

Enable Successful Software Installs with New Digital Signing Standards



Enable Successful Software Installs with New Digital Signing Standards

Digital signing is a critical security process that ensures the authenticity and integrity of software. It involves applying a cryptographic signature to software artifacts, such as installers and executables to confirm both the identity of the software's creator and that the code has not been tampered with since its creation. This paper will explore the latest changes to digital signing standards and equip you with the knowledge and tools necessary to navigate these changes effectively for a seamless installation experience.

Introduction to Digital Signing

Digital signing uses a cryptographic key pair, consisting of a private key and a public key, to establish trust. The private key is securely held by the software vendor and used to generate the digital signature, while the public key is distributed to users for verifying the signature. This process acts as a digital "shrink wrap," preventing tampering and assuring the end user that the software is authentic and unmodified.

In the same way that individuals will physically sign documents, trusting that others won't easily forge that signature, digital signing verifies the origin and authenticity of the software and prevents unauthorized tampering.

There are two primary reasons for applying a digital signature:

1. **Tamper Prevention**
Once signed, a piece of software cannot be altered without invalidating the signature, which serves as a safeguard against malicious modifications. This is essential when distributing software, ensuring that users install a product that is identical to what was originally released.
2. **Identity Verification**
A digital signature provides proof of origin, allowing users to verify that the software comes from a trusted source, such as a recognized software vendor. This eliminates the risk of installing code from unknown or potentially harmful sources.

Digital signing has greatly evolved over the years, particularly in how security is managed. Fundamentally, it is akin to a handshake between the private key (used to generate the signature) and the public key (used to verify it). The challenge has been making this verification universal, ensuring users don't need to manually exchange keys.

This is where the Certificate Authority and Browser Forum (CAB Forum) comes into play. This body establishes certificate authorities (CAs) across platforms such as Windows, Linux, and macOS. The public key is embedded in the signed software, but it is verified through a chain of trust. The public key links to an intermediate certificate, which in turn links to a root certificate, already present on the device where the software is being installed.

During installation, the system verifies the digital signature by checking this chain of trust, validating the public key, ensuring that the intermediate and root certificates are reliable, and confirming that the signature has not been revoked. This entire process ensures that the software has not been tampered with and is from a trustworthy source.

In practice, digital signing is widely used in software development and distribution, with platforms like InstallShield supporting the integration of these technologies. By verifying the signature during installation, users are reassured that the software they are installing is both legitimate and secure.

The Importance of Time Stamping in Digital Signing

While digital signing ensures the authenticity and integrity of software, time stamping is a crucial addition to this process that extends the trust and lifespan of the digital signature. Without a time stamp, a digital signature is only valid as long as the certificate used to sign the software is valid, typically around three years. However, many software applications, especially those used in long-life products like embedded systems or hardware devices, must remain operational for much longer than that.

Time stamping solves this challenge by ensuring that the software's dependability can extend beyond the certificate's expiration date. A time stamp attaches the exact moment when the software was signed, allowing systems to verify that it was signed during the period when the certificate was valid. As a result, even after the certificate itself expires, the signature remains valid, and the software is still trusted.

For example, imagine a scenario where software is built for devices like televisions or cars that are meant to last far longer than three years. Without time stamping, once the certificate used to sign the software expires, the system would no longer recognize the software as legitimate. With time stamping, however, the signature can be preserved for over a decade, ensuring that the software is verified for the device's full lifecycle.

Evolution of Digital Signing Standards

Over the past decade, digital signing standards have undergone substantial changes due to heightened security concerns and the growing need to safeguard software integrity. Originally, digital signing was a relatively simple process: software vendors would acquire a certificate along with a corresponding private key, often in the form of a PFX file. This file could be stored on a build server and used repeatedly to sign software. However, this approach posed significant risks, as stolen PFX files could be sold on the dark web, enabling attackers to sign malware with legitimate certificates and trick end-users into trusting compromised software.

To counter these risks, the CAB Forum established new guidelines to secure digital signing practices. A landmark change in June 2023 required that certificates could only be issued if the private key was generated within a FIPS-compliant device, such as a hardware security module (HSM). This mandate effectively barred the storage of PFX files on build servers, addressing vulnerabilities that had led to incidents like GitHub's accidental exposure of a PFX file on a server. This update compelled organizations to adopt secure hardware or cloud-based HSM services to safeguard their signing keys. Although initially met with resistance due to increased complexity, software vendors have come to appreciate the enhanced security and protection against breaches that these standards provide.

Simultaneously, a growing trend towards short-term certificates has gained traction within this framework, offering significant advantages in regulatory compliance and security. These certificates, typically valid for less than a year, allow organizations to adapt quickly to evolving security standards and regulatory mandates. Particularly in regulated industries like finance and healthcare, short-term certificates enable rapid compliance adjustments without committing to long-term certificates.

Short-term certificates also bolster security by allowing organizations to renew and rotate their keys frequently, mitigating the risk of exposure from breaches or vulnerabilities. Companies with robust certificate lifecycle management practices benefit from this flexibility, ensuring that no single key is used for extended periods. For organizations managing diverse signing requirements, holding both short-term and extended validation (EV) certificates simultaneously is viable, as it allows them to use EV certificates for high-trust applications while reserving short-term certificates for less critical deployments.

As discussions continue within the CAB Forum regarding reduced validity periods for EV code signing certificates, short-term certificates are likely to play an increasingly central role in digital signing. This shift reflects the industry's commitment to enhanced security, positioning organizations that adopt short-term certificates to better navigate compliance challenges and protect their software's integrity. By proactively embracing these evolving standards, businesses can strengthen their reputation as reliable, security-conscious vendors.

Types of Code Signing: Public Trust vs. Private Trust

When it comes to code signing, software vendors have two primary options: Public Trust Code Signing and Private Trust Code Signing. Each has its own set of pros and cons, depending on the ecosystem and the specific needs of the organization.

Public Trust Code Signing

Definition: Public Trust Code Signing is designed for software that will be distributed widely to end users across open ecosystems like the Internet. This includes platforms such as Windows, macOS, and Linux, where software needs to be easily recognized and trusted by a diverse user base.

Use Cases: Ideal for independent software vendors (ISVs) distributing applications to the general public, where trust and a smooth installation process are paramount.

Pros:

- **Widespread Recognition**
Public trust certificates are issued by recognized Certificate Authorities (CAs) and are built into the trust stores of major operating systems and browsers. This means users are less likely to encounter security warnings when installing the software.
- **Ease of Use**
Once obtained, public trust certificates are relatively easy to use, allowing for a seamless user experience when installing applications.
- **Standard Compliance**
Public trust certificates are subject to stringent validation requirements, ensuring a level of security and accountability that protects both the vendor and the end user.

Cons:

- **Cost**
Public trust certificates can be more expensive than private options due to the validation and management processes involved.
- **Regulatory Constraints**
Vendors must comply with the standards set by the CAB Forum, which can limit flexibility in how certificates are used or managed.

Private Trust Code Signing

Definition: Private Trust Code Signing is used within closed ecosystems, such as internal company networks or consortiums, where software is distributed to a limited audience and does not require public trust.

Use Cases: Suitable for companies that develop software solely for internal use, or for organizations that work within a consortium or partnership where all parties can be trusted to recognize the internal CA.

Pros:

- **Control**
Organizations can manage their own trust hierarchy, meaning they have complete control over who can trust their signed software. This is particularly beneficial for internal applications or software shared with partners.
- **Flexibility**
Private trust certificates can be issued without the same regulatory constraints as public trust certificates, allowing for rapid issuance and revocation.
- **Enhanced Security for Critical Systems**
In cases where keys must remain secure for years, organizations can implement stricter security protocols than those mandated by public trust standards.

Cons:

- **Limited Recognition**
Software signed with private trust certificates may encounter security warnings on end-user systems that do not recognize the internal CA. This can lead to installation issues or a lack of trust from users outside the organization.
- **Increased Responsibility**
Organizations must manage their own Certificate Authority, including establishing security practices and infrastructure, which can be resource-intensive.

Conclusion

Software supply chain security is vital, especially as threats continue to evolve. Digital signing plays a crucial role in establishing and maintaining trust within this ecosystem. As organizations strive to secure their software, they need to consider both public and private trust models for code signing, adapting their strategies based on the specific environments in which their software operates.

Incorporating vigilant digital signing practices not only helps protect the integrity of software but also aligns with the growing requirements for secure deployment. InstallShield enhances this process by providing a customizable signing solution that accommodates various certificate authorities and signing technologies. This flexibility is essential in adapting to the stringent security measures found in modern operating systems, such as Windows 11, which mandates that unsigned binaries be blocked.

By using InstallShield, organizations can implement a dual layer signing process for their binaries and installers, thereby strengthening their defenses against potential threats. The ease of access to custom signing options ensures that any utility capable of leveraging a signing service can be integrated seamlessly, facilitating secure software distribution without compromising security.

As the landscape of software security continues to evolve, embracing innovative digital signing solutions is not just beneficial; it's essential for maintaining trust and ensuring the safe distribution of software products.

InstallShield provides the comprehensive tools you need to adapt to new regulations and ensure trusted installations every time.

NEXT STEPS

Create secure software installers that are both reliable and repeatable with InstallShield by Revenera, the industry's premier installation development tool.

[LEARN MORE >](#)