VIER BEST PRACTICES FÜR DEN SCHUTZ VOR ANGRIFFEN AUF SOFTWARELIEFERKETTEN

digicert®

Integrieren Sie sicheres Code Signing, die Erkennung von Bedrohungen und Malware sowie die Generierung von SBOMs.

Angriffe auf die Softwarelieferkette nehmen immer mehr zu. Einem aktuellen Gartner-Bericht zufolge werden 45 % der Unternehmen weltweit bis 2025 einen Angriff auf ihre Softwarelieferkette erleiden. Das entspricht einem Anstieg um das Dreifache seit 2021.

Diese Art Angriff zielt auf große wie kleine Unternehmen ab. Mittlerweile dürften die Softwarelieferkettenangriffe auf SolarWinds, CircleCI und 3CX hinlänglich bekannt sein. Erst kürzlich wurde Micro-Star International (MSI) Opfer eines Cyberangriffs, bei dem private kryptografische Schlüssel gestohlen wurden, die das Unternehmen zum Signieren von BIOS-Updates und Intel Boot Guard nutzte.

Grundlegende Sicherheitsmaßnahmen wie Zero-Trust-Prinzipien bieten Unternehmen zwar eine erste Verteidigungslinie, doch im Kampf gegen Angriffe auf die Softwarelieferkette und den Softwareentwicklungszyklus eines Unternehmens braucht es weitaus mehr.

LESER-ZIELGRUPPE:

Sicherheitsexperten, DevSecOps und Softwareentwickler, die dafür verantwortlich oder daran beteiligt sind, die Softwareentwicklungs-infrastruktur ihres Unternehmens vor Sicherheitsverletzungen, Angriffen und anderen Schwachstellen zu schützen

HERAUSFORDERUNGEN

Für Angriffe auf die Softwarelieferkette nutzen Cyberkriminelle verschiedene Taktiken. Beispielsweise könnten sie mit dem geknackten Passwort eines Softwareentwicklers versuchen, in das Software-Build-System einzudringen, ungeschützte private Code-Signing-Schlüssel stehlen oder missbrauchen, Malware in vorgelagerte Teile gängiger Open-Source-Softwarepakete einschleusen – oder eine Kombination all dieser Taktiken anwenden. Wie die Sunburst-Attacke auf SolarWinds 2019 gezeigt hat, werden die Angriffsmethoden zunehmend anspruchsvoller und komplexer, weshalb auch immer ausgefeiltere Abwehrmethoden erforderlich sind (siehe Abbildung 1).

Abbildung 1: Lieferkettenangriffe können in jeder Phase der Softwareentwicklung auftreten

Angriffe können in jeder Phase des Softwareentwicklungszyklus auftreten. Einzelne Sicherheitstools mögen zwar bestimmte Angriffstypen erkennen, aber bei Weitem nicht alle. Deshalb müssen Unternehmen einen „Shift up"-Ansatz verfolgen – also sicherstellen, dass Software und Anwendungen die Erwartungen der Kunden und des Unternehmens erfüllen – und weitere Sicherheitsmaßnahmen im gesamten Softwareentwicklungszyklus implementieren.

Auch strukturelle und organisatorische Aspekte sowie die Unternehmenskultur spielen eine Rolle: Manche Softwareentwicklungsteams meiden bestimmte Sicherheitstools, weil sie mühsam zu bedienen sind, in automatisierte CI/CD-Build-Pipelines eingreifen oder die Freigabe von Software-Releases verzögern.

Sicherheitsteams wiederum stehen vor einer anderen Art von Herausforderung. Sie geben Sicherheitsstandards vor und müssen deren Einhaltung bei den verschiedenen Softwareteams sicherstellen, die oft im gesamten Unternehmen und nicht selten geografisch verteilt sind. Hinzu kommt, dass diese Teams oft unterschiedliche Programmiersprachen und -umgebungen verwenden, Software auf unterschiedlichen Plattformen bereitstellen und abweichende Build-Infrastrukturen und Softwareprozesse nutzen. Angesichts dieser Heterogenität kann es äußerst schwierig sein, Softwaresicherheitsrichtlinien zu definieren, unternehmensweit durchzusetzen und zu überwachen.

Eine neue Herausforderung für Unternehmen ist zudem noch die Einhaltung diverser behördlicher und branchenspezifischer Vorgaben, die speziell für den Kampf gegen Angriffe auf die Softwarelieferkette erarbeitet wurden, z. B. Vorschriften zur Softwaretransparenz.

BEST PRACTICES FÜR DIE ABSICHERUNG VON SOFTWARELIEFERKETTEN

Verschiedene Vorschriften und Branchenempfehlungen geben nicht nur Best Practices vor, sondern auch Anforderungen, die Unternehmen zur Wahrung der Integrität der Softwarelieferkette erfüllen müssen. Gemäß der „Executive Order" (Durchführungsverordnung) zur Verbesserung der Cybersicherheit in den Vereinigten Staaten und dem Cyberresilienzgesetz der EU müssen Softwareanbieter beispielsweise eine Software-Stückliste (Software Bill of Materials, SBOM) bereitstellen, um die Softwaretransparenz zu gewährleisten. Die „Security Considerations for Code Signing" des NIST (National Institute of Standards and Technology) geben Schritte vor, die für sichere Abläufe beim Code Signing zu befolgen sind.

Diese Lösungsübersicht stellt Ihnen vier Best Practices vor, die Unternehmen bei der Bewältigung dieser Herausforderungen anwenden können, und zeigt auf, wie DigiCert Ihnen helfen kann. Entscheidend ist dabei, dass diese Best Practices über den gesamten Softwareentwicklungslebenszyklus (SDLC) angewandt werden und nicht nur während einer einzelnen Phase (siehe Abbildung 2).

Abbildung 2: Vertrauen in Software: „Shift left“, „Shift right“ … UND „SHIFT UP“

Zur Bekämpfung von Angriffen auf die Softwarelieferkette ist ein vielseitiger, integrierter und moderner Ansatz erforderlich, der sich auf mehrere Aspekte des Softwareentwicklungslebenszyklus konzentriert.

Nr. 1: Sicheres Code Signing im gesamten Unternehmen

Code Signing ist eine grundlegende Sicherheitsmaßnahme, die seit über 30 Jahren angewandt wird. Dabei wird ein Softwarepaket (ausführbare Datei, Programmbibliothek, Anwendung usw.) mittels Kryptografie digital signiert, um nachzuweisen, dass es authentisch ist (z. B. dass eine von DigiCert signierte Software wirklich von DigiCert stammt) und dass es nicht durch Dritte manipuliert wurde. Viele Betriebssysteme wie iOS, macOS oder Windows überprüfen die digitale Signatur einer Software, bevor das Paket installiert oder ausgeführt wird.

Das Signieren von Software ist so effektiv, dass Angreifer es auf Code-Signing-Systeme abgesehen haben, um die Code-Signing-Zertifikate und privaten Schlüssel eines Unternehmens zu stehlen, zu manipulieren oder zu missbrauchen.

Der erste – und offensichtliche – Schritt zur Absicherung des Code Signing ist, alle privaten Code-Signing-Schlüssel sicher zu verwahren. Erst kürzlich veröffentlichte das Certificate Authority/Browser Forum (CA/B Forum) die Vorgabe, dass öffentlich vertrauenswürdige private Code-Signing-Schlüssel auf Hardware gespeichert werden müssen, die gemäß FIPS 140 Level 2, Common Criteria EAL 4+ zertifiziert ist (z. B. auf einem Hardware-Sicherheitsmodul, HSM).

Die Verwendung eines sicheren Speichermediums ist zwar ein notwendiger erster Schritt, genügt aber nicht, um das Code Signing im gesamten Unternehmen abzusichern. Grund hierfür ist, dass Angreifer mit einem geknackten Benutzernamen und Passwort Zugriff auf den sicheren Speicher erlangen könnten.

Deshalb empfehlen wir die Verwendung eines Code-Signing-Systems, das den Anforderungen großer Unternehmen gerecht wird und eine Reihe zusätzlicher Sicherheitsmaßnahmen bietet, darunter:

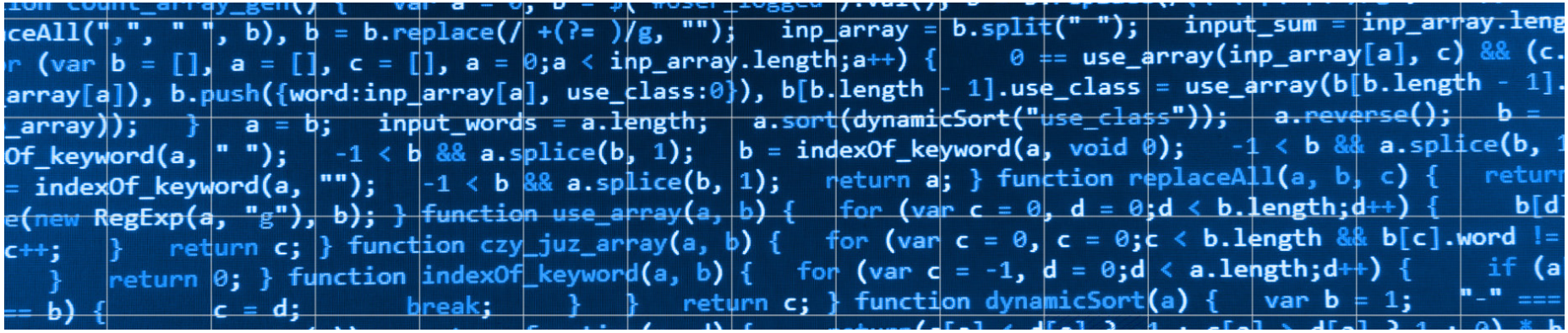
- Automatisierte Verwaltung von Code-Signing-Zertifikaten, einschließlich Ausstellung und Widerruf
- Ein Genehmigungsverfahren, das nicht nur festlegt, wer auf einen Code-Signing-Schlüssel zugreifen darf, sondern auch, wer den Zugriff genehmigen muss und unter welchen Umständen auf den Schlüssel zugegriffen werden darf (z. B. Tageszeit, Build-Maschine usw.)
- Ein rollenbasiertes System, das vorgibt, welche Benutzer auf bestimmte Code-Signing-Zertifikate und Schlüssel zum Signieren zugreifen dürfen, wer die Signiervorgänge genehmigen und wer sie überwachen darf
- Unwiderlegbare Aufzeichnung aller Code-Signing-Aktivitäten, auf die im Falle eines manipulierten Zertifikats verwiesen werden kann
- Ein zentrales Code-Signing-System, das eine große Menge von Code-Signing-Vorgängen und viele verschiedene Plattformen und Tools für die Softwarebereitstellung und -entwicklung unterstützt
- Ein richtlinienbasierter Ansatz für Prüfmechanismen, die erfüllt sein müssen, bevor Software veröffentlicht 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

Nr. 2: Umfassende Bedrohungserkennung

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 gibt eine bisher unerkannte Schwachstelle.

Verschiedene Sicherheitstools auf dem Markt zielen auf einige dieser Probleme ab, z. B. Tools für statische Softwaretests (Static Application Security Testing, SAST). Diese Tools nehmen jedoch oft nur einen einzigen Aspekt unter die Lupe (z. B. Sicherheitslücken in Open-Source-Software).

Stattdessen braucht es einen umfassenden Mechanismus, der die endgültigen Binärdateien auf Bedrohungen, Schwachstellen sowie das Vorhandensein eingebetteter Secrets und Malware überprüft. Dieser Scan sollte immer erfolgen, egal, aus welcher Quelle die Softwarekomponenten stammen, sei es Open-Source-Code, vom Unternehmen selbst entwickelter Code oder Code aus kommerziellen Programmbibliotheken von Drittanbietern.



Nr. 3: Softwaretransparenz

Wie bereits erwähnt müssen Softwareentwickler nun laut Behörden und gesetzlichen Vorgaben häufiger den Inhalt ihrer Software offenlegen, ähnlich der Angabe von Inhaltsstoffen bei Lebensmitteln. Diese Nachweisdokumente nennen sich SBOMs (Software-Stücklisten) und führen alle Komponenten auf, die in einer Software enthalten sind, unabhängig von der Quelle, aus der die Softwarekomponenten stammen (Open Source, externe Programmbibliothek usw.). Und wie bei der Inhaltsstoffangabe geben SBOMs nicht nur Auskunft über die „Zutaten" einer Software, sondern auch über deren Qualität, also die verwendeten Versionen, bekannte Schwachstellen und Ähnliches.

Selbst wenn eine solche Software-Stückliste nicht vorgeschrieben ist, empfiehlt es sich für Unternehmen, SBOMs zu generieren, um:

- **die Angriffsfläche ihrer Software zu bewerten und zu priorisieren**: Sie können sie auf Programmbibliotheken oder Komponenten überprüfen, die besonders anfällig für Angriffe sind bzw. Sicherheitslücken haben, Softwareabhängigkeiten analysieren und herausfinden, ob vorgeschriebene Sicherheitspatches fehlen.
- **nach veralteten Komponenten zu suchen**, wie fehlenden Fehlerbehebungen und unsicheren Code-Signing-Verfahren.
- **proaktive Verteidigungsmechanismen zu ergreifen**, indem sie die Sicherheits- und Incident-Response-Teams mit Informationen zur Softwarestruktur bei der Bedrohungserkennung und -abwehr unterstützen und Bedrohungsmodelle erstellen.

Nr. 4: Shift up: Kontrolle und Transparenz im gesamten Unternehmen und Softwareentwicklungslebenszyklus

Viele Sicherheitsmechanismen prüfen und schützen nur einen bestimmten Teil des Softwareentwicklungslebenszyklus (SDLC). Wir kennen die Ansätze, die Software entweder bereits früh im DevOps-Prozess („Shift left") oder nachlaufend („Shift right") prüfen. Das ist aber so, als würde man sich nur einzelne Bäume in einem ganzen Wald anschauen. Mit der zunehmenden Komplexität von Cyberangriffen müssen Unternehmen eine andere, ganzheitliche Perspektive einnehmen („Shift up"). Dazu müssen sie:

- den gesamten SDLC unter die Lupe nehmen und die Software als Ganzes betrachten.
- Transparenz der Sicherheitsrichtlinien, -maßnahmen und -ergebnisse im gesamten Unternehmen gewährleisten.
- in der Lage sein, Sicherheitsrichtlinien im gesamten Unternehmen zu definieren und durchzusetzen.
- eine Bedrohungs- und Schwachstellenanalyse der fertigen ausführbaren Softwaredateien durchführen, direkt bevor sie signiert und veröffentlicht werden.

Um dies effektiv zu gewährleisten – und um Akzeptanz bei Softwareentwicklungsteams zu finden –, müssen sich Sicherheitstools nahtlos in bestehende Softwareprozesse integrieren lassen, ohne den Software-Veröffentlichungsprozess auszubremsen. Sie müssen einfach bzw. transparent zu bedienen sein, damit Softwareentwickler sie auch wirklich nutzen.

DIGICERT SOFTWARE TRUST MANAGER

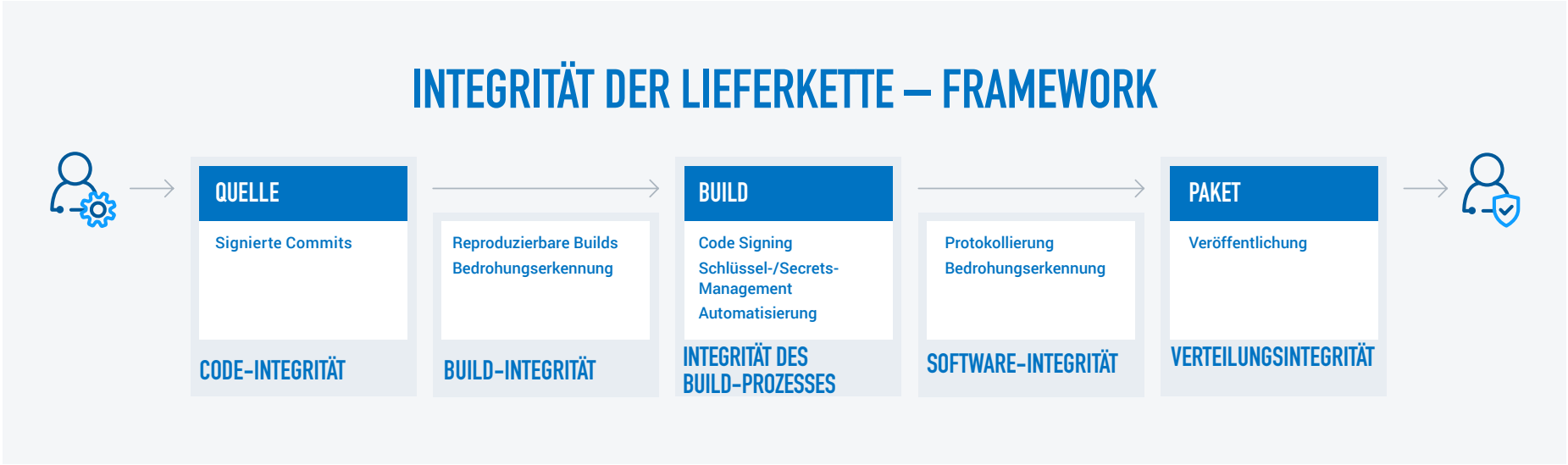
DigiCert® Software Trust Manager ist eine Digital-Trust-Lösung zum Schutz der Integrität von Software in der gesamten Lieferkette. Sie ermöglicht, das Risiko von Codemanipulationen zu reduzieren, unternehmensinterne und behördliche Vorgaben umzusetzen und detaillierte Schlüsselnutzungs- und Zugriffskontrollen 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 dabei, diese vier Best Practices einfach umzusetzen.

Weitere Informationen erhalten Sie per E-Mail an pki_info@digicert.com.

Abbildung 3: Software Trust Manager



DigiCert Software Trust Manager bietet Schutz über den gesamten Softwareentwicklungszyklus mit sicherem Code Signing, umfassender Bedrohungserkennung sowie unternehmensweiter Kontrolle und Transparenz der Software und des Entwicklungszyklus.