digicert®

Praktischer Leitfaden zur Code-Signing-Richtlinie: Bessere Sicherheit, Kontrolle und Governance

RICHTLINIEN-LEITFADEN

Praktischer Leitfaden zur Code-Signing-Richtlinie: Bessere Sicherheit, Kontrolle und Governance

Einleitung

Was bezweckt Code-Signing?

Code-Signing fügt Software ein manipuliationssicheres Siegel hinzu. So können die Nutzer von Software – Browser, App-Stores, Betriebssysteme usw. – die Identität des Softwareerstellers überprüfen und sicher sein, dass die Software seit ihrer Signierung und Veröffentlichung nicht verändert wurde.

Code-Signing basiert auf einer Public Key Infrastructure (PKI) und ist ein unwiderruflicher Bescheinigungsprozess zum Nachweis der Rechtsmäßigkeit und Integrität der Software. Mit anderen Worten: Code-Signing beweist zweifelsfrei, woher die Software stammt und dass es sich um Originalcode handelt.

Um vertrauenswürdige Signaturen zu gewährleisten, müssen Unternehmen ihren Softwareentwicklungszyklus (SDLC) und ihre Softwarelieferkette (SSC) sichern. Indem Sie Regeln und Praktiken in einer Code-Signing-Richtlinie erfassen, schaffen Sie eine Grundlage für einheitliche Codesicherheit in allen Entwicklungsteams.

Die Bedeutung einer zuverlässigen, dokumentierten Code-Signing-Richtlinie

Die Sicherheitsvorschriften und -standards, die für den SDLC und die Software-Infrastruktur gelten, unterliegen aufgrund von Angriffen auf die Softwarelieferkette und den SDLC selbst einem raschen Wandel. Außerdem kommen jeden Tag neue Vorschriften hinzu.

Unabhängig davon, ob Sie sich an den Mindestanforderungen des CA/B Forum, der NIS2-Richtlinie oder den PCI DSS-Branchen-standards orientieren, sollten Sie eine einzige, zusammenhängende Richtlinie erstellen, die alle für Ihr Unternehmen geltenden Vorschriften berücksichtigt. Denken Sie daran, dass nicht nur der Standort Ihres Unternehmens ausschlaggebend für die geltenden Vorschriften ist. Ihre Software muss auch den Vorschriften der Länder entsprechen, in denen sie vertrieben und verwendet wird.

Das Signieren von Code ist ein entscheidender Schritt bei Entwicklung und Schutz von Software, der einen großen Einfluss auf die Sicherheits- und DevOps-Teams hat. Sicherheits- und DevSecOps-Prozesse bestimmen, welche Sicherheitsmaßnahmen durchgeführt werden sollten und warum sie erforderlich sind. DevOps-Teams müssen dieses Wissen in konkrete Schritte, Systemeinstellungen und Integrationen umsetzen, um diese Ziele zu erreichen. Häufig werden diese Sicherheitsmaßnahmen als störend bei der Softwarebereitstellung empfunden. Die Einbettung der in den Richtlinien verankerten Grundsätze in die Signieranwendungen und CI/CD-Plattformen ermöglicht es den Entwicklern jedoch, Code korrekt und reibungslos zu signieren, was die Geschwindigkeit und Effizienz des Release-Prozesses erhöht.

Dieser Leitfaden kann dabei helfen, diese Gruppen zusammenzubringen und geeignete Sicherheitsmaßnahmen auf der Grundlage des Softwareproduktrisikos, der Software-entwicklungsumgebungen und der Sicherheitsreife auszuarbeiten.

Zweck und Umfang dieses Leitfadens

Teil 1: Zusammentragen von Informationen und Fachwissen
Code-Signing betrifft Stakeholder in verschiedenen Teams und Abteilungen. Vorbereitende Schritte und das Erfassen von Informationen tragen dazu bei, dass die Ausarbeitung von Richtlinien schneller vorankommt.

Teil 2: Allgemeine Richtlinienkomponenten
Machen Sie sich mit den Zielen der einzelnen Richtlinienabschnitte sowie mit den Praktiken und Verfahren vertraut, die von führenden Experten für Softwaresicherheit und DevOps als wichtig erachtet werden. Prüfen Sie, inwieweit sich die einzelnen Abschnitte auf Ihre aktuellen Release-Pipelines auswirken.

Teil 3: Verfassen einer eigenen Richtlinie
Hier finden Sie eine Mustervorlage, die Sie bei der Erstellung Ihrer eigenen Richtlinie unterstützen kann.

TEIL 1: Zusammentragen von Informationen und Fachwissen

Zusammentragen von Fachwissen

Da sich bei dieser Richtlinie die Bereiche Sicherheit und DevOps überschneiden, ist es wichtig, dass die Mitarbeiter der einzelnen Abteilungen sowie der Produkt- und Entwicklungsabteilung ihren Beitrag leisten. Sie sollten auch Beiträge aus anderen relevanten Bereichen einholen, die von den Sicherheits- und Softwareänderungen in Ihrem Unternehmen direkt betroffen sein könnten. Folgende Bereiche werden im Zusammenhang mit dem Verfassen von Code-Signing-Richtlinien regelmäßig als Informationsquellen genannt:

- CISO
- DevSecOps/SecOps/InfoSec
- DevOps
- Produktsicherheit
- Produktmanagement
- Qualitätssicherung und Standortzuverlässigkeit
- Compliance
- Risikomanagement
- Kunden und Partner

Erfassen von Informationen

Softwareteams und ihre Entwicklertools

Der Wert forensischer Analysen
Ein proaktiver Sicherheitsansatz ist am effektivsten. Doch auch Fehler bieten eine wichtige Gelegenheit zu lernen. Führen Sie regelmäßig forensische Untersuchungen aktueller Sicherheitsverstöße und Pannen durch. Wenn bei Ihnen selbst keine Verstöße oder Fehler aufgetreten sind, untersuchen Sie die anderer. Fehler und Schwachstellen können Ihnen beim Verfassen einer Signierrichtlinie helfen, die die Bedrohungslage mildert und besseren Schutz Ihres Codes bietet. Stellen Sie sicher, dass Ihre Richtlinie die Schaffung von Artefakten vorsieht, die bei der Untersuchung und Behebung von Schwachstellen verwendet werden können.

Vorschriften, Richtlinien und Gesetze

Ermitteln Sie die internen und externen Vorschriften, die eingehalten werden müssen. Häufig stellen Normungsgremien Vorschriften, Rahmenwerke und zusätzliche Anleitungen zur Einhaltung von Vorschriften zur Verfügung. Achten Sie vor allem auf staatliche Richtlinien (EU NIS2 und US-Durchführungsverordnungen), Branchenvorschriften (CA/B Forum und PCI) sowie Regulierungsstandards (ISO, ENISA und NIST).

Prävention und Abhilfe

Code-Signing hilft nicht nur, Softwaremanipulationen zu verhindern, sondern produziert auch prüfbare Protokolle und Artefakte, die bei der Behebung von Problemen hilfreich sind. Dokumente wie signierte SBOMs, Build-Skripte, Container, Signatur- und Aktivitätsprotokolle enthalten nützliche Daten für die Untersuchung eines Malware-Angriffs oder des Risikos einer neuen ausnutzbaren Schwachstelle.

TEIL 2: Allgemeine Komponenten einer Software-Signing-Richtlinie

Umfang

Richtlinien werden mit einem bestimmten Ziel vor Augen verfasst. Machen Sie deutlich, für wen und was die Code-Signing-Richtlinie gilt bzw. nicht gilt. Beispiele:

Das sollte eingeschlossen werden

- **Menschen:** interne und externe Entwickler, Signierer, Prüfer, Auditoren und Produktsicherheitsteams
- **Code:** Produkte, die Sie für externe Kunden erstellen – kostenfreie und kostenpflichtige Angebote (externe Produkte) sowie selbst erstellte Systeme (interne Produkte)
- **Software-Artefakte:** SBOMs, Makros, Bibliotheken, Container und Build-Skripte

Das sollte ausgeschlossen werden

Anwendungen von Drittanbietern: Software, die Sie bei der Entwicklung Ihrer Softwareprodukte verwenden, sollte nicht von Ihnen signiert werden (z. B. Produktivitätstools wie Microsoft Office oder Entwicklertools wie JFrog). Jede Software von renommierten Drittanbietern wird von ihren eigenen Entwicklern digital signiert. Andere Richtlinien sollten sicherstellen, dass Ihre Organisation nur Software nutzt, die von einem verifizierten Hersteller signiert wurde.

Zugriff und Zugriffsrechte

Authentifizierung

Authentifizierungsschritte bestätigen, dass es sich bei dem Nutzer, der versucht, Zugriff auf das System zu erhalten, um die erwartete Person handelt und dass diese Person berechtigt ist, das System zu nutzen. In der Regel ist bereits ein Authentifizierungssystem vorhanden. Die Richtlinie besagt dann lediglich, dass es verwendet werden muss.

Für den Zugriff und die Autorisierung der Signierung muss gemäß den Mindestanforderungen des CA/Browser Forum eine Multifaktor-Authentifizierung oder eine Server-zu-Server-Authentifizierung verwendet werden.[1]

Rollenbasierte Zugriffskontrolle (RBAC)

Zugriff und Zugriffsrechte legen die Eigenschaften von Personen oder Servern fest, die an Code-Signing-Aktivitäten teilnehmen sollten (bzw. nicht daran teilnehmen dürfen), welche Aktionen sie vornehmen und welche Ressourcen sie nutzen dürfen.

RBAC[2] ist eine Best Practice im Bereich der IT-Sicherheit, die die Definition von Rollen und die Zuweisung von unterschiedlichen Zugriffsrechten beinhaltet. Sie wird auch als Aufgabentrennung bezeichnet. Rollen sollten sich an den zu erledigenden Aufgaben orientieren, z. B. Einreicher, Signierer oder Sicherheitsbeauftragter, und nicht an unspezifischen Gruppen wie „Administratoren“ oder „Führungsebene“. Um „Single Points of Failure“ zu vermeiden, sollte jede Rolle nach Möglichkeit mehreren Nutzern zugewiesen werden.

Selbst in einem kleinen Team kann die Zuweisung von Zugriffsrechten an Einzelpersonen eine große logistische Herausforderung darstellen. Die Zuweisung von Zugriffsrechten über Rollen erleichtert das Management der Zugriffsrechte und sorgt für Skalierbarkeit, wenn das Team wächst. Überlegen Sie, ob nicht auch andere Rollen als nur Signierer erforderlich sind. Wer sollte in der Lage sein, Zertifikate zu beantragen, auszustellen und zu widerrufen? Gibt es Rollen, die von PKI-Managern, Sicherheitsbeauftragten oder DevSecOps-Experten übernommen werden sollten?

Risikominderung durch das Least-Privilege-Prinzip[3]

Um eine bessere Kontrolle über die Sicherheit zu haben, sollte allen Rollen nur das Minimum an Zugriff und Zugriffsrechten eingeräumt werden. Dies ist besonders wichtig für den Zugang zu wichtigen Ressourcen, für Aktivitäten, die mit einem hohen Risiko verbunden sind oder nicht rückgängig gemacht werden können, und für Software mit hohem Risikopotenzial. Beispiele hierfür sind:

- **Kernressourcen:** Build-Server und HSMs
- **Aktivitäten, die nicht rückgängig gemacht werden können:** Löschung von Schlüsselpaaren und Widerruf von Zertifikaten
- **Mit hohem Risiko verbundene Vorgänge:** Export von Schlüsselpaaren
- **Software mit hohem Risikopotenzial:** Code mit Low-Level-Systemzugriff, z. B. Root-Zugriff in Linux

Das mit einem Softwareprodukt oder -projekt verbundene Risiko muss auf der Grundlage des potenziellen Schadens bewertet werden, der entstehen würde, wenn die Software schädlichen Code oder ausnutzbare Schwachstellen enthielte, die zu unbefugtem Zugriff auf Daten, zum Verlust von Systemfunktionen oder zu anderen Fehlern führen würden.

Die Kritikalität von Software ist ein häufig angesprochenes Thema in Bezug auf Vorschriften. In den USA ergibt sich die rechtliche Bedeutung des Begriffs „Criticality“ aus der Durchführungsverordnung 14028 die im Mai 2021 erlassen wurde. Das NIST bietet zahlreiche Definitionen und ergänzende Materialien, die Ihnen bei der Bestimmung der Kritikalität helfen.

Das Konzept von Software mit hoher Kritikalität findet sich auch an anderen Stellen, etwa in der Cyberresilienz-Verordnung der EU, wo Netzwerksoftware und andere kritische Produkte zusätzlichen Regeln und Bewertungen durch eine zertifizierte Stelle unterliegen.

Bei Software mit hoher Kritikalität und Aktivitäten, die mit einem hohen Risiko verbunden sind oder nicht rückgängig gemacht werden können, sehen Code-Signing-Richtlinien oft mehrere Genehmiger vor, um sicherzustellen, dass keine einzelne Person eine Änderung vornehmen kann.

Eine weitere Möglichkeit zur Einschränkung von Zugriffsrechten besteht darin, Teams nur Zugang zu den Projekten und Produkten zu gewähren, an denen sie arbeiten. So sollte beispielsweise ein Entwickler, der an der Infrastruktur arbeitet, nicht in der Lage sein, Code für einen Release zu signieren, der in den Verantwortungsbereich des Teams für mobile Anwendungen fällt. Die Zugriffsrechte werden also nicht nur auf die Rolle einer Person beschränkt, sondern auch auf die Software, die eine Person entwickelt.

Management von Zertifikaten und Schlüsselpaaren

Software-Umgebungen

Sowohl für Nicht-Produktions- als auch für Produktionsumgebungen müssen eindeutige Code-Signing-Schlüsselpaare und Zertifikate verwendet werden. Testumgebungen, die aus der Ferne zugänglich sind und sensible Daten verarbeiten, müssen wie Produktionsumgebungen behandelt werden.

Schlüsselpaare und Zertifikate haben ein Verhältnis von 1:1 und Sie benötigen mindestens einen Satz zum Signieren in Produktionsumgebungen und einen für Nicht-Produktionsumgebungen. Theoretisch könnten Sie für jeden Build ein neues Zertifikat und ein Schlüsselpaar generieren. Manche Teams tun dies auch. Dieses Verfahren hat den Vorteil, dass die Auswirkungen eines kompromittierten Schlüssels begrenzt werden.

[1] Die MFA-Anforderungen des CA/B Forum sind zu finden im Dokument „Baseline Requirements for the Issuance and Management of Publicly Trusted Code Signing Certificates Version 3.9.0“, Abschnitt 6.2.7.3 zur Speicherung privater Schlüssel für Signaturdienste und Abschnitt 6.5.1 zu spezifischen technischen Anforderungen im Bereich Computer Security.

[2] Laut der Definition von RBAC durch das NIST (National Institute of Standards and Technology) stützen sich viele Normen zur Einhaltung von Vorschriften (wie NIST SP 800-95 und NIST SP 800-53) auf die ordnungsgemäße Implementierung von RBAC.

[3] Weitere Informationen über das Least-Privilege-Prinzip finden Sie in der NIST-Veröffentlichung SP 800-53 unter Punkt AC-6.

Speicherung von Zertifikaten und Schlüsselpaaren

Seit dem 1. Juni 2023 schreiben Richtlinien und Standards vor, dass private Code-Signing-Keys sowohl für Standard-/OV (Organization Validation)- als auch für EV (Extended Validation)-Zertifikate auf gemäß dem CA/Browser Forum als sicher geltenden Hardwaregeräten generiert und gespeichert werden müssen.[4] Diese Anforderung bedeutet, dass öffentliche Zertifizierungsstellen (CAs) nicht länger die browserbasierte Schlüsselgenerierung und Zertifikatsinstallation unterstützen können, auch keine Prozesse im Zusammenhang mit der Erstellung einer CSR (Certificate Signing Request) oder der Installation Ihres Zertifikats auf einem Laptop oder Server. Die Bereitstellungsmethode wird bei der Bestellung oder Erneuerung von Zertifikaten festgelegt.

Akzeptierte sichere Hardwaregeräte

- Hardware-Token (z. B. USB-Krypto-Token)
- Hardware-Sicherheitsmodule (HSMs) – cloudbasiert oder On-Premises –, die mit FIPS 140-2 Level 2 oder Common Criteria EAL 4+ konform sind

Wo sollten Sie Schlüssel für Nicht-Produktionscode generieren, speichern und nutzen? Idealerweise sollten Sie ein FIPS- oder CC-konformes HSM verwenden. Es gibt unter Umständen jedoch auch kostengünstigere Möglichkeiten, z. B. ein softwarebasiertes System für das Secrets-Management. Risikoreiche Praktiken wie das Speichern von Schlüsseln auf unsicheren Dateisystemen oder im Quellcode sind niemals akzeptabel und Teammitglieder sollten sich darüber bewusst sein.[5]

Unterschied zwischen öffentlichen und privaten PKI: Signieren interner Software

Code-Signing-Vorgänge sind klar und einheitlich, wenn die Software für die Bereitstellung in App-Shops, Betriebssystemen und regulierten Browsern bestimmt ist. Wenn eine Signatur notwendig ist, damit Software ausgeführt werden kann, dann wird der Code auch signiert. Doch diese Anforderungen sind nicht willkürlich oder aus der Luft gegriffen. Signaturen schützen Software nachweislich vor Manipulation, Angriffen und anderen böswilligen Aktivitäten. Wenn Plattformen Code-Signing durchsetzen, verfolgen sie kein hohes Ideal, sondern verlangen eine praktische Maßnahme gegen Bedrohungen.

In den meisten Fällen wird die Signierung unternehmensinterner Software ignoriert, weil Unternehmen den ihnen bekannten Entwicklerteams und den ihre Systeme umgebenden Schutzmaßnahmen vertrauen. In ihrer Wahrnehmung lauern Bedrohungen nur jenseits der Firewalls, wenn der Code bereits hinaus in die Welt geschickt wurde. Es erscheint durchaus plausibel, dass Software, die nur zwischen Teams innerhalb eines Unternehmens ausgetauscht wird oder zur internen Nutzung auf unternehmenseigenen Servern bereitgestellt wird, sicher ist.

In Wirklichkeit ist auch interne Software anfällig für Angriffe. Wenn sich jemand Zugang zum System verschafft, ist nicht signierter Code ein leichtes Ziel, das für Angriffe auf die Infrastruktur des gesamten Unternehmens ausgenutzt werden kann. Als Best Practice ist die unternehmensinterne Signierung kein Ideal, sondern eine praktische Maßnahme, die Ihre Software und Ihr Unternehmen genauso schützt wie die anderen Maßnahmen zum Schutz vor böswilligen Dritten.

Software, die außerhalb Ihres Unternehmens verwendet wird oder auf die außerhalb Ihres Unternehmens zugegriffen wird, sollte mit einem von einer CA ausgestellten Schlüsselpaar signiert werden, das über Browser oder Standard-PKI-Listen leicht überprüft werden kann.

Software, die nur für den internen Gebrauch in Ihrem Unternehmen entwickelt wurde, kann eine interne PKI (auch als private PKI bezeichnet) verwenden, bei der die Zertifikate an eine vertrauenswürdige Stammzertifizierungsstelle gekoppelt sind, aber von Ihrem Unternehmen über eine interne zwischengeschaltete Zertifizierungsstelle (ICA) ausgestellt werden. Dies ist gängig bei Software, die intern entwickelt und nur intern verwendet wird.

Private PKI-Zertifikate und -Schlüsselpaare müssen genauso kontrolliert, verwaltet und gesichert werden wie die entsprechenden öffentlichen PKI-Zertifikate. Einige CLM-Systeme (Certificate Lifecycle Management) können sowohl öffentliche als auch private PKI-Elemente verwalten.[6] Administratoren müssen dann zusätzlich das öffentliche Zertifikat der privaten Root-CA in die interne Liste der vertrauenswürdigen Root-Zertifikate aufnehmen.

Bestandsaufnahme und Lebenszyklus von Zertifikaten

Sie sollten ein Inventar aller Code-Signing-Zertifikate führen, Ablaufdaten überwachen und proaktiv neue Zertifikate erstellen. Außer in sehr kleinen Unternehmen ist ein effektives Bestands- und Zertifikatsmanagement in der Praxis nur mit einem CLM-System möglich, das den Prozess automatisiert. Wenn Sie nur sehr wenige Zertifikate und Schlüssel verwenden, z. B. einen Satz für Tests und einen für die Produktion, müssen Backups an einem sicheren Ort aufbewahrt werden.[7]

Zertifikate müssen je nach Bedarf – d. h. je nach Häufigkeit und/oder der erwarteten Lebensdauer von Software-Releases – widerrufen und neu ausgestellt werden. Durch die Automatisierung der Zertifikatserneuerung wird sichergestellt, dass Software-Releases nicht durch abgelaufene Zertifikate gestört werden. Die Code-Signing-Arbeitsgruppe des CA/B Forum diskutiert darüber, die Lebensdauer eines Code-Signing-Zertifikats von etwa drei Jahren auf 460 Tage (rund ein Jahr und drei Monate) zu verkürzen. Wenn Sie das Erneuern von Code-Signing-Zertifikaten jetzt automatisieren, können Sie eine zukünftige Umstellung reibungsloser bewältigen.

Zertifikate und Schlüssel sollten regelmäßig rotiert und/oder ersetzt werden, um das Sicherheitsniveau aufrechtzuerhalten. Ein regelmäßiger Wechsel begrenzt den Schaden, den ein verlorener oder gestohlener Schlüssel anrichten kann. Einige CLM-Systeme verfügen über Funktionen zum automatischen Rotieren von Schlüsseln.

[4] Weitere Informationen über das Ausstellen und Management öffentlich vertrauenswürdiger Code-Signing-Zertifikate finden Sie in den Mindestanforderungen des CA/Browser Forum.

[5] Eine Leidensgeschichte: Was passierte als Kali Linux den Repo-Signierschlüssel verlor.

[6] DigiCert bietet eine Lösung für öffentliche und private PKI.

[7] Siehe Fußnote 4.

Key-Verschlüsselung

Um größtmögliche Sicherheit zu gewährleisten, sollten Sie die Schlüssel auf der höchstmöglichen Stufe verschlüsseln, basierend auf den Entschlüsselungsmethoden, die die Ressourcen, die die Schlüssel nutzen, verwenden können. In Ihrer Richtlinie sollte eindeutig festgelegt sein, welcher Algorithmus zu verwenden ist und welches Mindestmaß an Key-Verschlüsselung für jeden Zertifikatstyp oder Anwendungsfall (z. B. Produktions-Release für mobile Anwendungen) akzeptabel ist.

Das NIST hat 2030 als Frist für die allmähliche Abschaffung weit verbreiteter, veralteter Verschlüsselungsalgorithmen wie RSA, ECDSA, EdDSA, DH und ECDH festgelegt. Diese Algorithmen sind anfällig für Quantencomputer und dürfen ab 2035 gar nicht mehr genutzt werden.

Bei einer automatischen Erneuerung auf der Grundlage von Profilen kann über eine Vorlage ein Verschlüsselungs-Upgrade konfiguriert werden, das dann automatisch und systematisch erfolgt, wenn ein Schlüssel ersetzt wird. Branchenexperten und Analysten gehen davon aus, dass Unternehmen ihre Verschlüsselung in den nächsten fünf Jahren mehr als einmal aktualisieren müssen.

Code-Signing-Prozess

Sicherheitsüberprüfungen

Der Software-Quellcode muss eine Sicherheitsüberprüfung gemäß den Sicherheitsrichtlinien des Softwareentwicklungszyklus (Software Development Life Cycle; SDLC) bestehen.[8]

Jede Version und jede zugehörige Komponente muss auf bösartige Indikatoren und bekannte Schwachstellen gescannt werden. Malware und anfällige Software sind in Test- und anderen Nicht-Produktionsumgebungen genauso inakzeptabel wie in der Produktion. Aus diesem Grund sollte Software im Entwicklungszyklus und im CI/CD-Prozess auf derartige Probleme hin überprüft werden.

Nicht alle Schwachstellen sind es wert, einen Build oder sogar eine Code-Lieferung zu stoppen. Viele haben ein geringes Risikopotenzial und andere lassen sich kaum oder gar nicht ausnutzen. Sie müssen einen Schwellenwert für Schwachstellen festlegen, um zu bestimmen, wann ein Build oder Release gestoppt werden soll.

Signieren Sie keine Software, die nachweislich Malware oder eine ausnutzbare Sicherheitslücke enthält.

Automatisierte Prozesse

Am besten ist es, den Code-Signing-Prozess als Teil des SDLC und der CI/CD-Pipeline zu automatisieren. Verwenden Sie eine zugelassene Zertifizierungsstelle (CA) für Code-Signing-Zertifikate in Produktionsumgebungen. Idealerweise sollten Entwickler einen eindeutigen Signierschlüssel verwenden, um jeden Code zu signieren, den sie einchecken. Die einzelnen Unterschriften müssen überprüft werden, bevor ein Release signiert wird.

Signing-Keys für Produktionscode dürfen nicht von mehreren Nutzern gemeinsam verwendet werden, es sei denn, die Aktivitäts-protokollierung kann die Aktivitäten einzelner Nutzer, einschließlich der Nutzer von Diensten, aufzeichnen und identifizieren.

Unternehmen, die den Prozess automatisieren, profitieren zudem von den Sicherheitsvorteilen, die sich aus einer regelmäßigen Schlüssel- und Zertifikatsrotation ergeben. Ohne einen automatisierten Ansatz ist eine Rotation nach dem „Lazy Key Rotation"-Prinzip möglicherweise der praktischste Ansatz.

Zeitstempel

Releases, die nach Ablauf des Code-Signing-Zertifikats gültig sind, müssen einen Zeitstempel eines zugelassenen Zeitstempeldienstes enthalten, um den Zeitpunkt der Signierung nachzuweisen.

Seit dem 15. April 2025 gelten bei den TSA (Timestamp Authorities) striktere Sicherheitsanforderungen in Bezug auf das Speichern von Zertifikaten und Schlüsseln und Hash-Algorithmen müssen mindestens SHA-2 entsprechen. Zeitstempel-Zertifikate müssen darüber hinaus über eine EKU (Extended Key Usage) für die Zeitstempelung verfügen. Achten Sie darauf, dass Ihre Zeitstempel den neuen Anforderungen des CA/B Forum entsprechen.

Signieren von Release-Artefakten und Software-Stücklisten (SBOMs)

Release-Artefakte wie Code-Branches, SBOMs, Build-Skripte und VEX-Dokumente (Vulnerability Exchange Documents) müssen signiert und gespeichert werden, um Manipulationen an Stellung-nahmen und Archiven zu erkennen.

Manche Organisationen signieren verschiedene Artefakte mit unterschiedlichen Schlüsseln, damit sich Probleme, die durch einen kompromittierten Schlüssel verursacht werden, nur auf ein Element des Build-Pakets auswirken.

Artefakte sollten sicher aufbewahrt werden und es müssen möglicherweise Methoden zum Teilen mit Kunden oder Leitungsgremien bereitgestellt werden. SBOMs können auch Teil eines Angebotspakets, einer Produkteinreichung oder eines Audits sein.

Protokollierung und Nachverfolgung

Pflegen Sie einen umfassenden Prüfpfad mit Nachweisen für Code-Reviews, Sicherheitsscans, Aktivitäten rund um Code-Signing-Keys sowie Zugriffs- und Berechtigungsänderungen.

Aktivitäten müssen einzelnen Nutzern und Dienstkonten zugeordnet werden. Dies kann durch eindeutige Schlüssel für jeden Signierer oder durch Software-Signaturprotokolle geschehen, wenn eine Gruppe Schlüssel gemeinsam nutzt.

Es sollte ein Standardverfahren für die Weitergabe von Protokollen an Auditoren und die zu verwendenden Dateiformate geben.

[8] Unterstützung bei der Entwicklung sicherer SDLC und anderer effektiver Sicherheitsrichtlinien erhalten Sie bei OWASP.

Krypto-Agilität zur Vorbereitung und Reaktion auf Veränderungen

Vorbereitung und Reaktion auf Vorfälle und unerwartete Änderungen

Angesichts der zunehmenden Anzahl und Häufigkeit von Angriffen auf die Softwarelieferkette und den SDLC ist es unerlässlich, auf Änderungen mit kurzer Vorlaufzeit vorbereitet zu sein.

Sie sollten Verfahren für das Dokumentieren, Widerrufen und Neuausstellen kompromittierter Zertifikate und Schlüsselpaare definieren und weitgehend automatisieren.

Im Anschluss an das Widerrufen und Wiederherstellen müssen Sie den Code und die Software-Artefakte, die mit einem kompromittierten Schlüssel signiert wurden, erneut signieren und bereitstellen.

Unternehmen, die diese Prozesse automatisieren, verwenden Zertifikats- und Schlüsselpaar-Aliase und nicht die eigentlichen Identitäten. Dadurch kann das CI/CD-System ohne Unterbrechungen und ohne Anpassen der Einstellungen weiterlaufen.

Richten Sie Prozesse für die Aktualisierung der Key-Verschlüsselung als einmaliges Ereignis, als Teil eines Zeitplans unter Berücksichtigung des Ablaufens von Zertifikaten und als Massenvorgang ein. Mit diesen Methoden decken Sie sowohl kleine als auch große Vorfälle sowie zeitlich festgelegte Upgrades auf der Grundlage von Vorschriften ab.

Vorbereitung und Reaktion auf erwartete Änderungen

Verordnungen ändern sich ständig; allerdings werden sie oft schrittweise eingeführt oder geben Zeit für Systemänderungen. Sie sollten Verfahren für das Dokumentieren und das schrittweise Durchführen von Änderungen definieren und weitgehend automatisieren. Nehmen Sie zum Beispiel Aktualisierungen vor, wenn die Zertifikate ablaufen und neu ausgestellt werden müssen.

Es gibt zwei große Veränderungen, auf die Unternehmen sich einstellen müssen: kürzere Lebenszyklen und Änderungen bei den Verschlüsselungsalgorithmen.

Kürzere Lebenszyklen sind schon seit einiger Zeit im Gespräch und die kürzlich abgesegnete Abstimmung zur Verkürzung der Lebensdauer von TLS-Zertifikaten ist der wichtigste Indikator dafür, dass es sich nur um eine Frage der Zeit handelt.

An der Schwelle zur Post-Quanten-Kryptografie (PQC), die die meisten der heute verwendeten Verschlüsselungsalgorithmen aushebeln wird, werden wir sowohl neue komplexe Algorithmen als auch die Abschaffung bestehender Algorithmen erleben.

Richten Sie Prozesse für die Aktualisierung der Key-Verschlüsselung als einmaliges Ereignis, als Teil eines Zeitplans unter Berücksichtigung des Ablaufens von Zertifikaten und als Massenvorgang ein. Mit diesen Methoden decken Sie sowohl geringfügige als auch umfassende Anpassungen sowie zeitlich festgelegte Upgrades auf der Grundlage von neuen Vorschriften ab.

Überprüfung und Aktualisierung

Überprüfen Sie Ihre Richtlinien jährlich und aktualisieren Sie sie bei Bedarf, um ihre Wirksamkeit und die Anpassung an Branchenstandards und organisatorische Veränderungen zu gewährleisten.

TEIL 3: Verfassen einer Richtlinie für die Softwaresignierung

Verschaffen Sie sich einen Vorsprung, indem Sie unsere bearbeitbare Richtlinienvorlage herunterladen. Sie enthält Stellen, an denen Sie die in diesem Leitfaden behandelten Konzepte aufnehmen können. Verwenden Sie die Vorlage in Kombination mit diesem Leitfaden, um eine gut durchdachte Software-Signing-Richtlinie zu verfassen. **Laden Sie die Richtlinienvorlage hier herunter.**

Über DigiCert

Als einer der weltweit führenden Anbieter digitaler Vertrauenslösungen sorgt DigiCert dafür, dass Unternehmen und Einzelpersonen digitalen Interaktionen in dem Wissen vertrauen können, dass ihre digitale Infrastruktur und ihre Anbindung an eine Welt voller Online-Transaktionen sicher und geschützt sind. DigiCert® ONE, die Plattform für Digital Trust, bietet Unternehmen eine zentrale Anlaufstelle für Einblicke in und die Kontrolle über eine Vielzahl von öffentlichen und privaten Anwendungsbereichen, in der das Vertrauen eine wichtige Rolle spielt. Dazu gehören der sichere Zugriff auf Unternehmenssysteme, sichere Business-Kommunikation sowie der Schutz von Websites, Software, Identitäten, Inhalten und Geräten. DigiCert bietet nicht nur preisgekrönte Softwarelösungen an, sondern hat sich nicht zuletzt auch durch seine branchenweite Führungsrolle bei Standards, Support und Betrieb als bevorzugter Anbieter digitaler Vertrauenslösungen bei Unternehmen auf der ganzen Welt einen Namen gemacht. Weitere Informationen finden Sie unter digicert.de.