ONEKEY.COM



LEITFADEN ZUR ABSICHERUNG IHRER SOFTWARE-LIEFERKETTE

Bevor es zu spät ist

ONEKEY

WHITEPAPER

ZUSAMMENFASSUNG

Leitfaden zur Absicherung Ihrer Software-Lieferkette

Lesezeit: ca. 15 min.

Dieses Whitepaper richtet sich an Führungskräfte der obersten Ebene (CIOs, CTOs, CISOs), Risikomanager in der Lieferkette sowie an Sicherheits- und Compliance-Teams. Ihre Aufgabe ist es, ihre Unternehmen vor verborgenen Risiken in der Software-Lieferkette zu schützen – einschließlich neu auftretender KI-gesteuerter Gefahren. Darüber hinaus bietet es Firmware-Ingenieuren und technischen Fachkräften praxisnahe Anleitungen zu Themen wie Binärcode-Transparenz, SBOM-Validierung und Schwachstellenmanagement. Es ist besonders relevant für Branchen, in denen Firmware und eingebettete Systeme eine zentrale Rolle spielen, wie Telekommunikation, Automobilindustrie, Medizintechnik, Fertigung und kritische Infrastrukturen. [1]

Dieses Whitepaper zeigt, dass in der modernen Entwicklung zentrale Sicherheitsprüfungen häufig übersprungen oder übersehen werden. Enge Zeitpläne, intransparente Firmware und fragmentierte Tools erschweren sowohl die Validierung vor der Veröffentlichung als auch die Überwachung nach der Bereitstellung.

Gleichzeitig formulieren Vorschriften wie der EU Cyber Resilience Act, die US Executive Order 14028 oder NIS2 klare Anforderungen an die Sicherheit, Rückverfolgbarkeit und Verantwortlichkeit digitaler Produkte.

Die Sicherheit der Software-Lieferkette ist inzwischen etwas für die Vorstandsebene. Da immer mehr eingebettete Systeme auf Drittkomponenten, Firmware und interne Entwicklung angewiesen sind, steigt die Wahrscheinlichkeit verborgener Schwachstellen und Compliance-Probleme kontinuierlich.

Die heutigen Lieferketten sind komplex und oft intransparent, sodass gravierende Risiken leichter unbemerkt bleiben. Laut Gartner werden bis 2025 45 % der Unternehmen einen Angriff auf ihre Software-Lieferkette erlebt haben – eine Verdreifachung gegenüber 2021. [1]

Als Reaktion darauf sind Unternehmen gezwungen, „Shift Left“ zu praktizieren: Sicherheit muss früh in den Entwicklungsprozess eingebunden werden, nicht erst nach der Auslieferung von Produkten oder Firmware.

1 https://www.gartner.com/en/newsroom/press-releases/2022-03-07-gartner-identifies-top-security-and-risk-management-trends-for-2022

Hier macht Automatisierung den Unterschied. Durch den Einsatz automatisierter Tools im Firmware-Entwicklungsprozess lassen sich Binärdateien auch ohne Source Code analysieren, Schwachstellen frühzeitig erkennen, Richtlinien durchsetzen, eingesetzte Geräte überwachen und Compliance sicherstellen. Automatisierung verwandelt Best Practices in wiederholbare Abläufe und reduziert so operative Risiken und Kosten.

Zur Veranschaulichung untersucht dieses Whitepaper, wie Swisscom die automatisierte Lösung von ONEKEY eingeführt hat und dadurch pro vermiedenem Vorfall 400.000 USD einsparen konnte.

Wenn Ihnen die Absicherung Ihrer Lieferkette unerreichbar erscheint, zeigt dieses Whitepaper, wie Sie sie zu einem festen Bestandteil Ihrer Entwicklungs- und Führungsstrategie machen können

EINFÜHRUNG IN DIE SOFTWARE LIEFERKETTENSICHERHEIT

Die Lieferkette beschreibt den Weg eines Produkts von der Entwicklung bis zur Bereitstellung. In der digitalen Welt sprechen wir von der „Software-Lieferkette“, deren Komplexität unbestritten ist. Sie umfasst eine Reihe von Schritten, Tools und Komponenten, die Source Code in Software oder Firmware verwandeln, welche schließlich auf intelligenten Geräten und eingebetteten Systemen ausgeführt wird.

So wie ein Auto aus Tausenden von Teilen besteht, die von verschiedenen Zulieferern stammen, wird Software heute selten vollständig neu entwickelt. Sie setzt sich aus unterschiedlichen Bausteinen zusammen:

- Von internen Teams geschriebene Codes
- Open-Source-Software Libraries und Frameworks
- Kommerzielle Komponenten von Drittanbietern
- KI-generierter Code
- Von Lieferanten bereitgestellte Binärdateien und Firmware-Module

Diese Komponenten werden integriert, transformiert und schließlich in die Software oder Firmware eingebunden, die auf modernen Geräten zum Einsatz kommt.

Greifen wir noch einmal das Beispiel der Automobilindustrie auf: Strenge Quality Gates, Compliance-Prüfungen und Rückverfolgbarkeit sind entlang der gesamten Lieferkette unverzichtbar, um sicherzustellen, dass jede einzelne Komponente die geforderten Sicherheits- und Leistungsstandards erfüllt.

In der digitalen Welt spricht man in diesem Zusammenhang von „Software Supply Chain Security". Darunter versteht man die Prozesse, Praktiken und Technologien, die gewährleisten, dass sämtlicher Code, alle Komponenten und Libraries, die in Entwicklung, Produktion und Verteilung von Software oder Firmware einfließen, vertrauenswürdig, sicher und konform sind.

Da sich dieses Whitepaper auf die Lieferkettensicherheit in der Embedded World konzentriert – also auf Geräte, die für spezifische Funktionen mit besonderen Betriebsanforderungen entwickelt werden – liegt der Fokus auf der Firmware, die diese Funktionen ermöglicht.

In manchen Abschnitten wird jedoch bewusst der allgemeinere Begriff „Software" verwendet, um die Überschneidung der Konzepte zu verdeutlichen.



Schlüssel Rollen in der Software-Lieferkette

Um eine wirksame Sicherheitsstrategie zu entwickeln, ist es entscheidend zu verstehen, welche Akteure in der Software-Lieferkette welche Aufgaben übernehmen. In ihr finden sich verschiedene Rollen wie Asset-Eigentümer, Wartungsdienstleister, Integrationspartner und Produktlieferanten.

Der Einfachheit halber unterscheidet dieses Whitepaper zwischen zwei Kategorien: Anbieter (Produktlieferanten) und Betreiber (Asset-Eigentümer, Wartungs- und Integrationsdienstleister).

Anbieter/Entwickler:

Sie entwickeln Firmware, integrieren Komponenten, unterstützen Produkte und liefern sie an Kunden, Betreiber oder Partner aus. Ihre Verantwortung: die Integrität, Sicherheit und Compliance der entwickelten Produkte sicherzustellen und diese zuverlässig bereitzustellen.

Betreiber

Sie verwalten und betreiben die eingesetzten Systeme. Dazu gehören die Inbetriebnahme, Wartung und Validierung der eingesetzten Lösungen.

Ihre Verantwortung: die Sicherheit der Firmware in ihren Umgebungen kontinuierlich zu überwachen und zu managen.

Die Software-Lieferkette vom Anbieter bis zum Betreiber

Im Idealfall umfasst die Software-Lieferkette fünf Phasen von der Entwicklung bis zur Bereitstellung. Sie gilt dann als sicher, vertrauenswürdig und überprüfbar, wenn in allen Phasen Sicherheitsprüfungen, Compliance-Vorgaben und kontinuierliche Überwachung integriert sind. Das folgende Diagramm veranschaulicht die ideale Software-Lieferkette zusammen mit kritischen Sicherheitskomponenten.

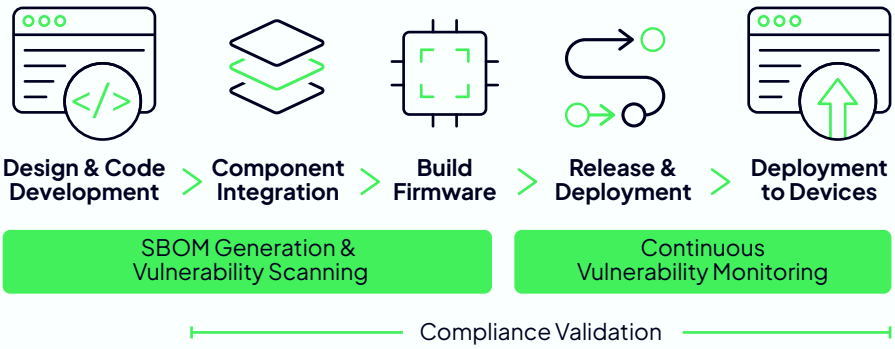


Abbildung 1: Eine ideale Software-Lieferkette integriert Sicherheitsvalidierungen während der Entwicklung und Erstellung, sorgt für die Einhaltung von Vorschriften vor der Veröffentlichung und gewährleistet während des Betriebs kontinuierliche Transparenz und Kontrolle.

Die fünf Stufen der idealen Software-Lieferkette sowie die Sicherheitsmaßnahmen umfassen:

1. Design und Code Entwicklung

Bevor auch nur eine Zeile Code entsteht, beginnt der Prozess mit einem Secure-by-Design-Ansatz. Schon in der Entwurfsphase werden Sicherheits- und Compliance-Anforderungen berücksichtigt. So wird sichergestellt, dass Architekturentscheidungen, Bedrohungsmodelle und die Auswahl der Komponenten mit der individuellen Risikolage und den regulatorischen Vorgaben des Unternehmens im Einklang stehen.

Erst danach starten die Entwickler mit dem Schreiben proprietären Codes und der Umsetzung von Funktionen auf Basis von User Stories und funktionalen Anforderungen. Dabei werden auch die erforderlichen Abhängigkeiten eingebunden.

Best Practice für die Sicherheit: Von Beginn an sollten Organisationen klare Sicherheits- und Compliance-Richtlinien für alle internen und externen Komponenten definieren, basierend auf den in der Entwurfsphase festgelegten Grundsätzen.

2. Komponentenintegration

Entwickler integrieren Open-Source-Pakete, Libraries von Drittanbietern und vom Lieferanten bereitgestellte Binärdateien, da moderne Firmware nur selten vollständig neu entwickelt wird.

Auf diese Weise lassen sich gängige Funktionen implementieren, ohne dass alles intern programmiert werden muss.

Best Practice für die Sicherheit: Teams sollten eine Software-Stückliste (SBOM) erstellen und Schwachstellenscans durchführen, um sicherzustellen, dass alle Komponenten bekannt, überprüft und freigegeben sind. Zudem sollte regelmäßig kontrolliert werden, ob die Komponenten aktiv gepflegt werden, sodass neu entdeckte Schwachstellen zeitnah behoben werden können.

3. Firmware erstellen

In dieser Phase werden der Code, die integrierten Komponenten und deren Abhängigkeiten kompiliert und zu einer Firmware gepackt.

Best Practice für die Sicherheit: Vor jeder Veröffentlichung sollten Sicherheits- und Compliance-Prüfungen (Quality Gates) durchgeführt werden, um sicherzustellen, dass alle Komponenten die definierten Kriterien erfüllen und wie vorgesehen zusammenarbeiten.

4. Freigabe und Bereitstellung

Nach der Genehmigung wird die Firmware entweder in Staging-, Test- oder Produktionsumgebungen freigegeben. An diesem Punkt wird die Bereitstellungsbereitschaft überprüft.

Best Practice für die Sicherheit: Firmware sollte stets zusammen mit einer Compliance-Dokumentation und einer SBOM ausgeliefert werden, damit nachgelagerte Teams den Inhalt nachvollziehen und überprüfen können.

5. Bereitstellung auf Geräten

Betreiber installieren die Firmware in unterschiedlichen Umgebungen – von Cloud- und Unternehmenssystemen bis hin zu eingebetteten Geräten und industriellen Endgeräten.

Best Practice für die Sicherheit: Diese Phase muss eine kontinuierliche Überwachung, die Verfolgung von Schwachstellen im Zeitverlauf, die Erkennung von Regressionen und die Verwaltung des Patch-/Update-Zyklus umfassen.



Ebenen der Firmware-Komplexität

In einem eingebetteten System liegt die Firmware direkt auf der Hardware und erstreckt sich über mehrere klar unterscheidbare Ebenen. Die Anwendungsebene ist der Bereich, in dem die Komplexität der Software-Lieferkette besonders deutlich wird. Hier wird die Hauptfunktionalität des Produkts implementiert – häufig unter Verwendung von intern entwickeltem Code, Libraries von Drittanbietern und von Lieferanten bereitgestellten Komponenten, die jeweils eigene Abhängigkeiten mitbringen. Jedes an dieser Stelle eingeführte Element birgt potenzielle Risiken und Abhängigkeitsketten, die sich auf das gesamte System auswirken können.

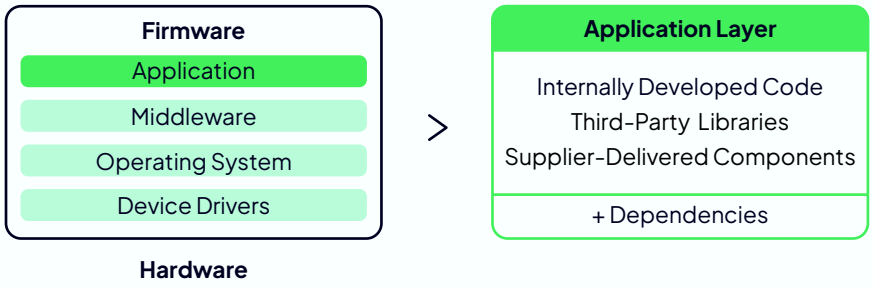


Abbildung 2: Komplexitätsebenen der Firmware in eingebetteten Systemen

Die anderen Ebenen lassen sich wie folgt zusammenfassen:

Middleware: Die Middleware bildet die Brücke zwischen der Anwendungsebene und den darunterliegenden Ebenen. Sie sorgt dafür, dass die verschiedenen Teile des eingebetteten Systems reibungslos zusammenarbeiten. Typische Bestandteile sind Gerätetreiber, Kommunikationsprotokolle und Sicherheitsdienste. Betriebssystem: Es verwaltet Systemressourcen wie Speicher, CPU und Peripheriegeräte und gewährleistet so einen effizienten Betrieb. Gerätetreiberebene: Diese Ebene steuert direkt Hardwarekomponenten mit Gerätetreibern wie Sensortreibern, Aktortreibern und Peripherietreibern. Hardwareebene: Sie ist die Grundlage des Systems und umfasst CPU, Speicher und Peripheriegeräte wie Sensoren, Aktoren, Kommunikationsschnittstellen, Timer usw.

Die Software-Lieferkette entwickelt sich rasant weiter, um den Anforderungen einer schnelleren Entwicklung und Bereitstellung gerecht zu werden. Gleichzeitig wird ihre Struktur immer vielschichtiger und komplexer und umfasst zahlreiche Tools, Komponenten und Plattformen. Diese Komplexität bringt eine wachsende Herausforderung mit sich, zu verstehen und zu kontrollieren, was tatsächlich zusammengestellt und bereitgestellt wird. Infolgedessen nehmen die Risiken und Schwachstellen in der gesamten Lieferkette stetig zu.

REALITÄT DER SOFTWARE-LIEFERKETTE UND VERBUNDENE RISIKEN

Im Idealfall umfasst die Software-Lieferkette eine strenge Validierung vor der Freigabe sowie eine kontinuierliche Überwachung nach der Bereitstellung. In der Praxis jedoch tun sich viele Unternehmen schwer, diese Schritte konsequent umzusetzen.

Realität der Software-Lieferkette

Viele Software-Lieferketten überspringen Sicherheits- und Compliance-Prüfungen oder führen sie nur unvollständig durch. Faktoren wie schnelle Release-Zyklen, fehlende Transparenz bei Lieferanten und fragmentierte Tools führen häufig zu unvollständigen SBOMs, ungeprüften Drittanbieterkomponenten und oberflächlichen Schwachstellenscans. Eine Überwachung nach der Bereitstellung fehlt oft – insbesondere bei eingebetteten Systemen.

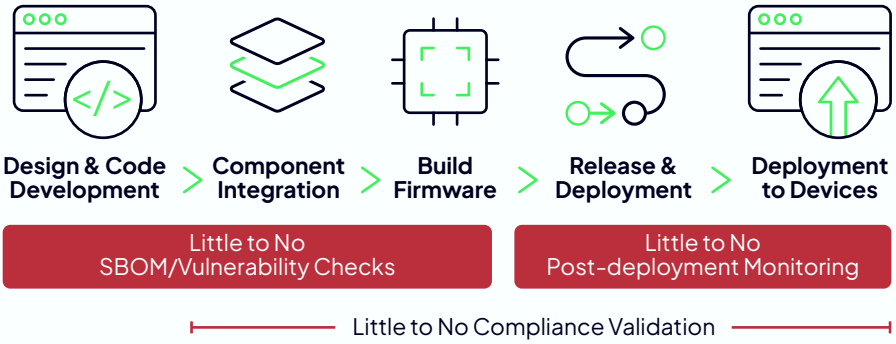
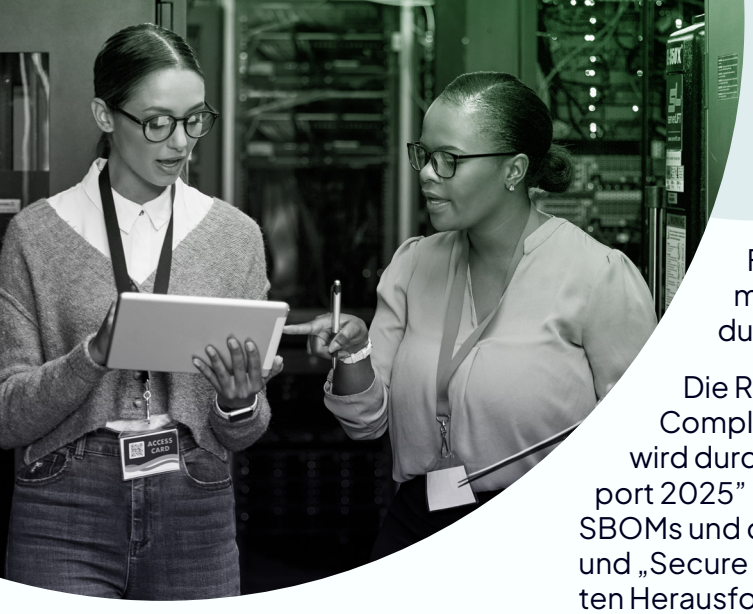


Abbildung 3: In vielen Unternehmen hat die Software-Lieferkette heute zwar die Liefergeschwindigkeit im Fokus, es mangelt jedoch an einer konsistenten Sicherheitsüberprüfung und Überwachung nach der Bereitstellung, sodass Risiken unbemerkt weitergegeben werden können.

Realität der Sicherheits- und Compliance-Validierung (Vorabversion)

- Viele Unternehmen führen nur eine teilweise oder unvollständige Validierung durch.
- Einige Anbieter verzichten vollständig auf eine Validierung, insbesondere wenn sie unter Zeitdruck stehen oder ihre Systeme nicht ausreichend in Sicherheitstools integriert sind.
- SBOMs sind oft unvollständig oder fehlen ganz.
- Schwachstellenscans werden möglicherweise mit herkömmlichen Scannern durchgeführt, möglicherweise aber auch nicht. Wenn sie durchgeführt werden, wissen die Unternehmen nicht, welche Schwachstellen relevant sind.



- Binärdateien von Drittanbietern und Lieferanten werden oft als Black Boxes behandelt; es wird keinerlei Validierung durchgeführt.

Folgen: Schwachstellen, Malware oder nicht konforme Komponenten können ausgeliefert werden, wodurch Lücken in der Lieferkettensicherheit entstehen.

Die Realität fehlender oder unvollständiger SBOMs und Compliance-Validierungen (insbesondere für den CRA) wird durch unseren aktuellen „IoT & OT Cybersecurity Report 2025" untermauert. Darin heißt es, dass die Erstellung von SBOMs und die Einhaltung der Grundsätze „Secure by Design" und „Secure by Default" für viele Unternehmen zu den drei größten Herausforderungen zählen. Link: https://www.onekey.com/resource/iot-ot-cybersecurity-report-2025

Realität der Betriebsüberwachung (nach der Bereitstellung)

- Viele Unternehmen verfügen über keine ausgereifte Überwachung der eingesetzten Firmware
- Manchmal gibt es keinen Prozess für das erneute Scannen von SBOMs nach der ersten Bereitstellung

Folgen: Schwachstellen in der eingesetzten Firmware bleiben oft über längere Zeiträume unbehandelt, wodurch Angreifer das große Zeitfenster gezielt ausnutzen können.

Auswirkungen in der realen Welt: SolarWinds Angriff auf die Lieferkette [2]

Der SolarWinds-Angriff ist eines der bekanntesten Beispiele für eine Sicherheitsverletzung in der Software-Lieferkette. Dabei kompromittierten Angreifer die Build-Umgebung des Unternehmens und schleusten eine bösartige Hintertür in ein routinemäßiges Software-Update ein. Dieses kompromittierte Update wurde dann unwissentlich an über 18.000 Kunden verteilt, darunter große Unternehmen und US-Behörden. Der Angriff machte deutlich, wie ein einziger Fehlerpunkt in der Software-Lieferkette massive Auswirkungen nachgelagert haben kann und selbst die sichersten Umgebungen erreichen kann.

Fazit: Ohne tiefgehende Einblicke in die endgültige Firmware-Binärdatei kann bösartiger Code unbemerkt über eigentlich vertrauenswürdige Kanäle eingeschleust und in großem Umfang verbreitet werden.

Auswirkungen in der realen Welt: Entdeckung einer XZ Backdoor enthüllt Angriff auf die Linux Lieferkette [3]

Anfang 2024 wurde eine kritische Hintertür in der weit verbreiteten Open-Source-Komprimierungsbibliothek XZ Utils entdeckt, die in zahlreichen Linux-Distributionen eingesetzt wird. Der Angreifer hatte sich über Jahre hinweg das Vertrauen der Maintainer erarbeitet und schließlich bösartigen Code in die Updates des Projekts eingeschleust. Nach der Bereitstellung ermöglichte diese Hintertür den Fernzugriff auf betroffene Systeme.

2 https://www.techtarget.com/whatis/feature/SolarWinds-hack-explained-Everything-you-need-to-know
3 https://www.techtarget.com/searchsecurity/news/366577602/XZ-backdoor-discovery-reveals-Linux-supply-chain-attack

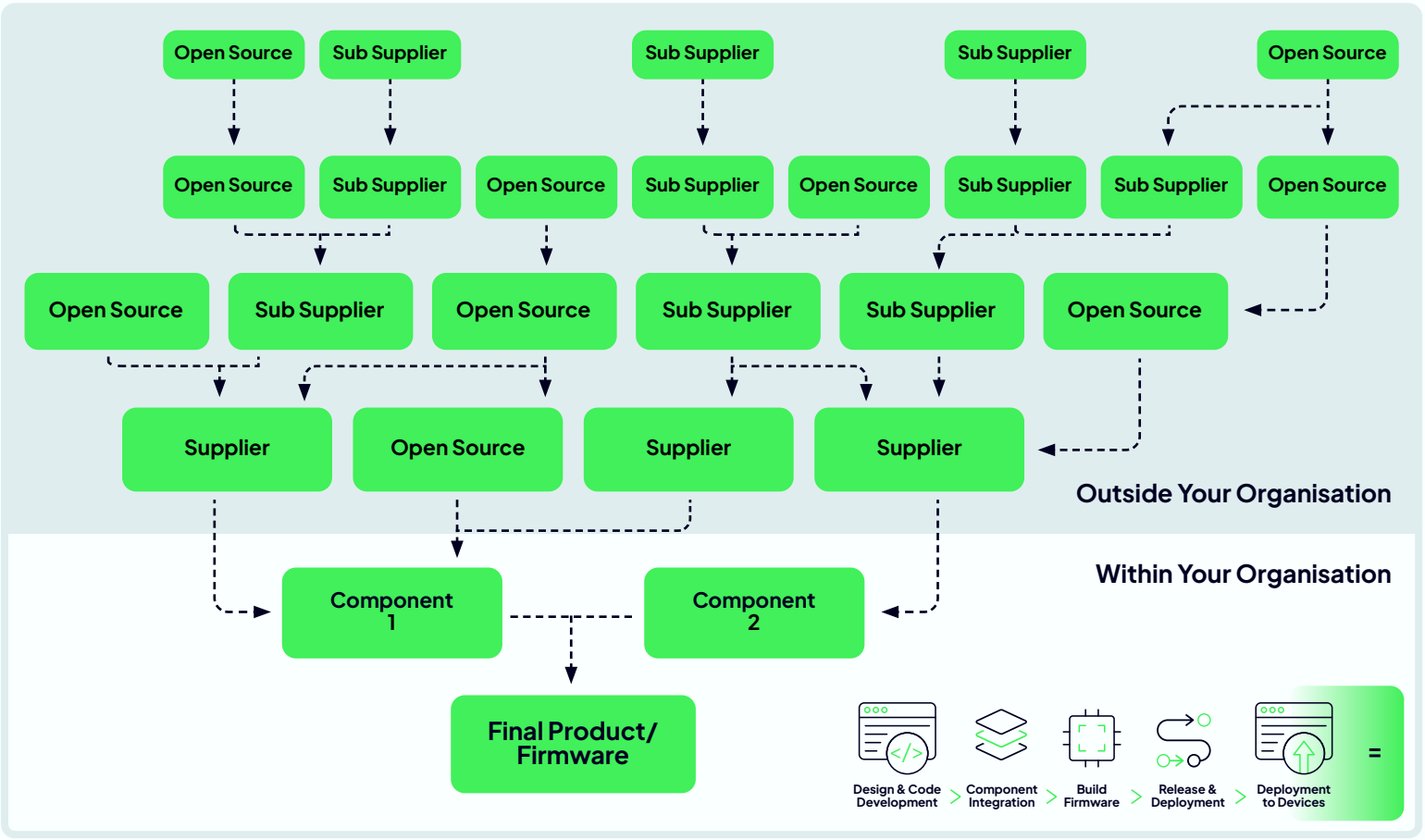


Abbildung 4: Die Software-Lieferkette ist aufgrund der Vielzahl von Abhängigkeiten ein komplexes Ökosystem und kann als „Embedded Software Supply Chain" bezeichnet werden

Der Vorfall machte deutlich, dass selbst zentrale und seit Langem vertrauenswürdige Open-Source-Komponenten an der Quelle kompromittiert werden können – mit Auswirkungen auf unzählige nachgelagerte Produkte und Dienste.

Takeaway: Selbst weit verbreitete und „vertrauenswürdige" Komponenten können bereits im Upstream-Prozess zur Waffe werden. Eine Überprüfung auf Binärcode-Ebene ist daher unerlässlich.

Komplexes Geflecht der Software-Lieferkette

Die Software-Lieferkette ist selten linear. Wie in Abbildung 4 dargestellt, funktioniert sie vielmehr als komplexes Netzwerk aus miteinander verbundenen Akteuren: dem Endprodukt bzw. der Firmware, Lieferanten, Unterlieferanten, internen Teams und Open-Source-Projekten. Jede dieser Parteien trägt in unterschiedlichen Phasen Komponenten, Code oder Prozesse bei. Bevor die Firmware ihr Ziel erreicht, durchläuft sie in der Regel mehrere Stufen – von der Codeentwicklung über die Integration von Komponenten und die Erstellung der Firmware bis hin zur Bereitstellung für Tests. Diese Schritte erfolgen oft verteilt über verschiedene Systeme, Teams und sogar Organisationen hinweg. Dieses vielschichtige Netzwerk von Abhängigkeiten macht die Lieferkette zwar hochgradig skalierbar, verdeutlicht aber auch, wie komplex und verteilt der Weg bis zur Bereitstellung geworden ist.

Damit verbundene Sicherheitsrisiken

Die Software-Lieferkette ist nicht nur lang, sondern auch komplex. Da in den verschiedenen Phasen häufig Sicherheits- und Compliance-Kontrollpunkte fehlen oder nicht verfügbar sind, bleibt sie permanent gefährdet. Mit jeder Phase sind spezifische Risiken verbunden, darunter:

1. Design und Code-Entwicklung

Das Fehlen von Security-by-Design-Prinzipien führt zu architektonischen Abkürzungen, fehlenden Bedrohungsmodellen oder ignorierten Sicherheitsmustern. Dadurch entstehen nicht nur strukturelle Schwächen, sondern auch die Gefahr, dass unsicherer Code und ungeprüfte Libraries frühzeitig integriert werden. Diese Probleme sind in späteren Phasen deutlich schwerer und kostspieliger zu beheben.

2. Komponentenintegration

Libraries von Drittanbietern und Binärdateien von Lieferanten werden oft ohne gründliche Prüfung eingebunden. Diese Komponenten können transitive Schwachstellen in verschachtelten Abhängigkeiten enthalten, die von außen nicht erkennbar sind.

3. Firmware erstellen

Ohne geeignete Kontrollen können während der Paketierung Injektionen, Konfigurationsfehler oder Image-Schwachstellen entstehen. Unvollständige SBOMs und schwache Validierungen führen zudem dazu, dass Teams nur vermuten können, was tatsächlich in der endgültigen Firmware enthalten ist.

4. Freigabe und Bereitstellung

Releases werden häufig ausgeliefert, ohne dass ihr Inhalt gründlich überprüft wurde. Dabei können Fehlkonfigurationen und unsichere Einstellungen ebenso unentdeckt bleiben wie nicht geprüfte Build-Ergebnisse oder zuvor nicht erkannte Compliance-Lücken.

5. Bereitstellung auf Geräten

Nach der Bereitstellung der Firmware werden nur selten regelmäßige Schwachstellenscans durchgeführt. Ohne ausreichende Transparenz bleiben bekannte Schwachstellen ungepatcht, und Zero-Day-Lücken können unentdeckt bestehen. Das insbesondere der Fall, wenn es keinen Prozess gibt, der nachverfolgt, was tatsächlich im Feld ausgeführt wird.

Neue Risiken für die Lieferkette durch KI-Coding

KI-Coding-Assistenten wie ChatGPT, GitHub Copilot oder CodeLlama werden von Entwicklern zunehmend in der Firmware-Entwicklung eingesetzt, um die Produktivität zu steigern. Diese Werkzeuge bergen jedoch neue Risiken für die Software-Lieferkette, da sie fehlerhafte oder gar nicht existierende Libraries und Komponenten vorschlagen können.

1. Slopsquatting (Ausnutzung von KI-Fehlern)

KI-Coding-Assistenten werden auf Basis umfangreicher Datensätze trainiert, darunter öffentliche Code-Repositories, Dokumentationen und andere frei verfügbare Quellen aus dem Internet. Modelle wie ChatGPT, GitHub Copilot oder CodeLlama verstehen Software jedoch nicht wie Menschen, da sie sich noch in einer frühen Phase ihres Lebenszyklus befinden.

Sie sagen lediglich den wahrscheinlich nächsten Codeabschnitt voraus – basierend auf Mustern, die sie im Training gesehen haben. Dabei können sie Paketnamen „halluzinieren", die zwar plausibel klingen, tatsächlich aber nicht existieren oder falsch eingegebene Abhängigkeiten darstellen.

Cyberkriminelle und andere böswillige Akteure beobachten diese populären LLM-Tools genau und analysieren solche halluzinierten oder wahrscheinlich vorgeschlagenen Namen. Anschließend registrieren sie die gefälschten Pakete vorab – ein Vorgehen, das als Slopsquatting bekannt ist. Sobald das Paket im Repository verfügbar ist, wird jeder Entwickler, der blind vertraut und den von der KI vorgeschlagenen Code installiert, angreifbar.

2. Vibe Coding befeuert Slopsquatting Angriffe



Kreativität über strikte Validierung stellen. In diesem Arbeitsablauf wird Code häufig ohne Peer-Reviews geschrieben, Abhängigkeiten werden nach Bauchgefühl eingebunden und KI-Tools werden unkritisch genutzt, um Lücken zu füllen.

Die Folgen: Vibe Coding schafft ideale Voraussetzungen für Slopsquatting. Wenn Entwickler zugunsten der Geschwindigkeit auf Validierung verzichten, verlassen sie sich unbewusst auf KI-generierte Halluzinationen. Angreifer können diese Vorhersagbarkeit ausnutzen und in großem Umfang bösartige Pakete einschleusen. Selbst Projekte in einem frühen Stadium mit vermeintlich geringem Risiko können so zu potenziellen Einstiegspunkten in die Software-Lieferkette werden.

Da diese Risiken in Umfang und Auswirkungen weiter zunehmen, greifen Regulierungsbehörden weltweit ein. Vom EU Cyber Resilience Act über U.S. Executive Orders bis hin zu FDA requirements lautet die Botschaft eindeutig: „Organisationen müssen Transparenz, Rückverfolgbarkeit und Verantwortlichkeit über ihre gesamte Software-Lieferkette hinweg nachweisen."

KONFORMITÄTS-ANFORDERUNGEN FÜR DIE SOFTWARE LIEFERKETTEN-SICHERHEIT

Regulierungsbehörden und Regierungen übernehmen zunehmend eine aktive Rolle dabei, wie Unternehmen ihre Software-Lieferketten absichern müssen. Was früher als „Nice-to-have“ galt, ist heute eine zwingende Voraussetzung.

Neue Gesetze und Rahmenwerke erhöhen branchen- und regionenübergreifend die Anforderungen an Transparenz, Schwachstellenmanagement, sichere Entwicklung und Lieferantenverantwortung.

In diesem Abschnitt werden die wichtigsten und relevantesten Compliance-Vorgaben vorgestellt, die künftig die Herangehensweise von Unternehmen an die Sicherheit ihrer Lieferketten prägen werden.

Relevante Vorschriften und Normen

Diese Vorschriften beziehen sich direkt auf die Sicherheit der Software-Lieferkette – einschließlich SBOM, Schwachstellenmanagement, Monitoring und weiterer Bereiche. Unternehmen, die vernetzte Geräte oder firmwarebasierte Produkte in regulierten Branchen herstellen, müssen diese Vorgaben und Standards zwingend berücksichtigen.



Regulation / Authority	Region	Vendor/ Developer or Operator	Mandatory or Voluntary	Manu-facturing	Auto-motive	Medical	Tele-commu-nication	Critical Infra-structure	Description
EU Cyber Resilience Act (CRA)	EU/Global	Both	Mandato-ry (once enforced)	✓			✓	✓	Mandates security features, vul-nerability handling, and SBOMs across digital products.
Radio Equipment Directive (REDII)	EU/Global	Both	Mandatory	✓			✓	✓	Cybersecurity requirements for internet-connected radio equipment devices; mandatory by Aug 2025.
NIS2 Di-rective	EU	Operators Primarily	Mandatory	✓	✓	✓	✓	✓	Requires risk management, supplier controls, and incident reporting. Reduce operational risk by actively scanning for vul-nerabilities in assets or devices. It is relevant for organizations that fall under the NIS2 Directive.
UNECE WP.29/ R155	EU	Both	Mandatory		✓				Automotive regulation focused on cybersecurity management systems and continuous risk management.
Executive Order 14028	USA	Vendors/ Developers Primarily	Mandatory	✓	✓	✓	✓	✓	Promotes SBOMs, secure soft-ware development, and incident response; primarily aimed at federal agencies, but applicable broadly.
FDA Cyber-security Guidance	USA	Vendor/ Developer	Mandatory			✓			Requires SBOMs, threat models, and patchability for medical devices.
NIST 800-218 (SSDF)	USA	Both	Voluntary	✓	✓	✓	✓	✓	Framework for secure software development practices.
METI Cyber-security Guidelines	Japan/ Global	Both	Voluntary	✓	✓	✓	✓	✓	Covers SBOM use, supplier soft-ware risk, and secure develop-ment practices for embedded systems.
CSA CII Supply Chain Progra	Singapore	Both	Mandatory	✓	✓	✓	✓	✓	Outlines software supplier risk controls, secure development, and SBOM practices for critical infrastructure
ISO/SAE 21434	Global	Both	Not Direct-ly Manda-tory - But Required for UNECE R155		✓				Cybersecurity engineering stan-dard for road vehicles.
IEC 62443	Global	Both	Voluntary	✓				✓	Industrial and OT cybersecurity framework applicable to em-bedded devices.
IMDRF Cyber-security Guidance	Global	Both	Voluntary			✓			Defines SBOM practices for medical devices, covering supplier software, transitive dependencies, and lifecycle risk management.

Haftungsausschluss: Die in der obigen Tabelle aufgeführten Vorschriften und Standards stellen einige Beispiele dar, die laut ONEKEY teilweise für die Sicherheit der Software-Lieferkette relevant sind. Sie können für Ihr Unternehmen aufgrund Ihrer individuellen Sicherheitslage und Ihres betrieblichen Kontexts relevant sein. Die Botschaft der Regulierungsbehörden lässt sich immer schwerer ignorieren: Von Unternehmen wird erwartet, dass sie wissen, was in ihrer Firmware enthalten ist, woher sie stammt und wie sie verwaltet wird. Es reicht nicht aus, erst zu reagieren, wenn etwas schiefgelaufen ist. Das Ziel besteht nicht darin, Probleme nach der Veröffentlichung zu beheben, sondern sie von vornherein zu vermeiden. Das bedeutet, dass die Sicherheit nach links verlagert und in die Entwicklungsabläufe integriert werden muss, anstatt sie als Last-Minute-Add-on zu behandeln.

Teilweise relevante Vorschriften und Normen

Diese Vorschriften beziehen sich zwar nicht ausdrücklich auf die Software-Lieferkette, betreffen jedoch zentrale Bereiche, in denen deren Sicherheit eine entscheidende Rolle spielt – etwa Gerätesicherheit, sichere Beschaffung oder die Integrität von Updates. Gerade Unternehmen, die grenzüberschreitend tätig sind oder in risikobehafteten Branchen agieren, sollten diese Vorgaben daher besonders aufmerksam berücksichtigen.

Regulation / Authority	Region	Vendor/ Developer or Operator	Mandatory or Voluntary	Manu-facturing	Auto-motive	Medical	Tele-commu-nication	Critical Infra-structure	Description
EU MDR (Medical Device Regula-tion)	EU	Vendor/ Developer	Mandatory			✓			Requires software validation and verification if medical devices are shipped with embedded software.
ETSI EN 303 645 (IoT)	EU/Global	Vendor/ Developer	Voluntary	✓	✓		✓		Baseline cybersecurity stan-dard for consumer IoT devices, including updates and access controls.
Protective Security Policy Framework (PSPF)	Australia/ Global	Both	Mandatory				✓	✓	Primarily addresses general sup-ply chain and procurement risks.

Haftungsausschluss:Die in der obigen Tabelle aufgeführten Vorschriften und Standards stellen einige Beispiele dar, die laut ONEKEY teilweise für die Sicherheit der Software-Lieferkette relevant sind. Sie können für Ihr Unternehmen aufgrund Ihrer individuellen Sicherheitslage und Ihres betrieblichen Kontexts relevant sein.

Die Botschaft der Regulierungsbehörden lässt sich immer schwerer ignorieren: Von Unternehmen wird erwartet, dass sie wissen, was in ihrer Firmware enthalten ist, woher sie stammt und wie sie verwaltet wird. Es reicht nicht aus, erst zu reagie-ren, wenn etwas schiefgelaufen ist. Das Ziel besteht nicht darin, Probleme nach der Veröffentlichung zu beheben, son-dern sie von vornherein zu vermeiden. Das bedeutet, dass die Sicherheit nach links verlagert und in die Entwicklungsabläufe integriert werden muss, anstatt sie als Last-Minute-Add-on zu behandeln.

SHIFT SECURITY LEFT UM VERSTÖSSE FRÜHZEITIG ZU VERHINDERN

Moderne eingebettete Systeme basieren auf komplexen Liefer-ketten, die Komponenten von Drittanbietern, Open-Source-Li-braries, vom Hersteller bereitgestellte Binärdateien und KI-gene-rierten Code umfassen.

Je stärker diese Elemente ineinandergreifen, desto schwieriger wird es, bekannte oder unbekannte Schwachstellen vor der Bereitstellung zu erkennen. Tritt ein Fehler auf, sind die Reparaturen häufig teuer – oder schlimmer noch: das eingebettete System ist bereits im Einsatz. Hier setzt das Konzept des „Shifting Security Left“ an. Sicherheit wird dabei nicht als abschließende Kontrolle am Ende des Prozesses verstanden, sondern frühzeitig in den Entwicklungsablauf integriert. Transparenz und Validierung beginnen schon auf Ebene des Codes, der Komponenten und der Builds. Für Unternehmen, die Firmware entwickeln, ist dieser Ansatz entscheidend, um nachgelagerte Risiken zu minimieren und die Zuverlässigkeit der Produkte sicherzustellen

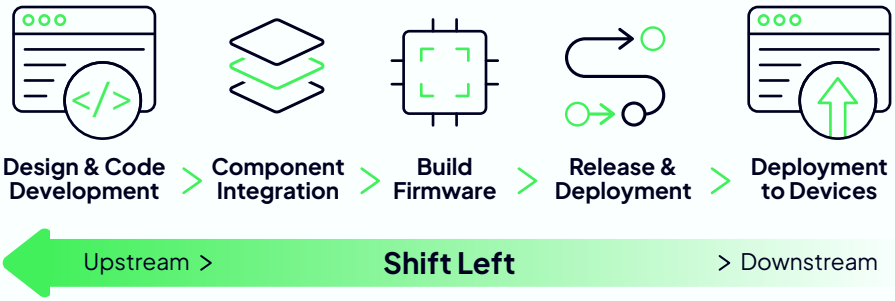


Abbildung 5: Shifting Left konzentriert sich stärker auf Sicherheitsmaßnahmen im Upstream Bereich als im Downstream Bereich

Konzept von Shift Left

Im Kern bedeutet Shift Left, dass Sicherheits- und Qualitätskontrollen frühzeitig stattfinden – näher an dem Zeitpunkt, an dem Code geschrieben, Komponenten integriert und die Firmware erstellt wird. Ziel ist es, Risiken nicht erst am Ende zu erkennen, sondern so früh wie möglich. Gerade in Lieferketten für eingebettete Systeme haben Teams nicht immer Zugriff auf den Source Code. Firmware wird oft nur als kompilierte Binärdatei von Drittanbietern oder Lieferanten bereitgestellt. Hinzu kommt, dass Teile des Codes oder Build-Skripte zunehmend mithilfe von KI generiert werden – mit nur begrenzter Transparenz über die zugrunde liegenden Abhängigkeiten. Deshalb werden klassische Sicherheitsprüfungen häufig übersprungen oder verzögert.

Die Folge: Schwachstellen, Lizenzprobleme oder veraltete Komponenten gelangen ungeprüft in die Bereitstellung. Shift Left ändert das. Es setzt auf frühzeitige Validierung, wiederholbare Richtliniendurchsetzung und proaktive Entscheidungsfindung, sodass Risiken gestoppt werden können, bevor sie zu einer Belastung werden.

Bedarf in der Software-Lieferkette

Wie bereits in der Einleitung erwähnt, ist die Software-Lieferkette kein linearer Prozess, sondern ein komplexes Geflecht aus internen Teams, externen Anbietern, Legacy-Code und weiteren Faktoren. Durch die Vielzahl an Beteiligten und beweglichen Teilen werden bekannte wie unbekannte Schwachstellen oft erst in späteren Phasen entdeckt – zu einem Zeitpunkt, an dem ihre Behebung deutlich schwieriger und teurer ist.

Shift Left setzt genau hier an, indem es:

- Schwachstellen und Fehlkonfigurationen bereits in frühen Phasen aufdeckt,
- verhindert, dass nicht konformer Code in die Pipeline gelangt,
- eine schnellere und sicherere Bereitstellung ermöglicht – mit weniger Überraschungen in späteren Phasen.

Das Ergebnis: mehr Sicherheit, kürzere Release-Zyklen, weniger Verzögerungen sowie geringere Kosten für Patches, Nachbesserungen und Produktrückrufe. Gleichzeitig stärkt dieser Ansatz das Vertrauen der Kunden.

Die Sichtbarkeit auf Binärniveau steht an erster Stelle

Wenn Sie dieses Whitepaper bis hierhin gelesen haben, ist Ihnen vermutlich klar geworden, dass die Überprüfung von Source Code in eingebetteten Umgebungen nur selten möglich ist. Komponenten werden häufig als Binärdateien geliefert – ohne Dokumentation, mit unklaren Lizenzbedingungen und unbekanntem Inhalt. Genau deshalb ist die Sichtbarkeit auf Binärcodeebene die Grundlage für Shift Left in eingebetteten Systemen. Falls Sie sich fragen, was eine Binärdatei ist, hier eine einfache Erklärung: Eine Binärdatei ist kompilierter Source Code, der in Maschinensprache übersetzt wurde, damit er von einer CPU ausgeführt werden kann. In eingebetteten Systemen besteht die Firmware in der Regel aus solchen Binärdateien, ergänzt durch Skripte und Konfigurationsdateien, die gemeinsam die Funktionsweise eines Geräts steuern. Von einer Firmware-Binärdatei spricht man meist dann, wenn all diese Elemente in einem einzigen komprimierten Image oder Archiv gebündelt sind, das anschließend auf das Gerät geladen wird. Die Sichtbarkeit auf Binärcodeebene ermöglicht es Teams:



- SBOMs zu erstellen, auch wenn der Source Code nicht verfügbar ist.
- Binärdateien von Drittanbietern und Lieferanten vor der Integration zu analysieren.
- Schwachstellen frühzeitig zu erkennen und zu entscheiden, ob eine Komponente für die Freigabe geeignet ist.

Ohne diese Einblicke ist es nahezu unmöglich, Sicherheit in einer frühen Phase durchzusetzen. Shift Left bliebe dann reine Theorie – ohne praktische Umsetzung.

Shift Left ist in erster Linie eine Denkweise, die jedoch die passenden Tools und wiederholbare Prozesse erfordert. Im nächsten Abschnitt betrachten wir die Best Practices auf Binärcodeebene, mit denen sich diese Strategie in Ihrer gesamten Embedded-Lieferkette praktisch umsetzen lässt.

SOFTWARE LIEFERKETTEN-SICHERHEIT: BINÄREBENE BEST PRACTICES

Die Sicherung der Software-Lieferkette und die Umsetzung von Shift Left beginnen mit der Einführung geeigneter Best Practices auf Binärcodeebene. Diese sind längst kein „Nice-to-have“ mehr, sondern praxisnahe Werkzeuge, auf die sich Teams im Arbeitsalltag verlassen können. Sie helfen, Risiken zu reduzieren, die Einhaltung neuer Vorschriften zu unterstützen und von Anfang an robuste Firmware zu entwickeln. Gleichzeitig bringen sie klare Vorteile: schnellere Bereitstellung, weniger bekannte und unbekannte Schwachstellen in späteren Prozessphasen sowie mehr Vertrauen in jeden Release. Die Best Practices lauten wie folgt:

Software-Stücklisten (SBOMs) erstellen und validieren

SBOM (Software Bill of Materials) ist eine vollständige Liste aller Bestandteile Ihrer Firmware. Sie umfasst Libraries, Komponenten, Versionen usw. – unabhängig davon, ob sie intern entwickelt, von Lieferanten bereitgestellt oder durch KI generiert wurden.

Funktionsweise: SBOMs verschaffen Transparenz darüber, welche Inhalte in einer Binärdatei enthalten sind. Durch die Validierung wird sichergestellt, dass diese SBOM korrekt und vollständig ist.

Bedeutung für die Lieferkettensicherheit und Shift Left: Viele Libraries liegen nur vorkompiliert vor, ohne Zugriff auf den Source Code. SBOMs schaffen hier Einblick und ermöglichen es, Probleme zu identifizieren, bevor Komponenten in die Firmware integriert werden.

Software Quality Gates integrieren

Quality Gates sind Regeln oder Sicherheitskontrollen, die bestanden werden müssen, bevor Sie im Entwicklungsprozess fortfahren können.

Funktionsweise: Diese Kontrollen verhindern, dass Binärdateien mit kritischen Schwachstellen, unbekannten Komponenten oder fehlenden SBOMs in die Produktion gelangen. Sie können die Kontrollen an Ihre internen Risikorichtlinien oder gesetzlichen Anforderungen anpassen.

Bedeutung für die Lieferkettensicherheit und Shift Left: Bei direkter Anwendung auf Release-Kandidaten der Firmware helfen Quality Gates dabei, riskante Firmware frühzeitig zu identifizieren und zurückzuweisen, bevor sie in die Bereitstellung gelangt.

So lassen sich Geschäftsunterbrechungen verhindern, ein vorhersehbarer Release-Prozess etablieren und das Vertrauen in die Marke schützen.

Schwachstellenmanagement und Folgenabschätzung Bewertung

Identifizierung von Sicherheitsproblemen in der Firmware durch Scannen der kompilierten Binärdateien und Erkennen der tatsächlich relevanten Schwachstellen.

Funktionsweise: Ein gutes System kennzeichnet nicht nur Schwachstellen in Binärdateien, sondern ermittelt auch, ob diese in Ihrer Umgebung erreichbar, genutzt oder ausnutzbar sind.

Bedeutung für die Lieferkettensicherheit und Shift Left: Die Analyse auf Binärdateiebene ermöglicht es Teams, Risiken frühzeitig im Prozess zu bewerten, auch für die von Lieferanten bereitgestellte Firmware. Anstatt sich von Listen potenzieller Schwachstellen überwältigen zu lassen, können sich Teams auf diejenigen konzentrieren, die die Produktintegrität vor der Veröffentlichung gefährden könnten.

Transparenz über den gesamten Lebenszyklus hinweg gewährleisten

Lebenszyklus-Transparenz bedeutet, Schwachstellen in Firmware und Komponenten über den gesamten Prozess hinweg nachzuverfolgen. Vom Build über die Bereitstellung bis hin zum laufenden Betrieb.

Funktionsweise: Selbst bei kompilierten Binärdateien ermöglicht Lebenszyklus-Transparenz einen klaren Überblick: Welche Komponenten sind wo ausgerollt, welche Versionen laufen aktuell und ob diese Schwachstellen oder veraltete Bestandteile enthalten.

Bedeutung für die Lieferkettensicherheit und Shift Left: Lebenszyklus-Transparenz ist nicht nur eine Überwachung nach der Bereitstellung. Sie liefert auch Erkenntnisse für frühere Builds. Wenn Sie wissen, dass bestimmte Komponenten problematisch oder risikobehaftet sind, können Sie sie auf eine Sperrliste setzen und verhindern, dass sie in zukünftigen Releases enthalten sind.

Flexibilität bieten – Benutzerdefinierte Regeln

Mit benutzerdefinierten Regeln kann Ihr Team festlegen, was „sicher“ für Ihr Produkt, Ihre Lieferkette und Ihr Risikoprofil bedeutet.

Funktionsweise: Anstatt sich nur auf standardisierte Scanregeln zu verlassen, können Sie Richtlinien auf Metadaten auf Binärniveau anwenden, z. B. auf gesperrte Libraries oder von Lieferanten bereitgestellte Firmware Komponenten.

Bedeutung für die Lieferkettensicherheit und Shift Left: Sicherheit ist keine Einheitsgröße.

Mit benutzerdefinierten Regeln können Sie Probleme erkennen, die für Ihre Lieferkette oder Ihr regulatorisches Umfeld spezifisch sind, und diese frühzeitig beheben.

Best Practices zeigen nur dann Wirkung, wenn sie konsequent angewendet werden. Dies ist nur mit Automatisierung möglich.

AUTOMATISIERUNG VON BEST PRACTICES FÜR DIE LIEFERKETTENSICHERHEIT

Best Practices wirken nur, wenn sie konsequent angewendet werden. Genau das ist für viele Teams eine große Herausforderung. Manuelle Überprüfungen sind nicht skalierbar. Das gilt besonders in komplexen Umgebungen, in denen Firmware von mehreren Anbietern stammt, KI-generierte Komponenten in Binärform vorliegen und Release-Zyklen sehr kurz sind. Automatisierung schließt diese Lücke. Sie bindet Sicherheits- und Compliance-Schritte direkt in die täglichen Entwicklungs- und Integrations-Workflows ein. So müssen sich Teams nicht mehr darauf verlassen, dass Einzelpersonen an Prüfungen denken oder Dokumentationen manuell nachverfolgen. Die Automatisierung übernimmt diese Aufgaben zuverlässig und wiederholbar.

Die wichtigsten Gründe für die Wahl einer automatisierten Lösung

- **Automatisierte Binär-Analyse:** Die meisten Tools arbeiten entweder mit Source Code oder einer sauberen SBOM. Beides ist in eingebetteten Umgebungen jedoch oft nicht verfügbar. Die automatisierte Analyse auf Binärebene löst dieses Problem. Sie untersucht kompilierte Firmware direkt und erkennt dabei Komponenten, Konfigurationen und Libraries – ganz ohne Transparenz aus vorgelagerten Quellen. So lassen sich Sicherheitsbewertungen auch dann durchführen, wenn der Code geschlossen ist, extern bereitgestellt wird oder nur unzureichend dokumentiert wurde.
- **Automatisierte Validierung von KI-generiertem Code:** KI-gestützte Entwicklungstools werden in rasantem Tempo eingeführt. Sie ermöglichen es Entwicklern, Code schneller als je zuvor zu schreiben und zu integrieren. Doch diese Geschwindigkeit hat ihren Preis: halluzinierte Pakete, unsichere Standardeinstellungen und versteckte Schwachstellen. Die automatisierte Überprüfung von KI-generiertem Code und seinen Abhängigkeiten stellt sicher, dass alles, was eingebunden wird, sicher, vertrauenswürdig und konform ist. Damit wird die Einführung von KI-Tools einfacher – Entwickler können schnell arbeiten, ohne unnötige Risiken in die Lieferkette einzuschleusen.

- **Automatisierung risikobewusster Entscheidungen vor der Auslieferung:** Go/No-Go-Entscheidungen basieren oft auf unvollständigen oder unklaren Risikodaten. Automatisierte Systeme kombinieren Schwachstellenerkennung mit Auswirkungsanalysen. So wird klar, welche Probleme tatsächlich ausnutzbar sind und welche nur existieren. Das erleichtert die Priorisierung der richtigen Korrekturen und verhindert unsichere Builds frühzeitig, ohne durch irrelevante Warnungen zu verzögern.
- **Echtzeit-Einblick in die eingesetzte Firmware:** Sicherheit endet nicht mit der Auslieferung. Automatisierte Überwachung verfolgt eingesetzte Firmware-Versionen und erkennt im Laufe der Zeit bekannte oder unbekannte Schwachstellen. Statt manueller Nachprüfungen oder seltener Audits scannt das System kontinuierlich und liefert Echtzeit-Einblicke, auch wenn Patches nur unregelmäßig erfolgen.
- **Durchsetzung von Richtlinien mit benutzerdefinierten Regeln:** Jedes Unternehmen hat eigene Risikotoleranzen, Compliance-Anforderungen und technische Umgebungen. Manuelle Regelanwendung ist zeitaufwändig und inkonsistent. Mit Automatisierung werden Sicherheitsregeln in den Build- und Validierungsprozess integriert – nicht erst nachträglich geprüft. Ob es um das Verbot veralteter Bibliotheken, das Blockieren nicht genehmigter Lizenzen oder das Kennzeichnen bestimmter Lieferanten-Binärdateien geht: Diese Regeln greifen automatisch. So werden interne Richtlinien zu verlässlichen Quality Gates – ohne die Teams auszubremsen.
- **Integrierte Compliance-Sicherung:** Die manuelle Vorbereitung von Audits ist mühsam und oft unvollständig. Automatisierung erfasst dagegen alles: SBOMs, Schwachstellenberichte, Regelverstöße und Firmware-Versionshistorien. Dadurch sind Teams jederzeit auditbereit und können vollständige Rückverfolgbarkeit für Standards wie EU CRA, RED2, NIST und andere nachweisen – ohne nachträgliche Dokumentation.
- **Automatisierte Erkennung von Zero-Day-Schwachstellen und Lizenzrisiken:** Einige Risiken sind tief in der Firmware verankert, andere sind noch unbekannt. Eine automatisierte Binäranalyse kann helfen, beide Arten aufzudecken. Sie erkennt frühe Anzeichen von Zero-Day-Schwachstellen, indem sie riskante Muster oder unsichere Funktionen identifiziert – auch ohne bereits veröffentlichte CVEs. Gleichzeitig scannt sie alle Drittanbieter- und Open-Source-Komponenten auf ihre Lizenzen. Das gilt auch für von Lieferanten bereitgestellte oder Closed-Source-Firmware, sodass nichts unentdeckt bleibt.



Vorteile der Automatisierung

Um mit der modernen Firmware Entwicklung Schritt zu halten, sehen sich Teams mit einer zunehmenden Komplexität durch Code von Drittanbietern, KI-generierte Vorschläge und undurchsichtige Binärdateien konfrontiert. Manuelle Prozesse können mit dem Umfang und der Geschwindigkeit der modernen Software-Lieferkette einfach nicht Schritt halten, insbesondere bei eingebetteten Systemen. Hier kommt die Automatisierung ins Spiel. Automatische Prüfungen, Richtliniendurchsetzungen und weitere Maßnahmen werden in jede Phase der Firmware-Entwicklung und -Bereitstellung eingebettet. So können Unternehmen den Schritt von reaktiven Patches hin zu proaktivem Schutz vollziehen. Die nachstehende Tabelle zeigt, was sich ändert, wenn Sie auf eine automatisierte Lösung umsteigen.

With Automated Solution	**Without Automated Solution**
Automated SBOMs Generated from Binaries	SBOMs are missing or incomplete
Automated Verification of AI-Generated Code and Dependencies	AI-Generated Code Adopted Without Instant Validation
Automated Scans with Context-rich Triage	Manual Vulnerability Checks
Integrated Checks in CI/CD and Supplier Intake	Delayed Security Validations
Active and Automated Firmware Monitoring	Blind Spots Post-deployment
Proactive and Automated Auditable Vulnerability and Compliance Check	Reactive and Rushed Compliance

Die Sicherung der Software-Lieferkette ist nicht nur ein technisches Ziel, sondern auch ein geschäftlicher Vorteil. Erfahren Sie, wie unser Kunde seine Lieferkette gesichert hat.

ROI UND GESCHÄFTS-AUSWIRKUNGEN DURCH AUTOMATISIERUNG

Die Sicherung der Software-Lieferkette muss nicht zwangsläufig zu Verzögerungen führen, sondern kann sogar das Gegenteil bewirken. Wenn sie richtig und automatisiert durchgeführt wird, reduziert sie operative Risiken, stärkt das Vertrauen der Kunden und beschleunigt die Lieferzyklen. ONEKEY schafft messbaren Mehrwert, indem es die Firmware Sicherheit vorhersehbarer, skalierbarer und besser auf die tatsächlichen Geschäftsziele abstimmt.

Wie Swisscom pro vermiedenem Vorfall 400.000 USD einspart

Mit mehr als 1,8 Millionen Kundengeräten im Feld steht für Swisscom bei jedem Firmware-Upgrade viel auf dem Spiel. Ein einziger fehlerhafter Rollout kann das Unternehmen schätzungsweise 374.000 CHF (~400.000 USD) an Support- und Reparaturkosten mit sich bringen. Durch die Integration von ONEKEY als Quality Gate in seine Entwicklungs- und Beschaffungs Workflows führt Swisscom nun jährlich eine tiefgehende Analyse von über 80 Firmware-Images durch und erkennt kritische Probleme, bevor sie die Kunden erreichen. Dieser proaktive Ansatz verhindert nicht nur kostspielige Upgrade Fehlschläge, sondern stärkt auch die Position von Swisscom in Lieferantenverhandlungen und festigt seine Marke als Security-First Telekommunikationsanbieter. Für Swisscom ist ONEKEY nicht nur ein Sicherheitstool; es ist ein messbarer Treiber für operative Effizienz, Kundenzufriedenheit und ROI. Lesen Sie die vollständige Swisscom-Fallstudie hier: https://www.onekey.com/resource/swisscom

ONEKEY's platform Vorteile

- **Die gesamte Firmware-Lieferkette wird zu 100 % automatisch überprüft:** Alle intern entwickelten oder von Anbietern bereitgestellten Firmwares werden automatisch auf Schwachstellen, fehlende SBOMs und Lizenzrisiken überprüft.
- **< 15 Minuten für die Firmware-Analyse – nicht Tage oder Wochen:** Automatisiertes Scannen auf Binärcodeebene liefert innerhalb von Minuten umfassende Einblicke in die Firmware
- **+60 % weniger Fehlalarme:** Kontextbewusstes Scannen hilft, echte Risiken zu beseitigen und gleichzeitig Störsignale und Fehlalarme zu reduzieren, die Teams ausbremsen.

- **10 oder 10.000 Firmware-Images – ONEKEY skaliert mit Ihnen:** Von Nischenprodukten bis hin zu globalen Flotten unterstützt ONEKEY die sichere Aufnahme und Überwachung über gesamte Produktlinien hinweg – ohne zusätzlichen Aufwand.

ZENTRALE ERKENNTNISSE

- **Die Lieferkettensicherheit ist unverzichtbar:** Eingebettete Produkte kombinieren internen Code, Open Source, Lieferanten Binaries und KI-generierte Code Schnipsel. Das vergrößert undurchsichtige Bereiche und verteilt Angriffe über die gesamte Lieferkette hinweg.
- **Realitycheck:** Teilanalysen, fehlende SBOMs und schwache Überwachung nach der Auslieferung lassen Risiken lange nach Release bestehen.
- **Strikte Regulierung:** CRA, RED, NIS2, EO 14028, FDA, IEC 62443 und andere verlangen Transparenz, Rückverfolgbarkeit und Verantwortlichkeit über die gesamte Lieferkette hinweg.
- **Shift Security Left:** Validierung bereits in der Design-, Komponenten- und Build Phase verhindert, dass Probleme bis in Staging, Produktion oder den Betrieb gelangen.
- **Binärsichtbarkeit ist notwendig:** Wenn kein Source-Code verfügbar ist, bleibt nur die Analyse der Firmware-Binärdatei, um zu wissen, was tatsächlich ausgeliefert wird.
- **Automatisiere die kritischen Arbeiten:** SBOM Erstellung, Prüfung von Komponenten (inkl. KI-generiertem Code), Scans, Triage und Policy Checks sollten automatisch in CI/CD und beim Lieferanteneingang erfolgen.
- **KI bringt neue Fallen:** Slopsquatting und „Vibe Coding“ können Fake- oder unsichere Pakete einschleusen – automatisierte Abhängigkeitsprüfung ist daher entscheidend.
- **Sicherheit über die Auslieferung hinaus:** Kontinuierliches Monitoring der eingesetzten Firmware zeigt Exposition gegenüber neuen CVEs auf und liefert Erkenntnisse für den nächsten Build.

Shift Left, Binärdateien auslesen und Kontrollen automatisieren, um die Lieferkettensicherheit zu einer täglichen Praxis zu machen. Sind Sie an einem weiteren Austausch mit unseren Sicherheitsexperten interessiert, wie Sie Produkt Cybersicherheit & Compliance durch ein effektives und effizientes SBOM Management erreichen können?

Bitte kontaktieren Sie unsere Sicherheitsexperten unter:
experts@onekey.com – oder rufen Sie an unter +49 211 1587 4104

WEITERFÜHRENDE INFORMATIONEN

Den EU Cyber Resilience Act verstehen und die Cybersicherheit von Produkten gewährleisten

In diesem Whitepaper werden die wichtigsten Anforderungen des EU Cyber Resilience Acts (CRA) erläutert und es wird aufgezeigt, wie Hersteller, Importeure und Händler von vernetzten Geräten ihre Verpflichtungen in Bezug auf Cybersicherheit und Compliance erfüllen können. Es behandelt die Compliance auf Produktebene, das SBOM-Management und den Umgang mit Schwachstellen, die für die Lieferkette von entscheidender Bedeutung sind. Leser erhalten einen Überblick über den Fahrplan zur Vorbereitung auf das CRA und zur Einbettung einer kontinuierlichen Sicherheitsvalidierung über den gesamten Produktlebenszyklus hinweg.

https://www.onekey.com/resource/understanding-the-eu-cyber-resilience-act-and-achieve-product-cybersecurity-compliance

Vorschriften für die Software-Lieferkette: Wie lässt sich ein effektives und effizientes SBOM-Management erreichen?

Dieses Whitepaper bietet einen detaillierten Einblick, wie Unternehmen SBOMs effektiv erstellen, validieren und pflegen können, um die Transparenz und Kontrolle in der Software-Lieferkette zu verbessern. Es nutzt das regulatorische Umfeld als Kontext, konzentriert sich jedoch auf SBOMs als Grundlage für Transparenz, Schwachstellenerkennung und Lieferantenverantwortung. Leser:innen erhalten ein praktisches Verständnis dafür, wie SBOMs für ein besseres Risikomanagement und eine effizientere Compliance in verschiedenen Software-Lieferketten eingesetzt werden können.

https://www.onekey.com/resource/software-supply-chain-regulations-how-to-achieve-effective-efficient-sbom-management

Bekämpfung von Risiken in der Software-Lieferkette mit IEC 62443 und SBOM

Dieses Whitepaper untersucht, wie das IEC-62443-Framework zur Sicherung industrieller Software-Lieferketten angewendet werden kann und wie SBOMs dessen Umsetzung für die Integrität von Komponenten und das Schwachstellenmanagement unterstützen. Es beleuchtet das Risikomanagement und die Lebenszyklussicherheit für Anbieter.

https://www.onekey.com/resource/tackling-software-supply-chain-risks-with-iec-62443-and-sbom

ONEKEY
onekey.com
+49 211 1587 41 04
info@onekey.com

SecurITy made in Germany

© 2025 ONEKEY. All rights reserved. Reproduction only permitted with the approval of ONEKEY. All brands listed are the brands of the respective owners. Errors, changes, and availability of the listed products, services, characteristics, and possible applications reserved. Software & services will be provided by ONEKEY. ONEKEY makes no guarantee for the information of third parties regarding characteristics, services and availability. ONEKEY reserves the right to make changes to products and services as a result of product development, even without prior notification. None of the statements and depictions represents legal advice or may be interpreted in such a manner. In case of deviations from the contract documents and general terms and conditions of ONEKEY and their affiliated companies and subsidiaries in conjunction with this document, the contract documents and general terms and conditions always have precedent over this document.

V20251001EN