



## **NISCC Technical Note 02/2005**

**Issued 13 December 2005**

### **The Importance of Code Signing**

This Technical Note describes the measures that can be used to ensure the authenticity and integrity of code delivered by software publishers via the Internet.

Reference to any specific commercial product, process or service by trade name, trademark manufacturer, or otherwise, does not constitute or imply its endorsement, recommendation, or favouring by NISCC. The views and opinions of authors expressed within this document shall not be used for advertising or product endorsement purposes.

NISCC shall also accept no responsibility for any errors or omissions contained within this document. In particular, NISCC shall not be liable for any loss or damage whatsoever, arising from the usage of information contained in this document.

**National Infrastructure  
Security Co-ordination Centre**  
PO Box 832  
London, SW1P 1BG

Tel: 0870 487 0748  
Fax: 0870 487 0749  
Email: [enquiries@nisc.gov.uk](mailto:enquiries@nisc.gov.uk)  
Web: [www.nisc.gov.uk](http://www.nisc.gov.uk)

## Key Points

- Secure access to IT systems is critical for organisations
- Security policy should be clear and efficient
- Code signing provides authentication and integrity
- File and Integrity monitors provide alternative solutions
- Consider the impact of Host-Based Intrusion Detection Systems, Firewalls and Anti-Virus Software

## Background

Today information and its secure access are increasingly critical to organisations. With reliance on software applications rising, and security breaches of these systems growing, their protection as well as the confidential and vital information that they control has never been more crucial.

As the Internet becomes more widely used, the traditional methods of distributing files are rapidly fading. Whereas in the past product updates and new components would be distributed by media such as CD-ROM, now they are mostly obtainable from the Internet. Also web pages are increasingly packed with downloadable code to enhance the user experience.

To protect information, users must be sure that they do not introduce any malicious code onto their systems; but how can users trust code that is published on the Internet?

In order to provide this assurance, there are two issues that must be addressed.

- Ensuring authenticity
- Ensuring integrity

This technical note aims to provide a brief description of what is available to aid users to achieve this assurance on IT technologies.

## ISO 15408 (The Common Criteria)

ISO 15408 is an international standard for evaluating secure hardware and software. Vendor products are evaluated under the scheme, which establishes the adequacy of not just their security features, but also of the environment in which these products were developed and used. The standard also requires vendors to address ongoing threats to these systems:

*“The ISO 15408 philosophy is that the threats to security and organisational security policy commitments should be clearly articulated and the proposed security measures be demonstrably sufficient for their intended purpose.*

*Furthermore, measures should be adopted that reduce the likelihood of vulnerabilities, the ability to exercise (i.e. intentionally exploit or unintentionally trigger) a vulnerability, and the extent of the damage that could occur from a vulnerability being exercised. Additionally, measures should be adopted that facilitate the subsequent identification of vulnerabilities and the elimination, mitigation, and/or notification that a vulnerability has been exploited or triggered.”*

- *The ISO 15408 Philosophy (ISO/IEC 15408-3:1999(E))*

ISO 15408 consists of three parts: Part 1 – Introduction and General Model; Part 2 – Security Functional Requirements; and Part 3 – Security Assurance Requirements. It is Part 3 that defines the assurance requirements of the standard.

Part 3 requires all products evaluated above EAL-1 (most evaluated products are EAL-2 to EAL-4) to have special procedures defined to “maintain security during transfer of the [product] to the user, both on initial delivery and as part of subsequent modification.” Where such evaluated products are used, these can be part of an organisation's ongoing security measures to assure the continued integrity of their application software, and thus the information they contain.

Although many techniques can be used here, one robust yet relatively simple technique already commonly used is code signing.

### Code Signing

One does not need to imagine what can happen if users cannot establish the integrity of code that is published on the Internet; they cannot be sure that a malicious person has not created a backdoor, installed Trojan software, added additional user accounts, or performed any other number of possible malicious activities without their knowledge.

For such an attack to succeed the attacker would not even require administrative access on the user's system, they simply need to subvert the

files the user are installing from, for example by breaking into an FTP server, or poisoning the user's DNS cache so that they are redirected to the attackers' site instead of a legitimate site for their system updates.

Code signing lets users verify the origin of the code, and prevents a malicious person from distributing "bad code". In addition to verifying the identity of the publisher, code signing can protect the code from tampering as the digital signature is invalidated if the code is changed. Thus, code signing provides two security protections.

- Authentication of the author, publisher or distributor of the code.
- Integrity of the code itself.

Code signing involves bundling digital signatures along with the original code, which remains unchanged. Most code signing implementations are based on X.509 certificates. In fact this is similar to their use for "secure" web sites using SSL (the familiar https:// prefix) where a web server's certificate can be checked by the user to ensure that they are connected to the "real" web site. With code signing the code's certificate can be used to ensure that they have the "real" code.

However there is a possible problem with using certificates and this is revocation. Some use a Certificate Revocation List (CRL) which has to be updated by the client; if the client has not updated their CRL then it is possible that code will be run which is not actually properly signed. A suggested alternative for the CRL is the "Online Certificate Status Protocol" (OSCP), which provides real time access to the CRL thus removing the need for users to manually update their clients.

Nevertheless, it is relatively simple to resolve these issues, and the checking of code signatures can be fully automated. Organisations already using their own PKI to manage X.509 certificates would probably find it trivial to do so (as is the case with Microsoft's Authenticode system described below). Other organisations would find it relatively simple to make the necessary changes.

Note that organisations can add their own signatures where no vendor ones exist. This could be used to ensure that internal distributions of software are legitimate and have not been tampered with.

## **Java™ Platform**

The Java™ Software Development Kit allows for JAR files to be optionally signed. This is done via the command-line tool "jarsigner". This capability was introduced with JDK 1.1 and greatly expanded with Java 2.

The ability to sign and verify files is an important part of the Java™ Platform's security architecture. Security is controlled by the security policy that is in force at runtime. Users can configure the policy to grant security privileges to applets and to applications. For example, users could grant permission to an applet to perform normally forbidden operations such as reading and writing

local files or running local executable programs. If users have downloaded some code that is signed by a trusted entity, they can use that fact as a criterion in deciding which security permissions to assign to the code.

Once the user (or their browser) has verified that an applet is from a trusted source, then there is the possibility of the applets escaping the restrictions of the sandbox. This ability in the Java™ Platform is a useful utility to allow users to securely sign and verify their own JAR files (i.e. applets).

However the Java engine itself does not fully utilise this security feature upon its own files. Although the Java bundles for the Windows platform are all signed, bundles for other platforms are not. Hence users of platforms other than Windows cannot be guaranteed the integrity of the Java™ Platform on their systems.

## **Microsoft's Authenticode**

Authenticode is Microsoft's code-signing technology and its security model is built on digital signatures and certificates. Support is built into Microsoft operating systems and Internet Explorer, although it is also supported on other platforms. It can be used for signing Java .class files, .cab files, .ocx files, ActiveX controls and all Win32 executables (PE files). Authenticode certificates can be issued by Verisign and other public or private certification authorities.

The CRLs used by Authenticode are downloaded and updated automatically. A signature validation will be failed if the certificate is revoked.

When a user attempts to download code from the Internet, the Authenticode dialog box provides information regarding the publisher, the Certification Authority that issued the certificate and the date the code was signed. The user can then select whether to install the software or not.

While Authenticode does not guarantee bug-free code, the technology is designed to identify the publisher of the code and to assure end users that code (i.e. updates) has not been tampered with before or during the download process.

Authenticode is also used by Microsoft to sign their own updates as well as software for the Xbox. However the security provided by this is minimised by the fact that users are given the option to "Always trust code from Microsoft". This may seem like a valid decision, but by doing so users are actually diminishing the security that is intended by Authenticode.

Another major problem with Authenticode is that most end users would simply click "OK", no matter what the actual signing/verification status of the file is. This also undermines the purpose of Authenticode to protect users from malicious code; although by training users that signed code does not necessarily mean safe code, this problem can be reduced.

## **Linux and UNIX Approach**

Code signing is also used by some Linux/Unix vendors (i.e. Red Hat, SuSE, etc) to sign their updates for their operating systems; this is achieved (by some) by making the default in update software to check the signature and reject the upgrade package if it is not signed properly.

Red Hat additionally also signs all the original installation, on a package-by-package basis, and beta releases of their products. However beta releases are signed with a different key such that a user will have to perform an action to explicitly trust.

Sun also provides signed patches, and the new Solaris 10 operating system has a new feature where digital signatures can be attached to all executable files via the “elfsign” command.

## **Alternative Solutions**

### **File Monitoring**

There are other products available, which although they cannot provide integrity or authenticity for code obtained from the Internet, can provide assurance that files on a system have not been modified without the user’s knowledge.

These are known as File Monitors or Integrity Monitors. Such programs monitor the system’s files and notify the user when any are modified, deleted or added.

This is particularly important when most systems are now connected to the Internet. By providing this assurance, users can rest in the knowledge that their online activities have not triggered any unwanted changes to their system.

Although there are many varieties of file monitors, most perform their protection in essentially the same way. These programs would normally scan the protected system for important system files, and compute a hash value for every important file and store this in a database. At scheduled intervals, the programs will scan the list of monitored files, compute a hash value again and test the current value against the stored value to determine if the file has been modified or not. If the program detects a change, it will notify the system administrator, and some may also log the occurrence.

However the effectiveness of the program depends on the security soundness of the underlying system that it is installed on. It must also be noted that file monitoring is more reactive rather than proactive.

## Host-Based Intrusion Detection

These are very much like File Monitors, but with the added functionality of monitoring communication traffic on the system as well.

## Host-Based Intrusion Prevention

As with Host-Based Intrusion Detection Systems (HIDS), Host-Based Intrusion Prevention Systems (HIPS) also monitor both the traffic entering and leaving the system, and the files residing on the system. However where HIDS only monitors and notifies the user of any abnormal behaviour, HIPS will detect and prevent malicious files from executing on the user's system.

## Firewalls

There are some personal firewalls that have HIDS or HIPS built into them; this allows the firewall to monitor files and traffic as well as enforcing the security policy of the system.

## Anti-Virus Software

Anti-virus software consists of computer programs that attempt to identify, prevent and eliminate computer viruses and other malicious software from affecting the system. However there are some vendors who now also incorporate an element of integrity checking into their products, although these should by no means replace a full File Monitoring program.

## Summary

Code signing was designed to prove the integrity of code, and to provide an authentication mechanism for software publishers that is based on digital signatures backed by digital certificates (i.e. X.509 digital certificates). Signed code is not however necessarily safe code, as code signing was never intended for this purpose. Nevertheless, it is a valuable weapon against the spread of viruses and other malicious code.

Code signing was never designed to certify that a piece of code was securely written. But by incorporating digital signatures, it does provide accountability – something that is often missing in a world of freely, and sometimes anonymously distributed code.

## Notes

[1] Code Signing: Is it a Security Feature?  
<http://www.windowsecurity.com/articles/Code-Signing.html>

[2] Beyond the Sandbox: Signed Code and Java 2  
<http://www.securingsjava.com/chapter-three/>

[3] Security with ActiveX Authenticode  
<http://docs.rinet.ru/ZhPP/ch23.htm>

[4] File integrity checkers  
<http://www.windowsecurity.com/software/File-integrity-checkers/>

[5] Intrusion Detection FAQ: What is host-based intrusion detection?  
[http://www.sans.org/resources/idfaq/host\\_based.php](http://www.sans.org/resources/idfaq/host_based.php)

[6] International Organization for Standardization (ISO)  
<http://www.iso.org/iso/en/ISOOnline.frontpage>

## **Acknowledgements**

NISCC would like to thank Steve Allen from Senselect Ltd. for his assistance with the writing of this document.

NISCC would also like to thank vendors for their input into this document.