SOLARWINDS MARKIERT EINEN WENDEPUNKT

digicert®

Inhalt

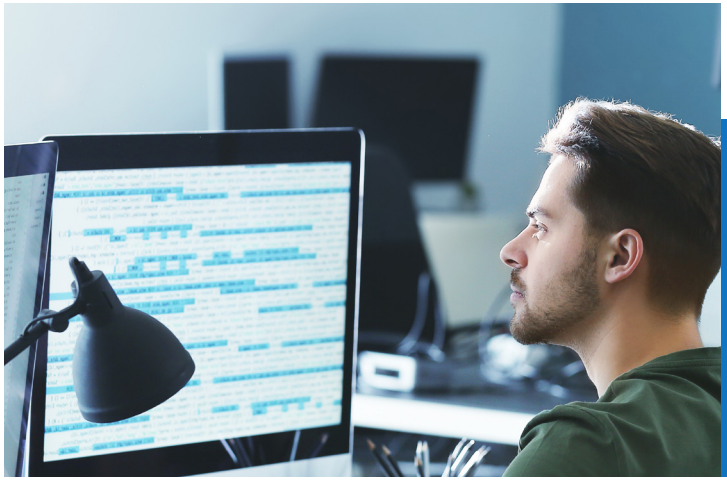
Der große Lieferketten-Coup

Am Sonntag, dem 13. Dezember 2020, bestätigten die USA, dass Hacker in mehrere Computernetzwerke der Bundesregierung eingedrungen waren. In den folgenden Wochen wurde festgestellt, dass dieser raffinierte Angriff mit höchster Wahrscheinlichkeit das Werk russischer Agenten war, die über Monate hinweg unbemerkt sensible Daten sammeln konnten. Im Zuge der Ermittlungen offenbarte sich das ganze Ausmaß dieses perfiden Coups. Das war kein bloßer erfolgreicher Spionageakt, sondern vielmehr einer der größten Malware-Angriffe seit Jahren.[1]

Der Angriff ließ sich zu einem Softwaresystem namens Orion zurückverfolgen. Dabei handelt es sich um ein IT-Verwaltungsprogramm des texanischen Cybersicherheitsunternehmens SolarWinds. So wurde SolarWinds innerhalb kurzer Zeit zum Synonym für Datenlecks. Aber SolarWinds ist nicht das erste Unternehmen, das es erwischte, und die Schlagzeilen erzählen nicht die ganze Geschichte.

In Wahrheit begann diese Manipulation weit vor Beginn des Angriffs – mit dem Einschleusen von Schadcode in die DNA von Orion. Die Malware, eingebettet in den CI/CD-Build, wurde so zur tickenden Zeitbombe in der Software selbst. Die Hacker hatten Geduld. Ihre Bombe hatte eine extrem lange Lunte. Sie warteten einfach den richtigen Zeitpunkt ab, an dem sie den größten Schaden verursachen würden. In den folgenden Monaten wurde Orion überall auf der Welt installiert – in Regierungsbehörden, Großkonzernen und auch anderen Softwareunternehmen.

Und dann, eines Tages, irgendwann nach März 2020, platzte die Bombe. Bis diese Infiltration bekannt wurde, war es längst zu spät – nicht nur, weil die Malware funktionierte, sondern auch weil Orion überall präsent war. Das Isolieren einer einzigen Schwachstelle oder eines Servers allein brachte in diesem Fall nichts. Es ging nicht um einen einzelnen Patch. Der Angriff war so riesig, dass immer noch nicht klar ist, wie viele Systeme, Unternehmen und Organisationen tatsächlich betroffen waren. Möglicherweise wird dieses Geheimnis nie gelüftet.



[1] https://www.reuters.com/article/us-cyber-solarwinds-microsoft/solarwinds-hack-was-largest-and-most-sophisticated-attack-ever-microsoft-president-idUSKBN2AF03R

SolarWinds – die Zeit davor und danach

Was den SolarWinds-Angriff so bemerkenswert macht, ist sein Ausmaß. Die infizierte Orion-Software wurde an über 33.000 Kunden verkauft. Sunburst, so der treffende Name des Schadcodes, fand in sage und schreibe 18.000 Organisationen Einlass. Über Monate verbarg sich dieses Trojanische Pferd in vielen tausenden ahnungslosen Unternehmen und Behörden hinter den Firewalls der befallenen Netzwerke.

Die Forensik ermittelte im Anschluss an die Katastrophe, dass es Sunburst in 425 der US-amerikanischen Fortune-500-Unternehmen, die 10 größten Telekommunikationsunternehmen, fünf der wichtigsten Wirtschaftsprüfungsgesellschaften und Microsoft selbst geschafft hatte. Sunburst ermöglichte ausländischen Akteuren den Zugang zum Weißen Haus, Finanzministerium, Pentagon, Außenministerium, Justizministerium und Heimatschutzministerium, dazu zur NSA, zur Nationalen Behörde für Nukleare Sicherheit und zu allen fünf Bereichen des US-Militärs. Zu den Opfern außerhalb der USA zählten unter anderem die NATO, die Regierung des Vereinigten Königreichs und das Europaparlament. Ob öffentliche, private, große oder kleine Einrichtungen: Sunburst durchdrang die Schutzwälle, als Vehikel diente dabei vermeintlich sichere Software, die signiert war.

Ein Jahrzehnt zuvor meinten Fachleute für Cybersicherheit noch, ein Lieferkettenangriff sei extrem unwahrscheinlich. Er würde nicht zum Schwachstellenprofil passen, auf das es Hacker absehen. Und da niemand – ob Einzelperson oder Unternehmen – jedes einzelne Element im Build absichern konnte, sei der Schutz der Lieferkette unergiebig. Diese Idee setzte sich dermaßen durch, dass Stacy Simpson von SAFECode zusammen mit einer Gruppe führender Köpfe aus der Branche eine Warnung veröffentliche, in der empfohlen wurde, dass DevOps-Fachleute diese Standpunkte ignorieren und Best Practices für Code Signing implementieren sollten, um Eintrittspunkte in der CI/CD-Lieferkette zu schließen.[2]

Die meisten Leute glaubten trotzdem nach wie vor, dass Angriffe auf die Lieferkette entweder nicht möglich oder nicht vermeidbar seien. Doch nach dem SolarWinds-Coup wurden sie eines Besseren belehrt.

Forensik einer vermeidbaren Katastrophe

Die Art der Schwachstelle, die zum SolarWinds-Angriff führte, ist nicht neu. Bereits seit Erfindung der CI/CD-Pipeline ist infizierter Code für Sicherheitsfachleute ein Problem. Doch vor dem SolarWinds-Angriff gab es nie Vorfälle diesen Ausmaßes, die so verwerflich, öffentlich und klar vermeidbar waren.

[2] http://safecode.org/publication/SAFECode_Software_Integrity_Controls0610.pdf

SolarWinds brachte Orion als legitimes, signiertes Softwarepaket heraus – das ist gängige Praxis. Durch das Signieren der Software versicherte SolarWinds seinen Kunden gegenüber, sie nach Build und Freigabe über mögliche Manipulationen oder Malware zu informieren. Als nun signierte Software einen Verstoß ermöglichte, war die Digitalbranche gezwungen, innezuhalten und den gesamten Lieferkettenprozess für Software zu überdenken. Wie können wir Kunden und Endbenutzer schützen, wenn Code bereits vor dem Signieren Malware enthält?

Code Signing selbst ist immer noch eine bewährte Technik, aber die SolarWinds-Schwachstelle zeigte, dass Code Signing allein nicht genügt, um die Lieferkette zu schützen. Ohne das Scannen von Quellcode und externen Bibliotheken, ordentlicher Personalüberprüfung und einer sicheren Build-Umgebung – zusätzlich zum Code Signing – ist Software gegenüber Angriffen ungeschützt. Das ist so wie ein hochwertiges Sicherheitssystem im Haus, das Sie aber immer nur eine Stunde aktivieren.

Nach SolarWinds identifizierte man neue Angriffsvektoren für die Lieferkette. Außerdem wurde klar, dass das Signieren von Software nur der letzte Schritt im DevOps-Zyklus ist, aber nicht der einzige. Der komplette Prozess muss abgesichert, kontrolliert und validiert werden. Ohne umfassende Sicherheit kommt es nie zu einem geschlossenen DevOps-Kreislauf.

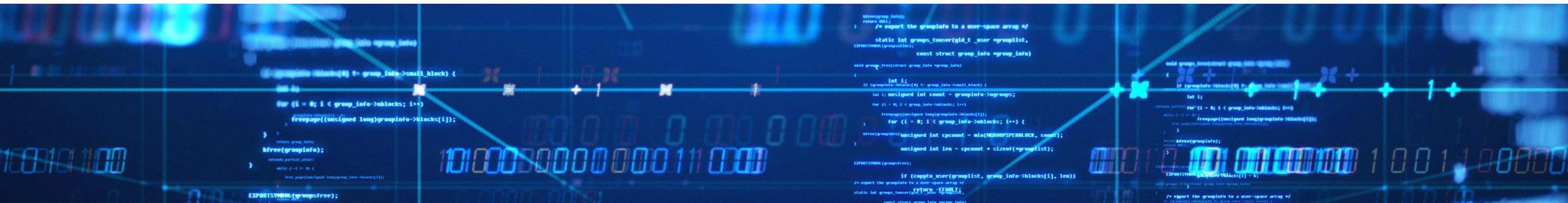
Signieren bei allen Schritten, nicht nur am Schluss

Beim Signieren von Software wird sie ab diesem Zeitpunkt gegen Manipulation abgesichert. Verschlüsselung ist dabei aber längst nicht alles. Code Signing prüft die Identität des Herausgebers und die Integrität des Codes nach dem Download. So wissen Benutzer, dass die Software nicht verändert wurde. Da signierte Software als vertrauenswürdig gilt, müssen DevOps-Teams bei Software-Releases nachweisen können, dass sie keine Malware enthält. Signierte Software mit Malware umgeht sämtliche Prüfungen öffentlicher und privater Verteilsysteme. Im Wesentlichen sorgt das Signieren von Software dafür, dass weder an den Security-Gateways der restlichen Lieferkette noch am benutzerseitigen Zugriffspunkt nach Problemen Ausschau gehalten wird – und das ist besonders bei Software mit manipuliertem Code gefährlich.

Absolute Vertrauenswürdigkeit erlangen Sie nur mit durchgängigem Schutz der Software-Integrität im gesamten CI/CD-Prozess. Es reicht nicht aus, Software am Ende des Entwicklungszyklus durch Ihr Team zu signieren. Code, der von anderen Teams in der Lieferkette geschrieben wird, einschließlich offener Quellcode und Code aus externen Bibliotheken, muss auf Malware und andere Anomalien geprüft werden. Zum Erstellen von Software darf ausschließlich sauberer Code verwendet werden, der dann mit den richtigen Signaturschlüsseln signiert wird, die ordnungsgemäß herausgegeben, verwaltet und gespeichert werden. Der Zugriff auf diese Signaturschlüssel muss auf die Benutzer oder automatischen Build-Server beschränkt werden, die zur Nutzung von Schlüsseln für das Signieren entsprechend autorisiert sind.

Da das eine vermeintliche Mammutaufgabe ist, behelfen sich Entwickler oft mit unzureichenden Sicherheitspraktiken oder Abkürzungen, damit der Build vorangeht. Es ist hinlänglich bekannt, dass DevOps von Tempo und Wirksamkeit bestimmt wird. Lästige Schritte, die Deadlines platzen lassen, können den CI/CD-Prozess zum Erliegen bringen, und Lösungen, die nicht agil sind, also Verzögerungen und Unterbrechungen verursachen, sind keine Lösungen.

In vielen Fällen lassen sich diese Signaturprozesse automatisieren, um menschliche Eingriffe zu verringern. Ihre Entwickler erhalten auf diese Weise mehr Zeit, um sich auf die Erstellung von Code zu konzentrieren – ohne Abstriche an die Sicherheit.



Best Practices und mangelnde Sicherheit

Das Signieren von Software ist für sich selbst genommen wichtig, aber die Verwaltung der entsprechenden Richtlinien und Praktiken kann dabei schnell ins Hintertreffen geraten. Und dies ist der Punkt, an dem Sicherheitslücken entstehen, die Hacker ausnutzen. In puncto Softwaresicherheit gibt es nur zwei Möglichkeiten: Sie können entweder Best Practices einsetzen oder Ihre Lieferkette schutzlos Angriffen aussetzen.

Wenn Sie Best Practices für Code Signing implementieren, verhindern Sie, dass das Signieren zum schwächsten Glied in Ihrer Lieferkette wird. Richtig umgesetzt verhindert Code Signing zudem die Einschleusung von Malware.

1. Schützen Sie Signatur- und private Schlüssel und steuern Sie den Zugriff darauf

Private Schlüssel können dupliziert bzw. an mehreren Speicherorten archiviert werden und physische Schlüssel wie USB-Tokens sind anfällig für Verlust oder Diebstahl. Frustriertes oder nachlässiges Personal achtet mitunter nicht auf den Verbleib der Schlüssel oder lässt sie mitgehen.

Wenn Sie private Schlüssel vollumfänglich absichern möchten, haben Sie zwei Möglichkeiten: Entweder bewahren Sie sie in einem ordentlich verwalteten HSM (Hardware-Sicherheitsmodul) auf oder Sie lassen die Schlüssel durch Managed PKI verwalten.

2. Setzen Sie Multifaktor-Authentifizierung und FIPS-konforme Speicherung für die Ausstellung öffentlicher Zertifikate durch

Ein Kennwort kann vergessen oder geknackt werden, Schlüssel können preisgegeben oder gestohlen werden und verloren gehen, und physische USB-Tokens können verlegt oder missbraucht werden.

Zum Schutz vor Benutzerfehlern oder unlauteren Eingriffen sollten Sie daher strenge Zugangskontrollen einrichten. Bei der Multifaktor-Authentifizierung (MFA) sind zwei Arten von Legitimierung seitens des Benutzers erforderlich, um dessen Identität zu bestätigen. HSMs bieten eine zuverlässige Authentifizierung und sind vor physischer Manipulation gefeit. Dadurch erhalten nur autorisierte Benutzer Zugriff auf geschützte Schlüssel.

3. Volle Transparenz

Ohne durchgängiges Auditing und Verantwortlichkeit der Benutzer lassen sich Missbrauch von Schlüsseln oder Anomalien beim Signieren nur schwer nachverfolgen, wenn überhaupt.

Sowohl Schlüssel als auch Benutzer müssen sich vor dem Signieren authentifizieren. Dieser Vorgang muss in jedem Schritt des CI/CD-Prozesses vorhanden sein. Die Signaturen müssen nachverfolgt werden und überprüfbar sein, damit Benutzer die Verantwortung übernehmen und Sie wissen, wer was signiert hat. Somit kann die Verwendung von Signaturschlüsseln von autorisierten Benutzern zu unzulässigen Zeiten IT-Sicherheitsteams dabei helfen, Probleme aufzudecken, bevor die Software zur nächsten Schnittstelle gelangt.

4. Kontrollieren Sie die Sicherheit auf Account-Ebene

Wenn Schlüssel ohne Steuerelemente für die Zugriffsberechtigung verwendet werden, ist es schwer bis unmöglich, Schlüssel und Signiervorgänge zu schützen und zu überwachen.

Schlüssel sollten anhand der Berechtigung und Rolle zugewiesen sowie nach Produkt, Projekt oder Team getrennt werden. Außerdem müssen Administratoren in der Lage sein, private Schlüssel und Zertifikate zu generieren und den Teams Schlüssel zuzuweisen. Benutzer dürfen nur die verwenden, die Signier-Workflows zugewiesen sind. Maschinelle Komponenten wie Build-Server sollten zu Automatisierungszwecken über zugewiesene Schlüssel verfügen.

5. Kontrollieren Sie die Sicherheit auf Organisationsebene

Richtlinien für das Unternehmen sind genauso wichtig wie die Rollen und Berechtigungen einzelner Mitarbeiter. Ohne eine ganzheitliche Richtlinien- und Kontrollstruktur können einzelne Sicherheitsverfahren abweichen und Schwachstellen entstehen.

Erstellen Sie daher interne Sicherheitsrichtlinien auf ganzer Linie und sorgen Sie für deren Durchsetzung in den relevanten Bereichen – vom gesamten Unternehmen bis hin zur Produktentwicklung. Diese Richtlinien sollten kryptografische Algorithmen oder Schlüsselgrößen, Zertifikatsgültigkeitszeiträume und Zertifikatstypen oder Genehmigungsabläufe umfassen.

6. Initiieren Sie Schlüsselrotation

Wenn mehrere Codestücke mit ein und demselben Schlüssel signiert werden, gefährdet ein kompromittiertes Stück sämtliche anderen.

Wenn Sie mehrere Schlüssel schnell und einfach ausstellen und rotieren können, vermeiden Sie, dass sich ein isolierter Eingriff zu einem weitgreifenden Angriff auswächst.

7. Legen Sie Richtlinien für die Zuweisung verschiedener Schlüsseltypen zu bestimmten Projekten fest

Ohne strenge Kontrolle über verschiedene Schlüsseltypen verlieren Unternehmen schnell den Überblick über den Zugriff auf und die Nutzung von Schlüsseln – das öffnet unbefugten bzw. unangemessenen Aktivitäten Tür und Tor.

Nutzen Sie mehrfach bestimmte Signaturschlüssel für dedizierte Produkte oder Projekte, die spezifischen Benutzern und Teams vorbehalten sind. Nutzen Sie nach Möglichkeit in Echtzeit erstellte private Schlüssel und Zertifikate, damit jede Freigabe mit einem eindeutigen Schlüssel und Zertifikat signiert wird.

8. Unsignierte Software muss immer in Ihrer Umgebung bleiben

Die Übertragung vollständiger Dateien zum Signieren ist nicht nur langsam und ressourcenintensiv, sondern birgt auch das Risiko, dass diese Dateien bei der Übertragung abgefangen oder manipuliert werden.

Die Lösung: Hash Signing. Dabei wird nur der Hash zum Signieren in die Cloud hochgeladen – das macht den Prozess fast so schnell wie lokales Signieren. Der Code selbst bleibt auf Ihren internen Servern. Ihr geistiges Eigentum kann also nicht gestohlen oder manipuliert werden.

9. Prüfen Sie regelmäßig Ihr geistiges Eigentum

Das klingt wie ein einfacher Schritt, aber Signaturen vermitteln Vertrauen. Benutzer gehen davon aus, dass signierter Code sicher ist. Wenn sich bereits ein Virus oder Malware darin eingenistet hat, sendet der Entwickler vermeintlich vertrauenswürdigen, aber tatsächlich infizierten, Code an den Kunden.

Scannen Sie daher Quellcode, Code aus externen Bibliotheken sowie kompilierten Code. Suchen Sie nach bösartigen Injektionen und signieren Sie nur, wenn Sie sich vergewissert haben, das alles in der Software sauber ist.

10. Sorgen Sie für Reproduzierbarkeit zur Prüfung der von Ihnen signierten Inhalte

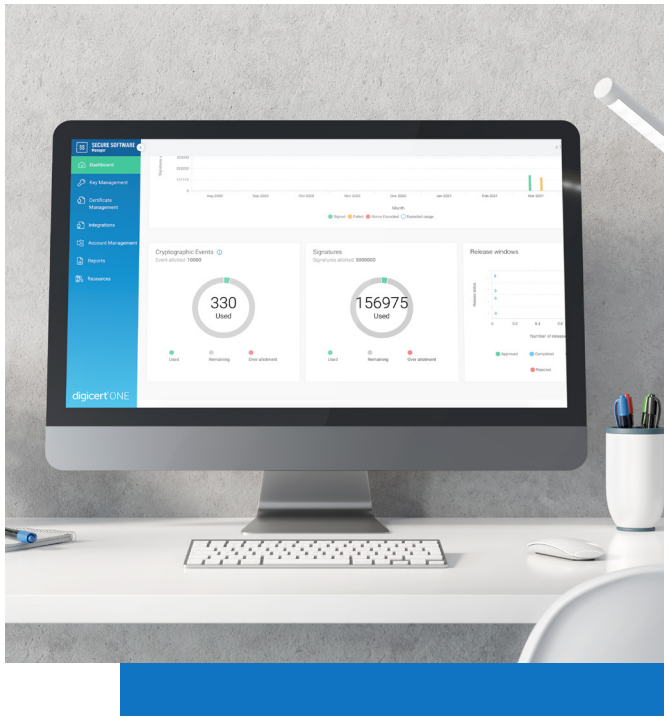
Wenn Sie Builds nicht gegen eine entsprechende Liste abgleichen können, können Sie nicht sicher sein, ob die Binärdateien mit den Eintragungen identisch sind und von einem Benutzer-Quorum reproduziert werden können. Dadurch steigt das Risiko, dass Malware unerkannt eingeschleust wird.

Vergleichen Sie die Hash-Werte der Binärdateien, die Sie zum Signieren erhalten haben, vorher mit den Listeneinträgen. Dieser Abgleich ist erforderlich, um sicherzustellen, dass der Production Build das erwartete Ergebnis aus Ihren Test- und QS-Läufen generiert.

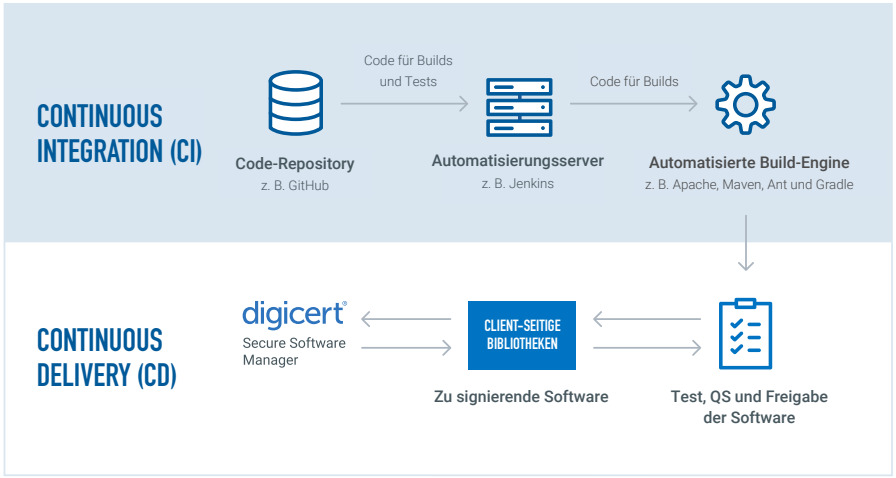
Kontinuierliche Signaturen machen alles schneller

DigiCert Secure Software Manager sorgt für agiles DevOps. Secure Software Manager nutzt automatisierte Prozesse und ermöglicht so kontinuierliche Signaturen für ein Höchstmaß an Vertrauenswürdigkeit mit durchgängiger Sicherheit – ohne zusätzliche Schritte oder auch nur eine Minute Zeitverlust.

Signieren Sie Code, Software und Binärdateien schnell, bequem und in großem Maßstab. Erstellen Sie reproduzierbare Build-Prozesse, die Listenparameter automatisieren, damit eingeschleuste Malware erkannt wird, bevor der beschädigte Code freigegeben wird. Sichere Schlüsselverwaltung, Steuerung des Zugriffs anhand von Berechtigungen und Reporting- und Tracking-Funktionen sorgen für Transparenz und Rechenschaftsnachweise. Mit DigiCert Secure Software Manager haben Sie eine einzige Lösung für die Implementierung und Verwaltung von Best Practices für Code Signing in jedem Schritt Ihres CI/CD-Prozesses.



Automatisches Code Signing in CI/CD-Prozessen mit DigiCert® Secure Software Manager



Wir sind gewarnt und ergreifen jetzt entsprechende Maßnahmen

Als Stacy Simpson den SAFECode-Bericht mit der Empfehlung veröffentlichte, die CI/CD-Lieferkette abzusichern, konnte sie noch nicht wissen, dass sie keinen potenziellen, sondern einen sehr realistischen Angriff beschrieb. Einen, der zu den wohl bedeutsamsten in der Geschichte der IT zählt. 2010 galt ein Angriff auf die Lieferkette als unwahrscheinlicher Eindringversuch. Zehn Jahre später wurde der SolarWinds-Coup zum prominenten Beispiel für einen extrem zerstörerischen, erfolgreichen Hack und Diebstahl.

Ironischerweise war die Lösung für diese Bedrohung der CI/CD-Lieferkette bereits 2010 bekannt und im SAFECode-Bericht veröffentlicht worden. Da einzelne DevOps-Teams und Unternehmen nicht einmal darauf hoffen können, sämtliche Elemente der Lieferkette zu kontrollieren, können entsprechende Cyberangriffe nur mit durchgehender Überwachung der Integrität des Codes während der Entwicklung unterbunden werden. Wenn alle am Build-Prozess Beteiligten ihren Code scannen und signieren, wird die gesamte Lieferkette vor Schwachstellen geschützt.

2010 wäre das Ausmaß dieser Aufgabe für viele vielleicht noch untragbar gewesen. Denn wenn Sicherheitsmaßnahmen den CI/CD-Prozess verzögern oder unterbrechen, ist die Sicherheit selbst genauso gefährlich für DevOps wie eine potenzielle Cyberbedrohung.

Heutzutage lässt sich Code Signing aber zum Glück automatisieren. Automatisierung gibt DevOps-Teams die Tools an die Hand, die sie zur massiven Eindämmung der Angriffsoberfläche im gesamten CI/CD-Prozess benötigen. Mit Automatisierung können sie kontinuierliche Signaturen implementieren, damit Code, Software und Anwendungen durch durchgängige Integrität und Verschlüsselung mit minimalem menschlichem Eingriff geschützt werden.

Aufgrund des erfolgreichen SolarWinds-Angriffs können wir nur annehmen, dass andere Hacker bereits eigene Angriffe auf die Lieferkette in die Wege geleitet haben. Zur Abwehr dieser Bedrohung gehört, dass wir branchenweit Best Practices für Code Signing etablieren. Wenn alle in der Lieferkette ihren Code überwachen und absichern, werden alle Bindeglieder bei der Entwicklung und Übergabe an den nächsten Prozessschritt geschützt.

Fazit: SolarWinds ist kein bloßer sensationeller Akt der Cyberspionage, sondern markiert vielmehr einen Wendepunkt in der Geschichte der Code- und Softwareentwicklung. Die DevOps-Branche kann es sich nicht leisten, zu behaupten, wir seien nicht gewarnt worden. Das potenzielle Risiko war bekannt. Das Maß an Zerstörung, das tatsächlich eintrat, hat nun Geschichte geschrieben. Die Tools zum Schutz vor dieser Gefahr – ohne Verzögerungen oder Unterbrechungen – stehen uns zur Verfügung. Jetzt ist es an der Zeit, diese Tools flächendeckend einzusetzen und Best Practices für Code Signing zu etablieren, damit die DevOps-Lieferkette von Anfang bis Ende geschützt ist.