digicert®

Frequently Asked Questions (FAQs) on code signing and Software Trust Manager

What is code signing?

Code signing is a method to confirm that code or other digital binaries have not been altered. This method leverages the Public Key Infrastructure (PKI) framework to attests to the integrity of the code or binaries. Code signing acts like a digital shrink wrap: the process of code signing creates a "package" with the signed code or file, if the "packaging" was not damaged when received, then the file has not been tampered with and can be trusted. And vice-versa: if the "packaging" was damaged then the file has been tampered with and cannot be trusted.

Why is code signing important?

Code signing minimizes the risks of code tampering. With signed code, the recipient gets a security warning when the integrity check fails during download. This helps recipients to avoid downloading tampered code which may contain malware.

What types of files can be signed?

You can sign all types of digital binaries including drivers, firmware, containers, applications, mobile apps as well as source code artifacts.

What is the traditional approach to code signing?

Public code signing certificates are issued based on the same fundamentals as Public SSL/TLS certificates: organization details are validated, and a certificate is issued to the requester to use on behalf of their organization. For TLS, that certificate usually resides in one location and provides trust based on the website domain(s) and sub-domain(s) that it secures. Code signing certificates, however, are typically used to sign a multitude of software releases whereby, multiple developers and or engineering teams will need access at different times.

The traditional approach to code signing is where public code signing certificates are issued to an organization from a Certificate Authority (CA) and the organization has to take responsibility for protecting the private key locally while still permitting those with signing responsibilities to have access at the appropriate time. This is a difficult task for organizations to undertake and can be both expensive and time consuming to implement correctly. In a dynamic product development environment, you may have team members entering, exiting, on-leave, or on-travels. Others may be assuming temporary roles in signing or need access to the keys at certain times. The appropriate keys need to be available at the right time to the right developer. All the keys will need to be protected always.

What could happen if my code signing private keys were to fall into the wrong hands?

Code signing private keys are critical assets in every organization. By signing their files or applications, a business is stating to the world that this file or application came from them and if the signature is intact, the user may download and install the software, firmware, or driver in confidence. The message provides confidence to the users that the application or file does not contain any malware, vulnerabilities nor corruption from tampering, i.e. it is safe to install.

However, if a bad actor did get their hands on a valid private key, they can impersonate your organization with the intention to cause harm to your users, business, or both. How do they do this? Malware authors create applications or files that contain malware or vulnerabilities to infect or exploit user devices or IT infrastructure. By signing these bad applications or files with a valid private key and associated code signing certificate, they can get their code successfully past browser and operating system checks and thus, present their applications or files to users as being in good standing.

Discovery of key misuse may take weeks or months. Post-discovery, the victim organization will have to initiate costly remediation procedures which may include notification and customer support of impacted users, revocation of compromised keys and affected binaries and reissuance of affected binaries that will need to be signed with new keys.

Therefore, it is very important for organizations to protect business critical code signing private keys as a company's reputation and financial well-being depends on it.

What are some ways to protect code signing keys?

- Store keys in Hardware Secure Modules (HSMs) and other highly secure services
- Store keys off-line until needed (bring online when required)
- Set up granular controls on signing to include user roles, file types and schedule
- Add approval workflows for business-critical signatures, as necessary. (Note: this may impact signing automation workflows)

What is the difference between traditional local code signing and next-gen code signing as a service?

The big challenge with local code signing is that the private key of the signing certificate needs to be available to the user when they sign applications or files. Traditionally this meant that developers shared the private keys in a casual and insecure way, for example on unsecured drives, to facilitate key availability. This type of practices resulted in multiple copies of private keys being kept on personal computers and/or build machines without Security Officers being able to discover or track the location of these critical assets.

When using the code signing-as-a-service approach such as with **DigiCert® Software Trust Manager**, the private keys are created and protected in a secure location. This could be in a hosted solution or locally on a service which is run on-premises. Regardless of where the code signing service runs, the private keys are protected in a central key repository or vault. Developers and machines (for example, build machines) have to be authorized and granted access prior to using the keys for signing. This approach

enables organizations to adhere to code signing best practices using controls to manage access and signing workflows.

What are some best practices in code signing?

Here is a list of best practices:

- Secure code signing keys
- Prevent code signing keys from being shared
- Set up controls over who is allowed to sign what
- Track, audit and report code signing activities including who signed what and when, what are the keypairs and associated certificates
- Leverage hash signing for expediency and security
- Integrate code signing into development process and leverage automation, for efficiencies
- Enforce organizational standards when creating keypairs and certificates

What is Software Trust Manager?

DigiCert® Software Trust Manager protects the integrity of software across the software development lifecycle and software supply chain, reducing risk of code compromise, enforcing corporate and regulatory policy, and delivering fine-grained key usage and access controls for code signing.

Software Trust Manager minimizes the risks of security breach and malware propagation in the development, build and release of software, enabling companies to protect against software supply chain attacks and comply with government regulations.

Software Trust Manager can be deployed on-premises, in the cloud, or accessed as a SaaS offering. It is built on the DigiCert ONE platform, which enables seamless overall management of PKI in one place. Secure Software Manager enables Audit trailrail of signing activity for forensics and full accountability.

Where can I get more information on Software Trust Manager or code signing?

Please download the DigiCert® Secure Software Manager datasheet or visit https://www.digicert.com/software-trust-manager for more information.