ISO/IEC JTC 1/SC 22/WG 23 N 0431

Date: 2012-11-08

ISO/IEC IS 17960

Secretariat: ANSI

Information Technology—Programming languages, their environments and system software interfaces—Code Signing for Source Code

Warning

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: International standard
Document subtype: if applicable
Document stage: (20) development stage
Document language: E

Table of Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2. Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. ISO/IEC IS 17960, which is an International Standard, was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

Introduction

Source code is written and is used in many critical applications. Knowing that the source code being relied upon is the same as that which was used in testing is vital to ensuring the safety and security of a particular application. Given the ease with which source code can be modified, some method of protection of the source code is necessary. Sequestration of the source code is one method, but ensuring protection in that way is impractical and unreliable. Virtual protection through the use of a digital signature offers a practical solution and provides integrity even though the source code may traverse an insecure supply chain.

Modifications to source code are frequently made to correct the software or to adapt it for other purposes. Rarely are the modifications made by the original author. Revision control software facilitates tracking of the software changes, but such tracking can be easily spoofed. Digital code signing provides a means to restrict the ability to spoof by assigning a responsible party to each of the modifications as they are made.

This International Standard specifies the necessary metadata for signing source code in a manner that allows signatures to be shared among applications to ensure the integrity and a means for reversing the application of the signatures to unwrap the source code to previously signed versions. Annex A contains a step by step description of a typical application of source code signing. A bibliography lists documents that were referred to during the preparation of this standard.

Information Technology — Programming Languages — Code Signing for Source Code

1. Scope

This document uses a language-neutral and environment-neutral description to define the methodology needed to support the signing of software source code. It is intended to be used by publishers of software source code and the recipients of their source code. This standard is designed for transfers of source code among disparate entities, not within the same entity.

The following areas are outside the scope of this specification:

- Determination of the trust level of a certificate authority
- Format used to track revisions of source code files
- Digital signing of object or binary code
- System configuration and resource availability

2. Normative References

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 10118–2:2000, *Information technology — Security techniques — Hash-functions — Part 2: Hash-functions using an n-bit block cipher*

ISO/IEC 10118–3:2003, *Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash functions*

ITU-T, "Information Technology - Open Systems Interconnection - The Directory: Public-Key and Attribute Certificate Frameworks", Recommendation X.509, August 2005, http://www.itu.int/rec/T-REC-X.509/en.

3. Terms and Definitions

For the purposes of this document, the following terms and definitions apply.

3.1. certificate Authority

entity that issues digital certificates

3.2. changeset format

set of all changes that are applied to a configuration to derive a new configuration

3.3. digital certificate

electronic block of data received from a trusted certificate authority that certifies the authenticity of the sender's public key, identifies the creator of the sender's public/private key, and contains the sender's public key

3.4. digital signature

data appended to, or a cryptographic transformation of, a data unit that allows the recipient of the data unit to prove the source and integrity of the data unit and protect against forgery

3.5. private key

key of an entity's asymmetric key pair which should only be used by that entity and should not normally be disclosed

3.6. public key

key of an entity's asymmetric key pair which can be made public

3.7. public key encryption

a cryptographic technique which enables users to securely communicate on an insecure public network and reliably verify the identity of a user via digital signatures

3.8. snapshot format

verbatim copy of a file

4. Conformance

An implementation of code signing conforms to this International Standard if it meets the requirements specified in Clause 6.

Clause 5 is informative, providing an overview of the concepts of code signing. Annex A, also informative, provides a possible scenario of usage for the standard specified in Clause 6.

5. Concepts

Code signing is a technique for providing a digital signature for source code and scripts to support a verification of the origin and a verification that the code has not been altered since it was signed.

Code signing can provide several valuable functions such as:

- knowledge of the origin of the code

- confidence that the code has not been accidentally or maliciously altered
- verification of the identity of the responsible party for the code
- accountability for the code
- non-repudiation of the source of the code

Code Signing identifies to customers the responsible party for the code and confirms that it has not been modified since the signature was applied. In traditional software purchases where a buyer can physically touch a package containing software, a buyer can confirm the source of the application and its integrity by examining the packaging. However, most software is now procured via the Internet. Software procurement is not limited to complete applications as code snippets, plug-ins, add-ins, libraries, methods, and drivers are all downloaded over the Internet. Verification of the source of the software is extremely important since the security and integrity of the receiving systems can be compromised by faulty or malicious code. In addition to protecting the security and integrity of the software, code signing provides authentication of the author, publisher or distributor of the code, and protects the brand and the intellectual property of the developer of the software by making applications uniquely identifiable and more difficult to falsify or alter.

When software source code is associated with a publisher's unique signature, distributing software on the Internet is no longer an anonymous activity. Digital signatures ensure accountability, just as a manufacturer's brand name ensures accountability with packaged software. Distributions on the Internet lack this accountability and code signing provides a means to offer the needed accountability. Accountability can be a strong deterrent to the distribution of harmful code. Even though software may be acquired or distributed from an untrusted site or a site that is unfamiliar, the fact that it is signed by a known and trusted entity allows the software to be used with confidence that it has not been changed.

In addition to the valuable functions that code signing offers, this International Standard will specifically facilitate the following capabilities:

- a tracking mechanism to show what has been altered in the code and by whom
- multiple signatures to allow for an audit trail of the signed object
- versioning information
- storage of other meta data about an object

The capability for a tracking mechanism and multiple signatures for one piece of code would be needed in some cases in order to create a digital trail through the origins of the code. Consider a signed piece of code. Someone should be able to modify a portion of the code, even if just one line or even one character, without assuming responsibility for the remainder of the code. A recipient of the code should be able to identify the responsible party for each portion of the code. For instance, a very trustworthy company A produces a driver. Company B modifies company A's driver for a particular use. Company B is not as trusted or has an unknown reputation. The recipient should be able to determine exactly what part of the code originated with company A and what was added or altered by company B so as to be able to concentrate their evaluation on the sections of code that company B either added or altered. This necessitates a means to keep track of the modifications made from one digital signature to the next.

An alternative scenario is software offered by company B that contains software from company A. Company B does not alter company A's software, but incorporates it into a package or suite of software.

It would be useful to a customer to be able to identify the origin of each portion of Company B's software package.

6. Requirements and Guidance

The code signing standard described below is intended to be language and platform independent. To assist in understanding code signing, Annex A provides an overview of the code signing process from a conceptual perspective.

Throughout this standard, *publisher* shall refer to the person or organization that is signing the source code. *Recipient* shall refer to the person or organization that is receiving the signed source code.

6.1. Code Signing

The publisher shall create a private/public key pair and obtain an X.509 compliant digital certificate. The level of trust in the Certification Authority (CA) who issues the X.509 compliant certificate is an important factor in the amount of trust associated with the signed code. The CA should be a trusted party to both the publisher and recipient. Though very important to the execution of a trusted transfer of software from a publisher to a recipient, the establishment or determination of the trust level associated with a certificate authority is beyond the scope of this standard.

Protection of the publisher's private key must be ensured to prevent impersonation by others. The private key part of the publisher's digital certificate must not be compromised from the control of whoever is authorized to sign the code.

For the initial signing of a source code file, a hash shall be generated for the source code. A hash is used since public key signing of large source code files is inefficient. The default hash function shall be the Secure Hash Algorithm-256 (SHA-256). Alternatively, hashing functions that are specified in ISO/IEC 10118-3:2004 or in later revisions of ISO/IEC 10118-3:2004 may be used.

The hash is encrypted with the private key of the publisher to generate a signature for the source code file. The recipient can then use the publisher's public key to validate the signature and verify that the source code file has not been altered.

6.2. Initial Code Signing

The initial signing of a source code file may be in a snapshot format or a changeset format. The changeset format shall be based on an empty file.

6.3. Modifying Previously Signed Code

When modifications are made to a previously signed source code file, sufficient information shall be recorded to allow the source code file and signature of a signed version to be recovered from future

revisions. This allows the series of modifications from one version to the next, which can be thought of as encapsulations, to be reversed one at a time.

6.4. Code Signing Format

This International Standard is not prescriptive as to which format should be used to create or track revisions. Conformance to this International Standard does require that the specifications for the revision control format that is used be freely and conveniently available so that recipients can use the revision control format specification to revert to previously signed versions.

The information contained in each encapsulation shall contain sufficient revision control information in order to recreate the previous version. As such, either a snapshot format or a changeset format may be used. Once an encapsulation is reversed, the recipient must be able to use the digital signature of the encapsulated version to verify its integrity.

Annex A
(*informative*)
Notional Code Signing Process

This annex describes the series of steps in a typical implementation of code signing of source code.

1. **Publisher obtains an X.509 compliant certificate from a global certificate authority**

2. **Publisher develops source code or modifies previously signed source code**

3. **Publisher calculates a hash of the source code file**

   There are two possible cases. The first is signing source code which does not have a signature. The second is signing source code that has been signed previously.

   In either case, a one-way hash of the source code is calculated.

   If the code has not been previously signed, a snapshot format or a changeset format is used to record a version of the source code file.

   If the code has been previously signed, sufficient information is documented to allow the changes to be undone to revert to any of the previous versions, though versions must be undone in the reverse order that they were signed.

4. **Publisher uses their private key to encrypt the hash to produce a digital signature of the source code file**

5. **The source code file and its associated digital signature is transmitted to the recipient**

6. **The recipient produces a one-way hash of the source code file using the same hash algorithm as the publisher**

7. **Using the publisher's public key contained, the recipient decrypts the signed hash**

8. **The recipient compares the two hashes**

   If the signed hash provided by the publisher matches the recipient's hash, the source code file is intact and hasn't been altered since it was digitally signed.

   To verify previously signed versions of the source code, the version signed most recently to the current one is “unwrapped” from the current version. This allows a reconstruction of each previously signed version in sequential order.

Bibliography

1. *Code-Signing Best Practices*, http://msdn.microsoft.com/en-us/windows/hardware/gg487309.aspxJuly 25, 2007

2. *Code Signing Certificate FAQ*, http://www.verisign.com/code-signing/information-center/certificates-faq/index.html, 2011

3. *Code Signing for Developers - An Authenticode How-To*, Tech-Pro.net, http://www.tech-pro.net/code-signing-for-developers.html, 2011.

4. Oliver Goldman, *Code Signing in Adobe AIR*, Dr. Dobb's, September 1, 2008.

5. *How Code Signing Works,* https://www.verisign.com/code-signing/information-center/how-code-signing-works/index.html, 2011.

6. *Introduction to Code Signing*, http://msdn.microsoft.com/en-us/library/ms537361(VS.85).aspx, June 21, 2011.

7. ISO/IEC 10118-3:2004, Information technology -- Security techniques -- Hash-functions -- Part 3: Dedicated hash-functions.

8. ISO/IEC 14750:1999, Information technology -- Open Distributed Processing -- Interface Definition Language, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=25486.

9. ITU-T Recommendation X.509:2008, Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, http://www.itu.int/rec/T-REC-X.509/en.

10. Steve Mansfield-Devine, *A Matter of Trust*, Network Security, Vol 2009, Issue 6, June 2009.

11. Regina Gehne, Chris Jesshope, Jenny Zhang, *Technology Integrated Learning Environment: A Web-based Distance Learning System*, AI-ED'95, 7th World Conference on Artificial Intelligence in Education, 2001.

12. Justin Samuel, Nick Mathewson, Justin Cappos, and Roger Dingledine, *Survivable Key Compromise in Software Update Systems*, The 17th ACM Conference on Computer and Communications Security, 2010.

13. Deb Shinder, *Code Signing: Is it a Security Feature?*, WindowSecurity.com, http://www.windowsecurity.com/articles/Code-Signing.html?printversion , June 9, 2005.