ITL BULLETIN FOR MAY 2018

PROTECTING SOFTWARE INTEGRITY THROUGH CODE SIGNING

David Cooper, Larry Feldman,[1] and Greg Witte,[1] Editors
Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
U.S. Department of Commerce

Background

From our computers to our smartphones, and even to our automobiles, nearly every aspect of our daily lives depends upon computing code ("code") or software. We need that software to be reliable and trustworthy. Recent security-related incidents highlight the need for a secure software supply chain that protects such software products during the development, build, distribution, and maintenance phases. To ensure integrity, a wide range of software products, including firmware, operating systems, mobile applications, and application container images, must be distributed and updated in secure and automatic ways that prevent forgery and tampering.

An effective and common method of protecting software is to apply a digital signature to the code. When securely implemented, digitally signing code provides both data integrity to prove that the code was not modified, and source authentication to identify who was in control of the code at the time it was signed. Verifying the signature assures the recipient that the code came from the source that signed it, and that it has not been modified in transit. However, code signing is not yet ubiquitous, and it is not always implemented with adequate security controls.

NIST recently published a new white paper, Security Considerations for Code Signing, to assist software developers and product vendors with implementing a code signing system, or with reviewing the security of an existing system. The goal of the paper is to help improve the security of and customer confidence in code authenticity and integrity. The paper will help system integrators and administrators to learn the properties they should expect from a code signing solution, thereby protecting the software supply chain.

The white paper describes features and architectural relationships of typical code signing solutions that are widely deployed today. It defines code signing use cases and identifies some security problems that can arise when applying code signing solutions to those use cases. Finally, it provides recommendations

[1] Larry Feldman and Greg Witte are Guest Researchers from G2, Inc.

for avoiding those problems and resources for more information. Properly applied, these recommendations will help to increase the software supply chain's resistance to attack.

The Basics of Code Signing

The new white paper provides high-level technical details about how the code signing process works. There are multiple roles in the process: developer, signer, and verifier.

The *developer* is the entity responsible for writing, building, and/or submitting the code that will be signed. This entity maintains a secure development environment, including the source code repository, and will submit code to the signer after it has completed the organization's software development and testing processes.

The *signer* is the entity responsible for managing the keys used to sign software. This role may be performed by the same organization that developed or built the software, or by an independent party able to vouch for the source of the code. The signer generates the code signing private/public key pair on a device that is sufficiently protected, as the security of the code signing process relies upon the protection of the private key. In many cases, the signer then provides the public key to a certification authority (CA) through a certificate signing request. The CA will confirm the signer's identity and provide a signed certificate that ties the signer to the provided public key. Anyone can use the public key associated with this certificate to validate the authenticity and integrity of code signed with this key pair. If no CA is used, the public key must instead be distributed using a trusted out-of-band mechanism.

The signer ensures through technical and procedural controls that only authorized code is signed. When code is submitted by developers for signing, the signer verifies their identities and their authority to request a signature. The signer may also take additional steps to verify that the code is trustworthy. Ultimately, two or more trusted agents of the code signing system may be needed to approve the request and generate a digital signature. In some cases, the signed code may also be provided to a time stamp authority to indicate when the code was signed.

The *verifier* is responsible for validating signatures on signed code. The verifier may be a software component provided by the same developer as the signed code (e.g., for a signed firmware update) or a shared component provided by the platform (e.g., the operating system).

Architectural Components

The code signing architecture is composed of a set of logical components that are responsible for different aspects of the code signing process. The code signing/verifying architecture, represented in Figure 1, potentially has four distinct components: the code signing system (CSS), the certification authority (CA), the time stamp authority (TSA), and the verifier(s).

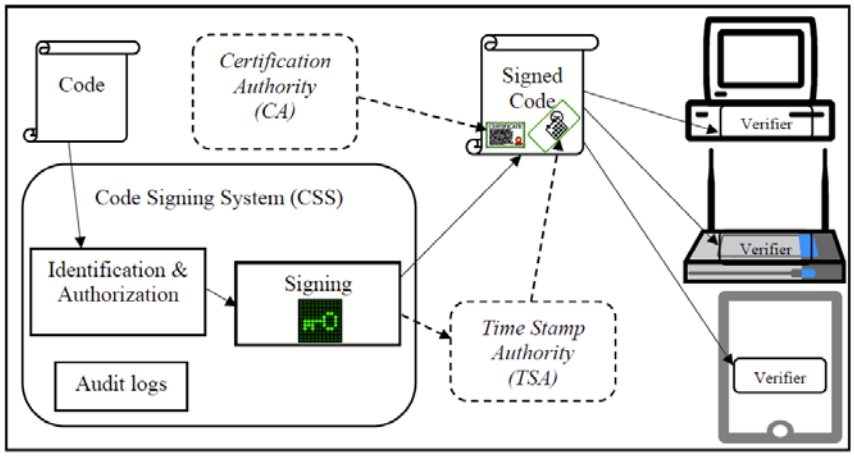

Figure 1: Code Signing Architecture

A critical aspect of managing the verification component is the management of trust anchors that are used to validate signatures, usually by verifying code signing certificates. Trust anchors are data objects, generally public-key certificates, that are installed and securely stored on the verifying platform. The trustworthiness of a trust anchor is based on the method of its installation, storage, and management.

Code Signing Use Cases

The organization establishes a policy that defines the developer(s) authorized to submit code to be signed by the CSS at particular stages of development. The implementation of the system is dependent on the use case for the CSS. The white paper describes four example use cases: firmware signing, driver signing, trusted application stores, and application software signing. These examples are for illustrative purposes; they neither reflect all instantiations of each use case nor are they comprehensive of all code signing use cases.

Threats to the Code Signing System and Recommendations

The white paper enumerates the following potential threats to the CSS:

- Theft of private signing key;
- Issuance of unauthorized code signing certificates;
- Misplaced trust in certificates or keys;
- Signing of unauthorized or malicious code; and
- Use of insecure cryptography.

Based on their threat analysis, the authors of the white paper present numerous high-level recommendations for improving the security of the software supply chain by protecting the code during

the development, build, distribution, and maintenance phases. These recommendations are summarized below:

- Software that hasn't been securely developed isn't part of a secure supply chain. Organizations should ensure that policies and procedures for secure software development, and for subsequent software review and approval, are in place.
- Identification & authentication (I&A) and authorization aspects are important. Ensure that appropriate roles are defined, that an adequate I&A scheme is in place, and that only trusted users can submit (or update) code.
- Organizations should apply safeguards to ensure that keys and certificates used in digital signatures are based upon reliable cryptography, are well protected, and follow appropriate practices such as for review, time stamps, verification, and revocation.

Conclusion

The white paper describes the security considerations for a code signing solution to protect the software supply chain. Code signing provides assurance that the software is authentic and has not been tampered with during the distribution and maintenance phases, and the verifier can validate these properties at runtime. The use cases dictate the deployment model but the core components of the solution include the CA, the TSA, the CSS, and the verifier.

NIST plans to develop further guidance to help organizations with evaluating, deploying, or managing code signing systems. The high-level recommendations described in the white paper are expected to form the basis for more detailed recommended practices for code signing.

ITL Bulletin Publisher: Elizabeth B. Lennon
Information Technology Laboratory
National Institute of Standards and Technology
elizabeth.lennon@nist.gov