IT-SICHERHEIT · DEVOPS · COMPLIANCE

CODE SIGNING

Warum es heute an Hardware gebunden ist – und wie es trotzdem skalierbar bleibt

Felix Heitländer
Senior Customer Support Engineer, SEH Computertechnik GmbH

Mit Zertifikaten auf USB-Token lässt sich sicherstellen, dass Software nicht manipuliert wurde.

Die Benutzung sogenannter Dongles ist sicher, kostengünstig und sogar auf virtuellen Maschinen in der Cloud einsetzbar.

Software ist längst mehr als ein Produkt. Sie steuert Maschinen, verarbeitet sensible Daten, automatisiert Geschäftsprozesse und entscheidet im Zweifel darüber, ob ein Unternehmen arbeitsfähig bleibt oder nicht. Entsprechend hoch ist der Schaden, den manipulierte oder kompromittierte Programme verursachen können. Die zentrale Frage lautet daher: Woher stammt eine Software – und ist sie frei von Manipulationen?

Die Antwort liefert Code Signing: Digitale Signaturen ermöglichen die eindeutige Prüfung von Herkunft und Integrität einer Software. Moderne Betriebssysteme und Unternehmensrichtlinien verlassen sich konsequent auf diese Mechanismen. Unsigned? Wird blockiert – oder zumindest sehr skeptisch beäugt. Code Signing ist damit eine grundlegende Voraussetzung für den professionellen Softwarebetrieb.

Diese Entwicklung betrifft längst nicht mehr nur klassische Desktop-Anwendungen. Auch Firmware, Skripte, Container-Images und automatisiert ausgelieferte Updates unterliegen strengen Signaturprüfungen. Je stärker Software in kritische Infrastrukturen und industrielle Prozesse eingebunden ist, desto höher ist der Anspruch an ihre Integrität. Code Signing wird damit zum verbindenden Element zwischen Entwicklung, Betrieb und Compliance.

Vom Sicherheitsmerkmal zur Pflichtdisziplin

Moderne Betriebssysteme blockieren unsignierte Programme standardmäßig oder versehen sie mit deutlichen Warnhinweisen. In Unternehmensnetzwerken ist die Ausführung unsignierter Software häufig vollständig untersagt.

Der Grund: Ohne Code Signing könnten Angreifer manipulierte Programme einschleusen, die äußerlich nicht von legitimer Software zu unterscheiden sind. Digitale Signaturen schaffen eine verlässliche technische Vertrauensbasis – unabhängig davon, ob die Software lokal installiert, aus dem Internet geladen oder automatisiert verteilt wird.

Damit ist Code Signing kein Randthema mehr für Entwickler, sondern ein zentraler Bestandteil der IT-Sicherheitsstrategie.

Verschärfte Regeln für private Schlüssel

Mit der wachsenden Bedeutung von Code Signing sind auch die Anforderungen an den Schutz der dafür verwendeten Zertifikate gestiegen. Besonders im Fokus stehen die privaten Schlüssel. Gelangen diese in falsche Hände, lassen sich selbst Schadprogramme scheinbar legitim signieren.

Regulatorische Vorgaben aus 2023 schreiben deshalb vor, dass diese Schlüssel nicht mehr ungeschützt auf Dateisystemen liegen dürfen. Stattdessen müssen sie in speziell gesicherten Hardware-Speichern abgelegt werden, die definierte Sicherheitsstandards erfüllen (EAL 2+ Common Criteria, FIPS 140-2). Damit soll sichergestellt werden, dass Schlüssel nicht kopiert, exportiert oder unbemerkt missbraucht werden können.

In der Praxis hat sich dafür der Einsatz von USB-basierten Sicherheits-Token etabliert. Kryptografisch sinnvoll – organisatorisch jedoch nicht ohne Nebenwirkungen.

Für Unternehmen bedeutet diese Regulierung einen klaren Umbruch: Verantwortlichkeiten, Zugriffskonzepte und technische Schutzmaßnahmen müssen dokumentiert und nachvollziehbar umgesetzt werden. Code Signing ist damit auch organisatorisch Teil von Governance und Compliance.

Wenn Hardware auf moderne Entwicklungsmodelle trifft

Moderne Softwareentwicklung ist hochgradig automatisiert. Build-Prozesse laufen in virtuellen Maschinen, CI/CD-Pipelines signieren Software automatisiert, Entwicklungsumgebungen werden dynamisch bereitgestellt. Cloud-Infrastrukturen erlauben es, Rechenleistung exakt nach Bedarf zu skalieren.

Ein physischer USB-Dongle passt nur bedingt in dieses Bild. Er ist lokal gebunden, nur eingeschränkt parallel nutzbar und nicht ohne Weiteres in eine Cloud-Instanz integrierbar. Hier entsteht ein Spannungsfeld zwischen Sicherheitsanforderungen und Entwicklungsrealität.

Ohne geeignete Konzepte führt das zu typischen Problemen: Build-Prozesse müssen unterbrochen werden, zusätzliche Token werden angeschafft, Sicherheitsmechanismen werden im Alltag „temporär“ umgangen – was selten temporär bleibt. Am Ende leidet entweder die Sicherheit oder die Effizienz.

Besonders deutlich wird dieser Zielkonflikt in dynamischen Umgebungen mit parallelen Build-Jobs. Jede fest an Hardware gebundene Ressource wird dort zum potenziellen Engpass.

Das Kernproblem: fehlende Entkopplung

Der zentrale Konflikt entsteht nicht durch Code Signing selbst, sondern durch die enge Kopplung von Sicherheits-Token und physischem Rechner. Solange ein Zertifikat ausschließlich dort verfügbar ist, wo der Dongle physisch angeschlossen ist, bleibt jede Virtualisierung ein Sonderfall.

In der Praxis bedeutet das: steigender Administrationsaufwand, zusätzliche Token, schlechte Skalierbarkeit – und Workarounds, die Sicherheitskonzepte langfristig aushöhlen.

Die logische Konsequenz ist eine klare Entkopplung: Das Zertifikat bleibt physisch geschützt, der Zugriff darauf wird flexibel gestaltet.

USB über das Netzwerk: altbekannt, neu gedacht

Ein praktikabler Ansatz besteht darin, USB-Sicherheits-Token zentral zu betreiben und über das Netzwerk bereitzustellen. Spezialisierte Software emuliert auf den Zielsystemen einen lokalen USB-Controller, während der Dongle physisch an einem anderen Ort verbleibt.

Für die Signiermaschine erscheint der Token wie lokal angeschlossen. Tatsächlich wird der USB-Datenverkehr kontrolliert und sicher über das Netzwerk übertragen. Virtuelle Maschinen, Container oder Cloud-Instanzen können so hardwaregeschützte Zertifikate nutzen – ohne physisch von einem direkt angeschlossenen Dongle abhängig zu sein.

Beispiel:
Ein Softwarehersteller betreibt seine CI/CD-Pipeline in einer Cloud-Umgebung mit automatisch skalierenden Build-Agents. Statt jedem Agent einen eigenen Token zuzuweisen, wird ein zentraler Hardware-Token im Rechenzentrum betrieben. Die Build-Instanzen greifen bei Bedarf über das Netzwerk darauf zu, signieren das Artefakt und geben den Token wieder frei. Ergebnis: volle Automatisierung bei gleichzeitig hardwaregeschütztem Schlüssel.

Automatisierung als Schlüssel zum Erfolg

Der eigentliche Mehrwert entsteht durch die Automatisierung dieses Zugriffs. Über Kommandozeilen-Schnittstellen kann ein Sicherheits-Token gezielt reserviert, genutzt und wieder freigegeben werden. Die Signierung wird damit integraler Bestandteil des Build-Prozesses.

Ein Token kann von vielen Systemen nacheinander genutzt werden – gebunden nur für die Dauer des Signiervorgangs. Das reduziert Hardwarebedarf und Kosten erheblich.

Neben Effizienz steigert die Automatisierung auch die Prozesssicherheit. Wiederholbare Abläufe vermeiden Fehlkonfigurationen und schaffen eine klare Trennung zwischen Entwicklung und sicherheitskritischen Schlüsseln.

Beispiel:
Ein Industrieunternehmen entwickelt Firmware für Steuerungssysteme an mehreren Standorten. Früher hatte jedes Team einen eigenen Dongle – inklusive Verwaltungsaufwand und unklarer Zugriffsdokumentation. Durch die zentrale Bereitstellung eines kleinen Pools von Hardware-Token lassen sich Signiervorgänge nun standortübergreifend steuern und vollständig protokollieren. Das Audit wird entspannter. Und die IT-Koordination auch.

Cloud-Nutzung ohne Sicherheitsverlust

Auch in Cloud-Umgebungen lässt sich dieses Modell umsetzen. Voraussetzung ist eine saubere Netzwerkarchitektur: USB-Datenverkehr wird über definierte TCP/IP-Verbindungen übertragen und per VPN, Firewall-Regeln oder Segmentierung abgesichert.

Zusätzliche Schutzmechanismen wie verschlüsselte Kommunikation über TLS Kanäle und passwortgestützte Zugriffsbeschränkungen sorgen dafür, dass Zertifikate auch außerhalb der eigenen Infrastruktur geschützt bleiben.

Gerade in hybriden Szenarien lassen sich so Signierprozesse zentralisieren, während Rechenleistung flexibel verteilt wird.

Positive organisatorische Effekte

Weniger Zertifikate bedeuten weniger Verwaltungsaufwand. Zentrale Token lassen sich besser überwachen, Zugriffe protokollieren und Prozesse standardisieren.

In größeren Entwicklungsumgebungen entsteht so Transparenz, wo zuvor individuelle Workarounds dominierten.

Sicherheit und Effizienz schließen sich nicht aus

Hardware-gebundene Code-Signing-Zertifikate sind heute unverzichtbar. Gleichzeitig erfordern moderne Entwicklungsprozesse Flexibilität und Skalierbarkeit. Die netzwerkbasierte Bereitstellung von USB-Sicherheits-Token verbindet beide Welten.

Unternehmen profitieren von stabilen Build-Prozessen, reduzierten Kosten und vollständiger Einhaltung regulatorischer Vorgaben – ohne ihre Entwicklungsfreiheit einzuschränken.

Wohin die Reise geht

Die Anforderungen an Code Signing werden weiter steigen. Softwarelieferketten werden komplexer, Entwicklungszyklen kürzer und regulatorische Vorgaben strenger. Gleichzeitig wächst der Druck, Prozesse vollständig zu automatisieren.
Ohne geeignete Infrastruktur droht Code Signing zum strukturellen Engpass zu werden. Fehlende Skalierbarkeit und starre Hardware-Abhängigkeiten stehen im Widerspruch zu modernen DevOps-Modellen.
Zugleich steigen Transparenz- und Audit-Anforderungen. Wer wann welchen Code unter welchen Bedingungen signiert hat, muss nachvollziehbar sein.
Code Signing entwickelt sich damit vom Einzelwerkzeug zum infrastrukturellen Baustein. Der Zugriff auf geschützte Zertifikate wird zunehmend als kontrollierbarer, skalierbarer Service verstanden.
Langfristig entsteht eine Umgebung, in der Code Signing kaum noch auffällt – nicht, weil es unwichtig wird, sondern weil es zuverlässig im Hintergrund funktioniert.

Hintergrundinformationen

DigiCert. (2024). Code signing changes in 2023 (Private key storage requirements on FIPS 140-2 Level 2 / Common Criteria EAL4+ hardware). DigiCert Knowledge Base. **https://knowledge.digicert.com/alerts/code-signing-changes-in-2023**

DigiCert. (2025). New private key storage requirement for Code Signing certificates. DigiCert Knowledge Base. **https://knowledge.digicert.com/general-information/new-private-key-storage-requirement-for-standard-code-signing-certificates-november-2022**

CA/Browser Forum. (2025). Code Signing Baseline Requirements (CSBR)– Private key protection and hardware requirements (FIPS 140-2 / Common Criteria). **https://cabforum.org/working-groups/code-signing/requirements**

Entrust. (2022). CA/Browser Forum updates: Requirements for code signing certificate private keys (hardware storage mandate). **https://www.entrust.com/blog/2022/09/ca-browser-forum-updates-requirements-for-code-signing-certificate-private-keys**

Made in Germany