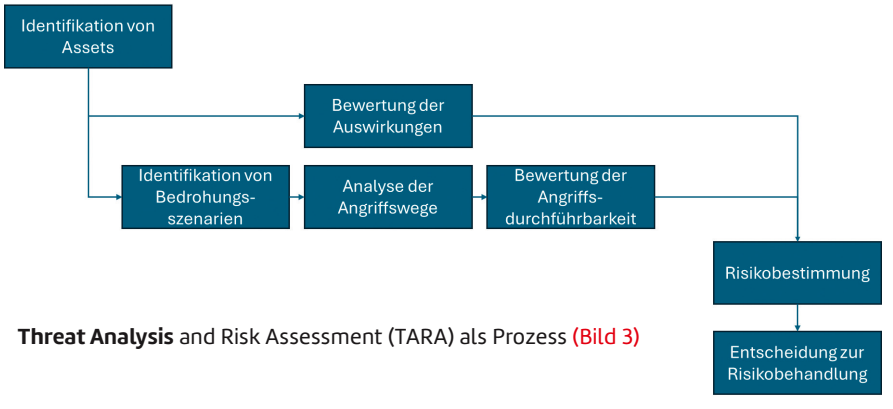
Für Entwickler bedeutet das, Update-Fähigkeit neu zu denken. Sie ist kein nachgelagertes Betriebsproblem, sondern eine Eigenschaft, die tief in Architektur und Implementierung verankert ist. Modularisierung, saubere Trennung von Konfiguration und Code, klare Versionsgrenzen und robuste Rollback-Mechanismen sind keine optionalen Optimierungen mehr. Sie bestimmen, ob ein Produkt langfristig sicher betrieben werden kann. Der CRA macht diese Zusammenhänge explizit, indem er Update-Fähigkeit als Voraussetzung für den Umgang mit Schwachstellen betrachtet.

Eng damit verknüpft ist die Frage, wie mit Abhängigkeiten umgegangen wird. Moderne Software besteht zu einem großen Teil aus Fremdkomponenten, insbesondere aus Open-Source-Bibliotheken. Entwickler sind sich dessen bewusst und nutzen etablierte Werkzeuge zur Verwaltung dieser Abhängigkeiten. Was jedoch häufig fehlt, ist eine konsistente Sicht über das gesamte Produkt hinweg. Abhängigkeiten werden pro Projekt, pro Modul oder sogar pro Entwickler betrachtet, nicht als Bestandteil eines Produkts mit klar definierter Verantwortung. Der CRA verschiebt hier den Fokus. Er verlangt, dass Hersteller jederzeit nachvollziehen können, welche Komponenten in welcher Version Teil eines Produkts sind und welche Risiken daraus resultieren.

Die Software Bill of Materials spielt in diesem Zusammenhang eine zentrale Rolle. Sie ist weniger ein neues Dokument als vielmehr das Ergebnis eines bewussten Umgangs mit Abhängigkeiten. Eine SBOM setzt voraus, dass Abhängigkeiten nicht nur technisch eingebunden, sondern auch versioniert, bewertet und gepflegt werden. In vielen Teams existieren die notwendigen Informationen zwar implizit, etwa in Build-Dateien oder Paketmanagern, sie sind jedoch nicht konsolidiert oder für eine systematische Risikobewertung nutzbar. Der CRA fordert genau diese Konsolidierung, weil ohne sie ein strukturiertes Schwachstellenmanagement kaum möglich ist.

Ein weiteres Spannungsfeld betrifft die Verantwortung für Third-Party-Komponenten. In der täglichen Entwicklung ist es üblich, Verantwortung entlang der Lieferkette weiterzugeben. Sicherheitslücken in externen Bibliotheken gelten als Problem des jeweiligen Maintainers. Der CRA akzeptiert diese Sichtweise nur eingeschränkt. Wer eine Komponente integriert, trägt Verantwortung für deren Einsatz im eigenen Produkt. Das bedeutet nicht, dass jede Schwachstelle sofort behoben werden muss. Es bedeutet jedoch, dass jede Schwachstelle bewertet werden muss. Entwickler und Hersteller müssen entscheiden, ob eine Lücke relevant ist, welche Auswirkungen sie haben könnte und welche Maßnahmen angemessen sind. Diese Entscheidungen dürfen nicht implizit bleiben, sondern müssen nachvollziehbar getroffen werden.

In der Praxis zeigt sich hier häufig ein Mangel an klaren Prozessen. Schwachstellenmeldungen werden reaktiv behandelt, wenn sie akut erscheinen, und ansonsten verdrängt. Der CRA verlangt mehr Systematik. Er zwingt Teams dazu, sich frühzeitig mit der Frage auseinanderzusetzen, wie Schwachstellen erkannt, priorisiert und behandelt werden. Diese Systematik lässt sich nicht allein durch Tools herstellen. Sie erfordert klare Verantwortlichkeiten und technische Voraussetzungen, die in der Entwicklung angelegt sein müssen. ▶



Threat Analysis and Risk Assessment (TARA) als Prozess (Bild 3)

Gerade in dieser zweiten Ebene wird deutlich, warum der CRA als unbequem wahrgenommen wird. Er greift nicht dort ein, wo offensichtliche Mängel bestehen, sondern dort, wo über Jahre hinweg pragmatische Kompromisse eingegangen wurden. Diese Kompromisse waren oft rational und nachvollziehbar, geraten nun jedoch unter einen neuen Maßstab. Der CRA bewertet nicht, ob eine Lösung praktikabel war, sondern ob sie langfristig tragfähig ist. Für viele Entwickler ist das eine ungewohnte Perspektive, weil sie den Blick von der kurzfristigen Projektlogik auf den gesamten Produktlebenszyklus lenkt.

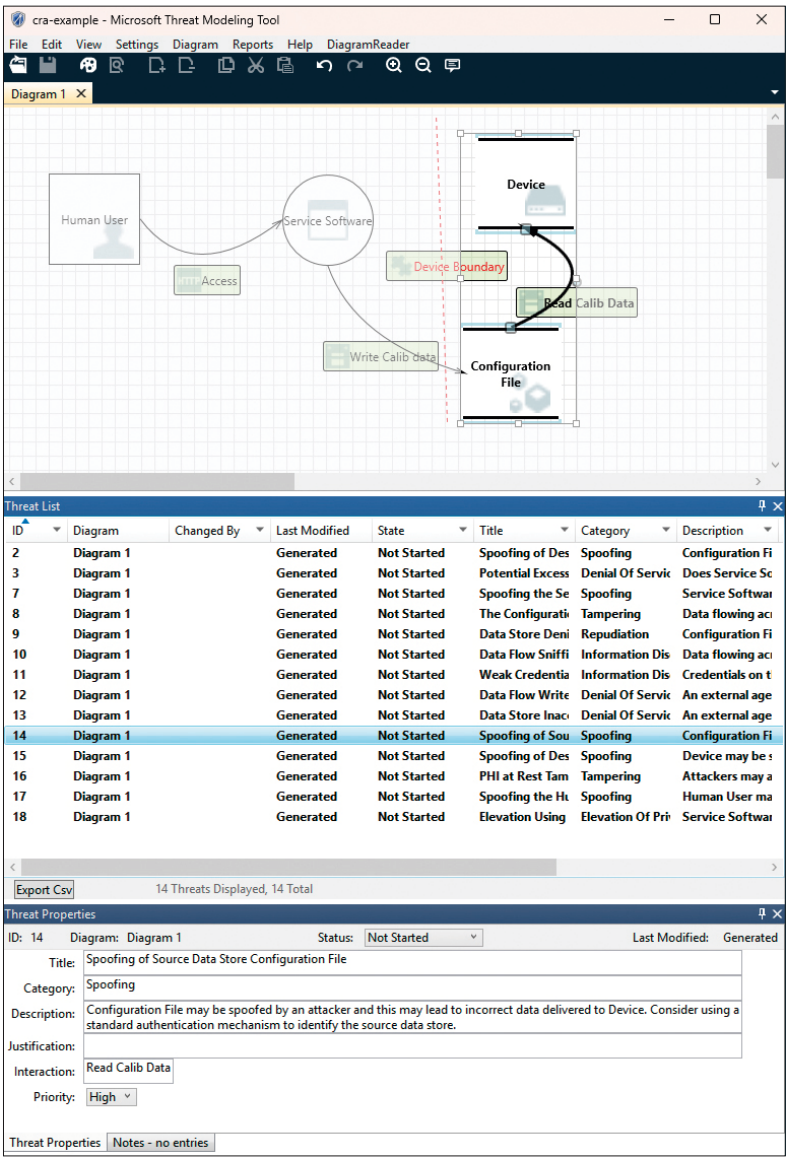
Genau dieser Perspektivenwechsel bereitet den Boden für die dritte Ebene, in der sich der eigentliche methodische Wandel vollzieht.

Neu für viele Teams: Risiken benennen, Angriffe denken, Entscheidungen begründen

Die dritte Ebene markiert den Punkt, an dem der CRA für viele Entwickler wirklich Neuland betritt. Hier geht es nicht mehr um technische Mechanismen oder bekannte Best Practices, sondern um eine veränderte Art, über Software nachzudenken. Der CRA verlangt, dass Annahmen, Abwägungen und Entscheidungen explizit gemacht werden. Für viele Teams ist das ungewohnt, weil diese Überlegungen bisher implizit Teil der Erfahrung einzelner Entwickler oder Architekten waren und selten formalisiert wurden.

Im Zentrum steht dabei die Risikoanalyse (vergleiche Bild 3). Entwickler treffen ständig Risikoentscheidungen, aber oft ohne sie so zu benennen. Sie entscheiden, welche Funktionen priorisiert werden, welche Komplexität akzeptabel ist und wo Vereinfachungen vertretbar erscheinen. Diese Entscheidungen basieren auf Erfahrung, Bauchgefühl und Kontextwissen. Der CRA fordert nicht, diese Intuition abzuschaffen, sondern sie nachvollziehbar zu machen. Eine Risikoanalyse zwingt dazu, die eigenen Annahmen offenzulegen. Welche Werte sollen geschützt werden? Welche Schäden wären realistisch, wenn diese Werte kompromittiert würden? In welchem Umfeld wird das Produkt voraussichtlich eingesetzt, und welche Nutzung oder auch Fehl- und Missbrauchsszenarien sind plausibel?

Für viele Entwickler ist neu, dass nicht nur der vorgesehene Gebrauch eines Produkts betrachtet werden soll, sondern auch dessen vorhersehbarer Missbrauch. Das bedeutet nicht, jedes denkbare Angriffsszenario zu berücksichtigen. Es bedeutet, typische Abweichungen von der idealen Nutzung mitzudenken. Produkte werden falsch konfiguriert, in ungeeigneten Netzwerken betrieben oder länger eingesetzt als ursprünglich geplant. Der CRA macht diese Realität zum Bestandteil der Bewertung. Entscheidungen, die solche Szenarien ignorieren, gelten nicht automatisch als falsch, müssen aber begründet werden.



Threat Modeling und Risikobewertung unter Nutzung des Microsoft Threat Modeling Tools (Bild 4)

Eng mit der Risikoanalyse verknüpft ist das Threat Modeling. Auch hier handelt es sich weniger um ein neues Werkzeug als um eine neue Perspektive. Threat Modeling zwingt Entwickler dazu, Funktionen aus der Sicht eines Angreifers zu betrachten. Welche Schnittstellen existieren? Welche Datenflüsse lassen sich missbrauchen? Wo entstehen Vertrauensgrenzen? Diese Fragen sind technisch, aber sie folgen einer anderen Logik als klassische Feature-Entwicklung. Viele Entwickler empfinden diesen Perspektivenwechsel als schwierig, weil er die gewohnte Denkweise durchbricht, in der Funktionen primär im Sinne des vorgesehenen Nutzers betrachtet werden.

In der Praxis führt Threat Modeling häufig zu unbequemen Erkenntnissen. Update-Mechanismen, die als rein technische Notwendigkeit betrachtet wurden, entpuppen sich als zentrale Angriffsflächen. Wartungsschnittstellen, die selten genutzt werden, besitzen oft hohe Privilegien und damit ein hohes Schadenspotenzial. Selbst Logging- und Monitoring-

Funktionen können Informationen preisgeben, die Angriffe erleichtern. Der CRA verlangt nicht, all diese Risiken vollständig zu eliminieren. Er verlangt, dass sie bewusst erkannt und bewertet werden. Genau diese bewusste Auseinandersetzung ist für viele Teams neu. Sie bedingt oft die Erweiterung der eigenen Werkzeugkette, zum Beispiel durch den Einsatz von Threat-Modeling-Tools [5] [6], vergleiche Bild 4.

Ein weiterer ungewohnter Aspekt ist die Forderung nach Begründbarkeit. Der CRA akzeptiert, dass Risiken existieren und dass nicht jedes Risiko auf technisch oder wirtschaftlich sinnvolle Weise mitigiert werden kann. Was er jedoch nicht akzeptiert, ist Unklarheit darüber, warum ein Risiko akzeptiert wurde. Entscheidungen müssen nachvollziehbar sein, sowohl intern als auch gegenüber Dritten. Das verändert die Rolle von Dokumentation. Sie wird nicht mehr als lästige Pflicht betrachtet, sondern als Spiegel der technischen Entscheidungsfindung. Gute Dokumentation erklärt nicht nur, wie es funktioniert, sondern auch, warum etwas so gebaut wurde, wie es gebaut wurde.

Für Entwickler bedeutet das auch eine Verschiebung der Verantwortung. Risikoanalyse und Threat Modeling lassen sich nicht vollständig an andere Rollen delegieren. Sie erfordern technisches Detailwissen und ein tiefes Verständnis der Architektur. Gleichzeitig sind sie keine rein technischen Übungen. Sie verbinden Technik mit Nutzungskontext, Geschäftsanforderungen und potenziellen Schadensszenarien. Genau diese Verbindung macht sie anspruchsvoll, aber auch wertvoll. Sie schafft ein gemeinsames Verständnis dafür, welche Risiken bewusst eingegangen werden und welche nicht.

In dieser dritten Ebene wird deutlich, warum der CRA mehr als eine Sammlung technischer Anforderungen ist. Er zwingt Entwickler dazu, ihre Entscheidungen nicht nur umzusetzen, sondern auch zu reflektieren und zu erklären. Das ist ungewohnt und erfordert Zeit und Übung. Gleichzeitig schafft es Klarheit. Implizite Annahmen werden sichtbar, blinde Flecken werden erkennbar und Entscheidungen werden belastbarer. Damit bildet diese Ebene das Fundament für den langfristigen Umgang mit Sicherheit über den gesamten Produktlebenszyklus hinweg.

Der CRA als Spiegel professioneller Softwareentwicklung

Am Ende lässt sich der Cyber Resilience Act nicht sinnvoll als rein technisches Regelwerk oder als juristische Pflichtübung einordnen. Er ist vielmehr ein Spiegel, der zeigt, wie professionell Softwareentwicklung in einem Unternehmen tatsächlich betrieben wird. Nicht im Sinne einzelner Codezeilen oder Tools, sondern im Zusammenspiel aus Architektur, Prozessen, Entscheidungsfindung und langfristiger Verantwortung. Genau deshalb wirkt der CRA auf viele zunächst unbequem. Er konfrontiert Organisationen nicht mit völlig neuen Anforderungen, sondern mit der Frage, wie konsequent bekannte Prinzipien wirklich gelebt werden.

Der Act verlangt keine neuen Programmiersprachen, keine spezifischen Frameworks und keine festgelegten Sicherheitslösungen. Er schreibt nicht vor, wie ein Produkt intern aufgebaut sein muss. Stattdessen fordert er, dass Produkte nachvollziehbar sicher sind, dass Risiken bewusst behandelt werden und dass Sicherheit über den gesamten Lebenszyklus hinweg berücksichtigt wird. Diese Offenheit ist zugleich Stärke und Herausforderung. Sie lässt Spielräume, zwingt aber dazu, Entscheidungen zu begründen und Verantwortung zu übernehmen.

Gleichzeitig bietet der CRA auch eine Chance. Er liefert ein starkes externes Argument für Themen, die in vielen Teams seit Jahren bekannt sind, aber immer wieder hintangestellt werden. Sie lassen sich nun nicht mehr als Nice-to-have abtun. Sie werden zu Voraussetzungen für Marktfähigkeit. Für Entwickler kann das entlastend sein, weil es lange bekannte technische Notwendigkeiten legitimiert und priorisierbar macht.

Der CRA zwingt nicht zur Perfektion, sondern zur Ehrlichkeit. Er akzeptiert, dass Risiken existieren und dass nicht jede Schwachstelle sofort beseitigt werden kann. Was er nicht akzeptiert, ist Unklarheit darüber, warum Entscheidungen getroffen wurden und welche Konsequenzen sie haben. In diesem Sinne ist der Act weniger ein Kontrollinstrument als ein Katalysator für reifere Entwicklungspraktiken.

Wer den CRA ausschließlich als Bedrohung wahrnimmt, verkennt diesen Kern. Er ist kein Innovationskiller und kein Selbstzweck. Er ist ein Rahmen, der sichtbar macht, ob Softwareentwicklung bereit ist, die Verantwortung zu tragen, die mit der zunehmenden Bedeutung digitaler Produkte einhergeht. Für Entwickler ist das unbequem, aber auch konsequent. Denn letztlich fordert der CRA nichts anderes ein als das, was professionelle Softwareentwicklung seit Langem verspricht: Systeme zu bauen, die nicht nur funktionieren, sondern auch langfristig verantwortbar sind. ■

[1] Cyber Resilience Act, https://digital-strategy.ec.europa.eu/en/policies/cyber-resilience-act

[2] EUR-Lex, Document 32024R2847, https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32024R2847

[3] BSI, TR-03183 Cyber-Resilienz-Anforderungen, www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr03183/TR-03183_node.html

[4] BSI, TR-03183, Part 2: Software Bill of Materials (SBOM), www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03183/BSI-TR-03183-2_v2_1_0.pdf?__blob=publicationFile&v=5

[5] Microsoft, Threat Modeling, www.microsoft.com/en-us/securityengineering/sdl/threatmodeling

[6] OWASP Threat Dragon, www.threatdragon.com

Lars Roith
ist Solution Consultant, erfahrener Digitalisierungs- und Softwareexperte und Gründer von Lunaris Digital Solutions. Er verfügt über mehr als 25 Jahre Erfahrung in der Softwareentwicklung, IT-Beratung und Unternehmensführung.
lars.roith@lunaris.digital