Die 5 wichtigsten Strategien zur Sicherung Ihrer Softwarelieferkette

Fokus auf die Automatisierung des sicheren Code Signings, die Erkennung von Bedrohungen und Malware sowie die SBOM-Generierung

Die Häufigkeit und Schwere von Zwischenfällen in der Softwarelieferkette nimmt seit mehreren Jahren zu. Einem Bericht der Enterprise Strategy Group von TechTarget zufolge gab es 2023 bei 91 % der Unternehmen mindestens einen solchen Vorfall.

Betroffen sind sowohl große als auch kleine Unternehmen. Inzwischen hat fast jeder von den Angriffen auf die Lieferkette bei SolarWinds, CircleCI und 3CX gehört. Im März 2024 wurde der Betreuer eines sehr beliebten Linux-Komprimierungsprogramms durch einen Trick dazu gebracht, eine Backdoor in den Code einzubinden. Das Problem wurde entdeckt, bevor die meisten populären Linux-Distributionen es in Produktionsversionen veröffentlichten, aber die Auswirkungen hätten verheerend sein können.

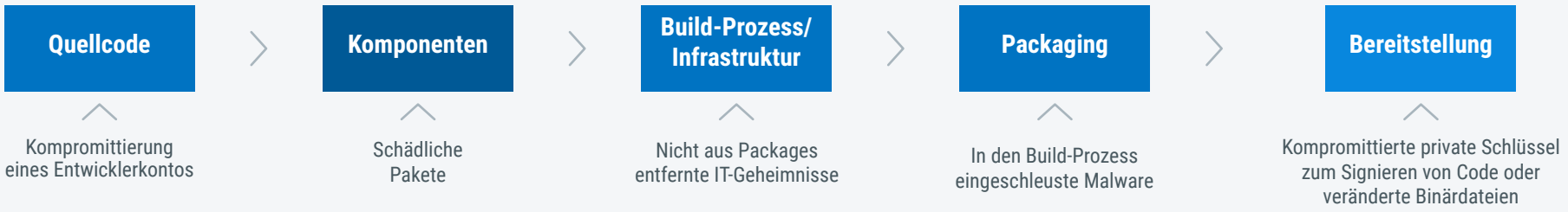
Obwohl grundlegende Sicherheitsmaßnahmen, wie z. B. die Zero-Trust-Prinzipien, eine erste Verteidigungslinie in Unternehmen bilden, ist ein Defense-in-Depth-Ansatz erforderlich, um die komplexen Softwarelieferketten und Softwareentwicklungszyklen von Unternehmen umfassend zu sichern.

Herausforderungen

Bedrohungsakteure verwenden eine Vielzahl von Taktiken, wenn sie eine Softwarelieferkette angreifen (siehe Abbildung 1). Sie könnten die geknackten Anmeldedaten eines Softwareentwicklers missbrauchen, um in ein Software-Build-System einzudringen, ungeschützte private Schlüssel für die Code-Signierung stehlen und missbrauchen, Malware in gängige Open-Source-Softwarepakete einschleusen oder eine Kombination aus all diesen Taktiken anwenden. Wie der Sunburst-Angriff auf SolarWinds im Jahr 2019 gezeigt hat, können die Angriffstaktiken ziemlich ausgeklügelt und komplex sein, aber wirksame Abwehrmaßnahmen müssen nicht so hochmodern sein. Die Einhaltung von Best Practices und Defense-in-Depth-Prinzipien im Rahmen eines konsistenten Sicherheitsansatzes sind in jeder Organisation möglich.

Oft gibt es jedoch auch strukturelle und organisatorische Herausforderungen: Softwareentwicklungsteams sträuben sich aus vielen Gründen gegen Sicherheitstools: Sie können umständlich sein, automatisierte CI/CD-Build-Pipelines unterbrechen oder die Veröffentlichung der Software verlangsamen.

Abbildung 1: Angriffe auf die Lieferkette können in jeder Phase stattfinden



Angriffe können in jeder Phase des Softwareentwicklungszyklus auftreten. Bereits vorhandene, für bestimmte Zwecke entwickelte Sicherheitstools erkennen möglicherweise einige Angriffsarten, aber nicht alle. Daher sollten Unternehmen sowohl „Shift Left"- als auch „Shift Right"-Strategien anwenden, um den gesamten Softwareentwicklungszyklus abzusichern.

Sicherheitsprodukte generieren mitunter Fehlalarme und selbst ihre legitimen Warnungen können als lästig und wenig hilfreich empfunden werden.

Die Sicherheitsteams stehen vor anderen Herausforderungen. Sie müssen Sicherheitsrichtlinien bereitstellen und die Einhaltung von Vorschriften bei vielen Softwareteams durchsetzen, die sowohl organisatorisch als auch geografisch über das Unternehmen verteilt sind. Darüber hinaus verwenden diese Teams oft unterschiedliche Programmiersprachen und -umgebungen, stellen Software auf verschiedenen Plattformen bereit und nutzen unterschiedliche Software für die Erstellung von Infrastruktur und Anwendungen. Diese Vielfalt kann die Definition, Durchsetzung und Überwachung von Softwaresicherheitsrichtlinien in einem Unternehmen extrem schwierig machen.

Schließlich müssen die meisten Organisationen, insbesondere in Regionen wie der EU, eine wachsende Zahl von staatlichen Vorschriften und privaten Standards einhalten. In mehreren Vorschriften und Branchenempfehlungen sind nicht nur Best Practices, sondern auch Anforderungen festgelegt, die Unternehmen befolgen müssen, um die Integrität der Softwarelieferkette zu gewährleisten. So verlangen zum Beispiel die US Executive Order zur Verbesserung der Cybersicherheit der Nation und der Cyber Resilience Act der EU, dass Softwarestücklisten (Software Bills of Materials, SBOMs) erstellt werden, um die Softwaretransparenz zu belegen. Die Sicherheitsüberlegungen für Code Signing des NIST beschreiben Schritte, die für eine sichere Codesignierung befolgt werden sollten. Im Payment Card Industry Data Security Standard sind in Anforderung 6 mehrere Vorgaben für die Entwicklung und Pflege sicherer Software und Systeme enthalten.

Diese Lösungsübersicht skizziert fünf Strategien oder Best Practices, die Unternehmen nutzen können, um diese Herausforderungen zu bewältigen, und wie DigiCert dabei helfen kann. Das wichtigste Merkmal dieser Best Practices ist, dass sie nicht nur in einzelnen Phasen des Softwareentwicklungszyklus (SDLC), sondern während des gesamten SDLC angewendet werden (siehe Abbildung 3, Seite 3).

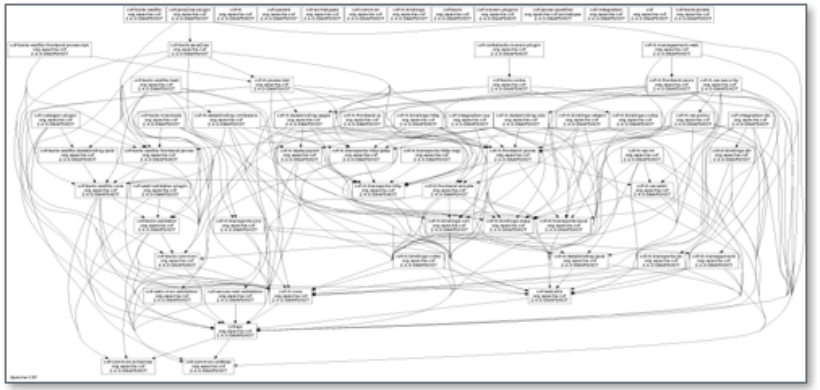
Nr. 1: Unterschätzen Sie die Komplexität Ihrer Software-Entwicklungsumgebungen nicht

Warum sind so viele Angriffe auf die Softwarelieferkette erfolgreich? Dafür gibt es viele Gründe: IT-Teams sind oft unterbesetzt und arbeiten isoliert voneinander mit verschiedenen Tools an unterschiedlichen Aufgaben.

Ein großes Unternehmen hat wahrscheinlich eine große und komplexe Angriffsfläche, die offen für verschiedene Angriffe aus der ganzen Welt ist. Einer der Hauptgründe ist jedoch zweifellos die Komplexität moderner Softwaresysteme.

Abbildung 2 zeigt eine strukturelle Darstellung des Apache-HTTP-Softwareprojekts im Jahr 2024. Was 1995 von acht Entwicklern begonnen wurde, umfasst heute etwa 2 Millionen Codezeilen, zu denen weltweit über 630.000 Personen beitragen.

Abbildung 2: Apache HTTP Softwareprojekt

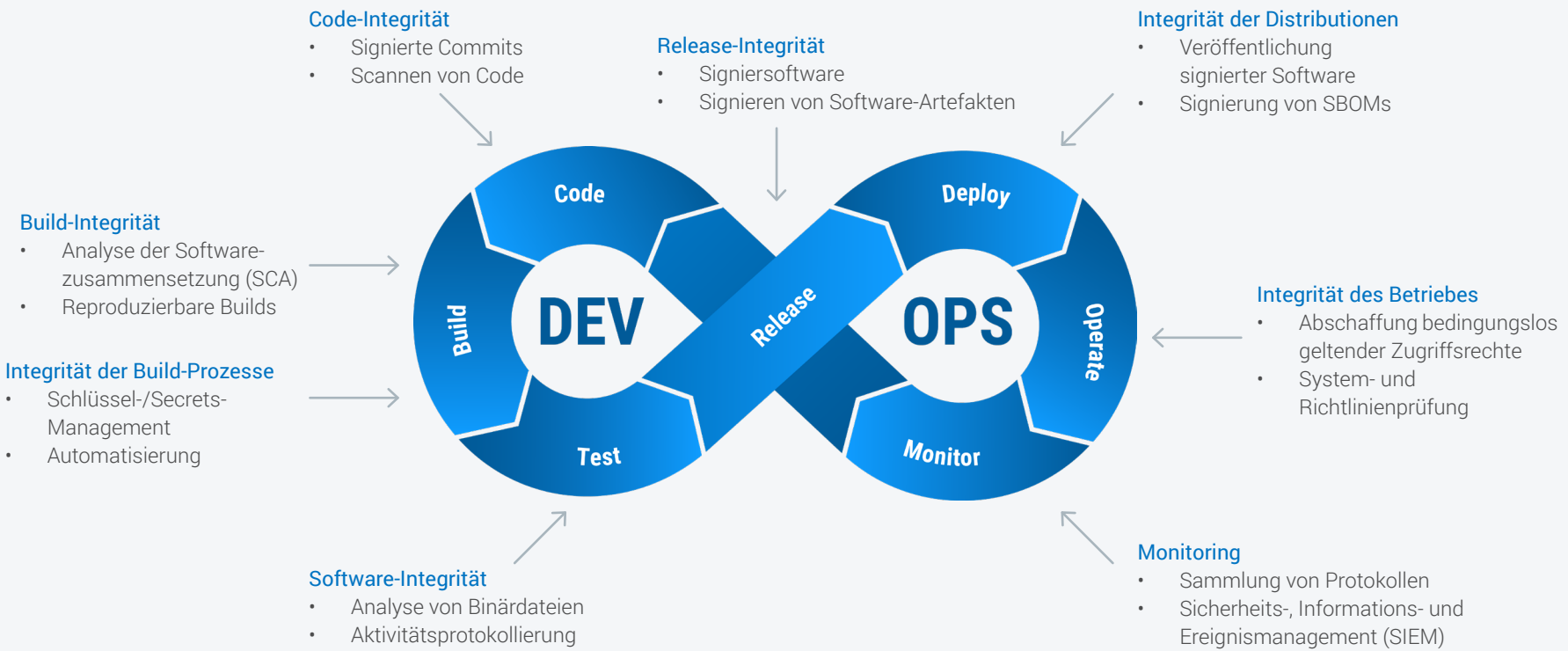


Viele wichtige Anwendungen in aller Welt bauen auf diesem Projekt auf, aber wahrscheinlich kann niemand von sich behaupten, mit allen Aspekten vollständig vertraut zu sein.

Wenn Sie im Internet tätig sind, verwenden Sie mit ziemlicher Sicherheit hin und wieder Apache HTTP, wahrscheinlich sogar ständig. Es hat verantwortliche Betreuer, und die Probleme sind überschaubar, aber zahlreich, wie bei jedem Projekt dieser Komplexität.

Selbst die Software, die Sie zur Gewährleistung der Sicherheit kaufen, ist nicht perfekt und muss durch andere Maßnahmen ergänzt werden, was wir als „Defense-in-Depth" bezeichnen. Ende 2023 stellte der Authentifizierungsdienst Okta fest, dass Angreifer seine Systeme infiltriert und die Identitätsdaten einer großen Anzahl von Okta-Nutzern gestohlen hatten. Wie KrebsOnSecurity es ausdrückte, gelang es ihnen zudem, „... Authentifizierungs-Token von einigen Okta-Kunden zu stehlen, die die Angreifer dann nutzen konnten, um Änderungen an Kundenkonten vorzunehmen, wie etwa das Hinzufügen oder Ändern von autorisierten Benutzerkonten."

Abbildung 2: Auf Unternehmensrichtlinien basierte Vorgehensweise und Transparenz



Als Teil der Nachuntersuchung des Vorfalls kündigte Okta eine verbesserte Unterstützung für mehrere Funktionen an, die überall genutzt werden sollten, wo sie verfügbar sind:

- Multifaktor-Authentifizierung für alle Konten
- IP-Bindung: Lassen Sie soweit möglich nur Verbindungen von oder zu bestimmten Diensten über bestimmte, als legitim bekannte IP-Adressen zu. Dies ist eine Möglichkeit, die – von Angreifern häufig genutzte – Wiederverwendung von entwendeten HTTP-Token zu verhindern.
- Zudem sollten Sie bedingungslos geltende Zugriffsrechte so weit wie möglich abschaffen. Damit sind standardmäßige, aber oft unnötige Berechtigungen für ein Konto gemeint. Beschränken Sie die Kontoberechtigungen auf diejenigen, die für die Durchführung der Aufgaben erforderlich sind, für die das Konto genutzt wird.

Nr. 2: Schützen Sie die Authentizität und Manipulationssicherheit Ihrer Software

Code Signing ist eine grundlegende Sicherheitsmaßnahme, die seit über 30 Jahren angewandt wird. Es verwendet Kryptographie, um eine digitale Signatur eines Softwarepakets (ausführbare Datei, Bibliothek, Anwendung usw.) zu erstellen. Benutzer, die das Paket erhalten, können mit kryptografischer Software und dem öffentlichen Schlüssel des Softwarehändlers prüfen, ob die Software authentisch ist, d. h. ob der angebliche Herausgeber der Software sie tatsächlich erstellt hat und ob der Code seit der Herausgabe nicht von Dritten manipuliert wurde. Viele Betriebssysteme wie iOS, macOS und Windows prüfen die digitale Signatur von Software, bevor sie Pakete installieren und ausführen.

Die Methode ist so effektiv, dass Hacker Code-Signing-Systeme angegriffen haben, um die Code-Signing-Zertifikate und privaten Schlüssel von Unternehmen zu stehlen,

zu manipulieren und zu missbrauchen. Gestohlene Codesignaturschlüssel waren eine Schlüsselkomponente des Stuxnet-Wurms, eines staatlich gesponserten Sabotageangriffs auf die iranischen Urananreicherungsanlagen. Stuxnet war ein ausgeklügelter Angriff, der vier Zero-Day-Schwachstellen in Windows ausnutzte und 2010 öffentlich bekannt wurde.

Wie bei allen PKI-Anwendungen ist es von entscheidender Bedeutung, dass der Unterzeichner die Unterzeichnungsschlüssel des privaten Codes schützt. Aus diesem Grund verlangt das Certificate Authority/Browser Forum, dass private Schlüssel, die mit öffentlich vertrauenswürdigen Code Signing-Zertifikaten verbunden sind, in einem Gerät gespeichert werden, das die Anforderungen von FIPS 140 Level 2, Common Criteria EAL 4+ erfüllt, z. B. in einem Hardware-Sicherheitsmodul (HSM).

Während die sichere Speicherung von Schlüsseln ein notwendiger erster Schritt ist, erfordert die Sicherung des Code Signing im gesamten Unternehmen zusätzliche Sorgfalt, schon allein deshalb, weil die gestohlenen Anmeldedaten eines einzigen Nutzers einem böswilligen Akteur den Zugriff auf den sicheren Speicher ermöglichen könnten.

Stattdessen empfehlen wir die Verwendung eines Code-Signing-Systems der Enterprise-Klasse, das viele zusätzliche Sicherheitsmaßnahmen bietet, darunter:

- Automatisierung der Verwaltung von Code-Signing-Zertifikaten und Schlüsseln, einschließlich Rotation und Widerruf.
- Ein Genehmigungsverfahren, das nicht nur festlegt, wer einen Code-Signing-Schlüssel verwenden darf, sondern auch, wer die Nutzung genehmigen muss und unter welchen Umständen der Schlüssel genutzt werden darf (z. B. Tageszeit, Build-Maschine usw.)
- Ein rollenbasiertes System, das festlegt, welche Benutzer Signiervorgänge durchführen dürfen, welche berechtigt sind, Signiervorgänge zu genehmigen, und welche sie überwachen dürfen.
- Wenn Genehmigungen erforderlich sind, sollte das System Mehrfachgenehmigungen und m-von-n-Genehmigungen zulassen, d. h. die Genehmigung durch eine bestimmte Anzahl m aus einer Gruppe von n genehmigungsberechtigten Benutzern.
- Aufrechterhaltung einer unveränderlichen Aufzeichnung aller Code-Signatur-Aktivitäten, auf die im Falle eines kompromittierten Zertifikats Bezug genommen werden kann.
- Ein zentrales Code-Signing-System, das eine große Anzahl verteilter Code-Signing-Vorgänge und viele verschiedene Plattformen und Tools für die Softwarebereitstellung und -entwicklung unterstützt.
- Ein richtlinienbasierter Ansatz für Kontrollmaßnahmen, der im Wesentlichen die in diesem Dokument beschriebenen Best Practices umsetzt, bevor Software freigegeben werden kann.
- Signieren von übergangsweisen Software-Artefakten während des gesamten Softwareentwicklungszyklus, einschließlich Quellcode vor dem Commit, Build-Skripten und Build-Recipes sowie Container- und anderen Software-Infrastrukturdateien
- Unternehmensweite Transparenz sowie Richtliniendefinition und -durchsetzung von einer zentralen Stelle aus, unabhängig davon, wo sich die verschiedenen Softwareteams befinden

Wenn Sie eine schriftliche Code-Signing-Richtlinie benötigen, sind Sie bei uns genau richtig. Beginnen Sie mit unserer herunterladbaren Vorlage, um ganz einfach eine auf Ihr Unternehmen zugeschnittene Code-Signing-Richtlinie zu erstellen.

Nr. 3: Suchen Sie nach Bedrohungen, Schwachstellen und Malware

Zwar kann Code Signing Manipulationen nach der Veröffentlichung der Software verhindern, aber was ist, wenn die Software bereits vor dem Signieren und Veröffentlichen manipuliert wird? So könnte beispielsweise ein Angreifer mit einem manipulierten Build-System Malware in das Quellcode-Repository eines Unternehmens einschleusen. Oder ein Softwareentwickler könnte ein Open-Source-Paket herunterladen, in dem Malware versteckt ist. Oder es könnte eine bisher unbekannte versteckte Schwachstelle entdeckt werden.

In eine Open-Source-Bibliothek eingebetteter bösartiger Code ist seit langem ein Problem und die Ursache für viele Schwachstellen in der Softwarelieferkette. In ihrem Bericht über den Zustand der Sicherheit der Softwarelieferkette beschreibt ReversingLabs eine Kampagne namens „Operation Brainleeches“, die im Mai 2023 begann. Brainleeches nutzte npm-Pakete, um Dateien zu hosten, die in weit verbreiteten Phishing-Kampagnen verwendet wurden, die auf Microsoft 365-Kunden abzielten, um deren Anmeldedaten zu erhalten. Es gibt unzählige Beispiele für solche Fallen in öffentlichen, angesehenen Open-Source-Repositorys.

Verschiedene Sicherheitstools auf dem Markt befassen sich mit einigen dieser Probleme, wie z. B. die statischen Sicherheitstests für Anwendungen (SAST). Diese Tools konzentrieren sich jedoch oft nur auf einen einzigen Aspekt (z. B. Schwachstellen bei Open-Source-Software) oder auf eine bestimmte Phase des SDLC.

Stattdessen ist eine umfassende Methode erforderlich, um den endgültigen Software-Quellcode und die Binärdateien auf Bedrohungen, Schwachstellen, das Vorhandensein eingebetteter Geheimnisse und Malware zu überprüfen. Diese Überprüfung sollte unabhängig von der ursprünglichen Quelle der Softwarekomponenten erfolgen: Open-Source-Code, kommerzielle Bibliotheken von Drittanbietern oder vom Unternehmen selbst geschriebener Code. Viele Unternehmen gehen bei der Vermeidung von Malware neue Wege, indem sie beispielsweise Open-Source-Repositorys mit ausgewählten Inhalten bereitstellen.

Nr. 4: Sorgen Sie für Software-Transparenz (SBOM)

Wie bereits erwähnt, verlangen staatliche und branchenspezifische Vorschriften von Softwareherstellern immer häufiger, dass sie die Inhaltsstoffe ihrer Software offenlegen, ähnlich wie Etikette mit Nährwertangaben die Inhaltsstoffe von Lebensmitteln auflisten. Diese als Softwarestücklisten (SBOMs) bekannten Dokumente katalogisieren alle Komponenten eines Softwarepakets, unabhängig von der Quelle der Komponente (z. B. Open Source, kommerzielle Bibliothek usw.). SBOMs können nicht nur dazu verwendet werden, die „Softwarebestandteile" zu katalogisieren, sondern auch, um die Qualität dieser Bestandteile zu bewerten, z. B. die verwendeten Versionen, bekannte Sicherheitslücken usw.

Selbst wenn eine solche Softwarestückliste nicht vorgeschrieben ist, empfiehlt es sich für Unternehmen, sie zu generieren, um:

- **Änderungen von einem Build zum nächsten zu verstehen:** Technologieexperten kennen die Frustration bei der Fehlersuche zwischen zwei vermeintlich identischen Softwareversionen. Eine Softwarestückliste ermöglicht es Unternehmen, wie ihr Pendant in der physischen Fertigung, zu verstehen, welche Bestandteile für die Reproduktion ihrer Software erforderlich sind und wie sich diese Bestandteile im Laufe der Zeit verändern.
- **die Angriffsfläche ihrer Software zu bewerten und zu priorisieren:** Suchen Sie nach anfälligen oder anvisierten Versionen von Bibliotheken oder Komponenten, überprüfen Sie Softwareabhängigkeiten und suchen Sie nach fehlenden vorgeschriebenen Sicherheitspatches.
- **auf veraltete Komponenten** wie fehlende Abhilfemaßnahmen und unsichere Code-Signierungsverfahren **zu achten**.
- **proaktive Abwehrmaßnahmen zu ergreifen**, indem Sie Softwarestrukturen zur Unterstützung der für die Bedrohungserkennungs und -abwehr verantwortlichen Teams und zur Modellierung von Bedrohungen nutzen.

Erzeugen Sie nicht nur selbst SBOMs, sondern fordern Sie Ihre Softwarelieferanten auf, diese bereitzustellen, und implementieren Sie eine Software, um sie im Rahmen Ihrer Bewertung zu überprüfen. Einige SBOMs können als Ganzes oder komponentenweise digital signiert werden.

Wenn Sie SBOMs einsetzen, vereiteln Sie aktiv die Bemühungen von Cyberkriminellen, die versuchen, durch Verschleierung schädlichen Code in Ihre Software einzuschleusen. Eine Softwarestückliste ist eine der effektivsten Methoden zum Schutz Ihres Unternehmens.

Nr. 5: Automatisieren Sie mit unternehmensweiter Transparenz und Kontrolle

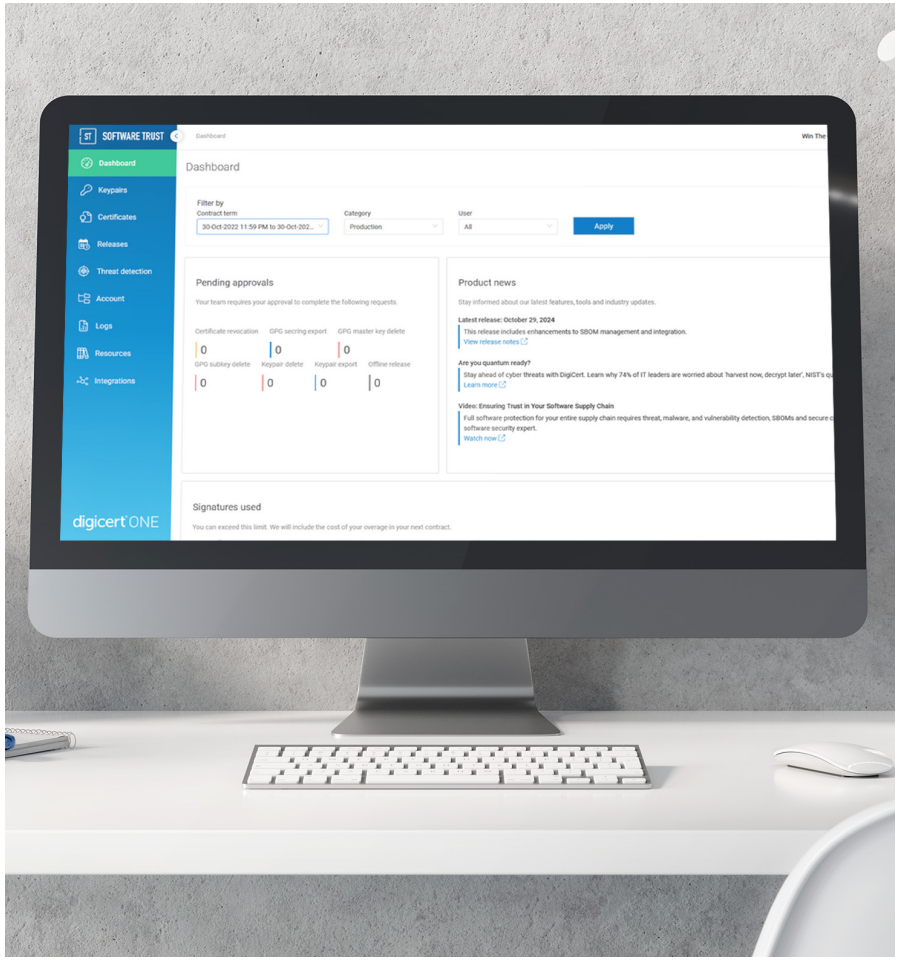
Sie haben vielleicht bemerkt, dass viele Sicherheitsmaßnahmen fast ständig durchgeführt werden müssen. Sicherheitsexperten sind sich darin einig, dass sie nur oft genug durchgeführt werden, wenn ihr Betrieb automatisiert ist.

Im SDLC müssen die Sicherheitsabläufe in den Prozess integriert werden, heutzutage in der Regel durch eine CI/CD-Pipeline. Diese Systeme sind so konzipiert, dass sie Sicherheitsprozesse beinhalten und die Arbeitsabläufe deren Ergebnisse dynamisch berücksichtigen.

Der richtige Ansatz besteht darin, diesen Prozess durch Unternehmensrichtlinien zu steuern und in andere Sicherheitssysteme des Unternehmens zu integrieren. Mit zentraler Transparenz und Kontrolle können Unternehmen die Einhaltung von Vorschriften und die Berichterstattung verbessern und gleichzeitig die allgemeine Sicherheit erhöhen. Darüber hinaus ist es wichtig, diese Prozesse so zu implementieren, dass sie die Arbeitsabläufe der Entwickler unterstützen und rationalisieren, anstatt sie zu belasten, um eine effizientere und sicherere Entwicklungsumgebung zu schaffen.

Agil werden oder scheitern

Die Sicherheit in einer Softwarelieferkette ist ein bewegliches Ziel. Es gibt viele Möglichkeiten für Angriffe, und Sie müssen sich auf die Sicherheit anderer Parteien verlassen. Zu den Best Practices gehört, dass ein möglichst breites Spektrum an Sicherheitsvorkehrungen in alle Phasen des CI/CD-Prozesses integriert wird. Die einzige Möglichkeit, dies zu erreichen, besteht darin, den Prozess zu automatisieren. Ein hoher Automatisierungsgrad beugt nicht nur Vorfällen vor, indem er die regelmäßige Anwendung von Sicherheitsmaßnahmen sicherstellt, sondern hilft Ihnen auch, auf Vorfälle zu reagieren, Prozesse bei Bedarf zu ändern und die für die Compliance-Berichterstattung erforderlichen Informationen zu generieren.



DigiCert Software Trust Manager

DigiCert® Software Trust Manager ist eine Digital-Trust-Lösung zum Schutz der Integrität von Software in der gesamten Lieferkette. Sie ermöglicht, das Risiko von Codemanipulationen zu reduzieren, unternehmensinterne und behördliche Vorgaben umzusetzen und detaillierte Schlüsselnutzungs- und Zugriffskontrollen beim Code Signing anzuwenden.

DigiCert Software Trust Manager minimiert das Risiko von Sicherheitsverletzungen und Malware-Verbreitung während der Entwicklung, Erstellung und Veröffentlichung der Software. Unternehmen sind so vor Angriffen auf die Lieferkette ihrer Software geschützt und können gesetzliche Vorschriften einhalten.

DigiCert Software Trust Manager hilft großen Unternehmen, diese fünf Best Practices einfach umzusetzen.

Entdecken Sie, wie Sie Ihre Software mit Software Trust Manager schützen können