Code signing systems

How to manage digital certificates, Software Publishing Certificates and private keys for code signing

Saqib Ali

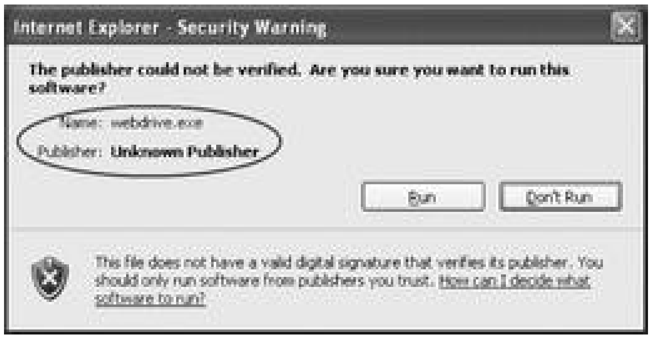
This article looks at the management of the private key for the Software Publishing Certificate (SPC). SPCs are used to digitally sign binaries that are produced by software development vendors. Digitally signing executables proves the identity of the software vendor and guarantees that the code has not been altered or corrupted since it was created and signed. Signing the code requires access to the SPC and the Private Key (PVK) associated with the SPC.

Digitally signing executables proves the identity of the software vendor and guarantees that the code has not been altered or corrupted since it was created and signed

Background

In cryptography, key management includes secure generation, distribution, and storage of keys. Appropriate and successful key management is critical to the secure use of every crypto-system without exception. Secure methods of key management are extremely important. Once a key is randomly generated, it must remain secret to avoid misuse (such as impersonation). It is, in actual practice, the most difficult aspect of cryptography, for it involves system policy, user training, organizational and departmental interactions in many cases, and coordination between end users,

Windows XP SP2 will produce this warning message for unsigned binaries. The user can NOT verify the authenticity of the code



Internet Explorer - Security Warning

The publisher could not be verified. Are you sure you want to run this software?

Name: webdrive.exe

Publisher: Unknown Publisher

This file does not have a valid digital signature that verifies its publisher. You should only run software from publishers you trust. How can I decide what software to run?

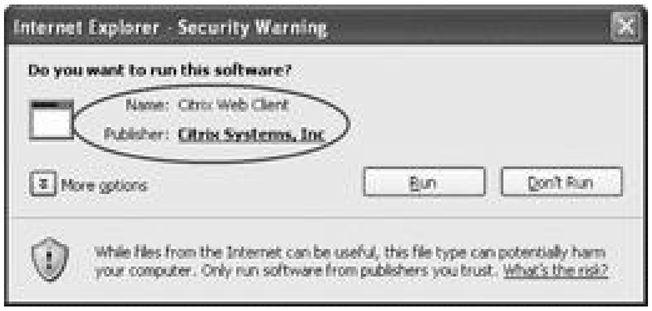
etc. Most attacks on public-key systems will probably be aimed at the key management level, rather than at the cryptographic algorithm itself.

Many of these concerns are not limited to cryptographic engineering and are outside a strictly cryptographic domain. The responsibility of the proper key management falls on the upper management in an organization of any size.

Users must be able to store their private keys securely, so no intruder can obtain them, yet the keys must be readily accessible for legitimate use.

There are many solutions available for proper key management of private keys owned by single individuals. Verisign, Entrust, RSA, and Microsoft's Active Directory all provide a good mechanism for managing keys owned by individuals.

For signed binaries the source (vendor name) is displayed. The user knows that the code is authentic, and has not been tampered with during the transmission


Internet Explorer - Security Warning

Do you want to run this software?

Name: Citrix Web Client
Publisher: Citrix Systems, Inc

While files from the Internet can be useful, this file type can potentially harm your computer. Only run software from publishers you trust. What's the risk?

However, the management of private keys owned by groups or organizations is an issue that lacks proper tools and guidelines. Examples of these types of private keys include:

- The private key for the SSL certificates owned by groups or organizations
- The private key for SPC owned by a software development vendor
- The private key for the root CA (certification authority)

This article looks at the management of the private key for the SPC.
A digital signature informs the user:

1. Of the true identity of the publisher.
2. Of a place to find out more about the binaries.

Code signing digital certificates can be purchased from Verisign (`http://www.verisign.com/products-services/security-services/code-signing/index.html`)

So what's the problem?

In any software development firm more than one person needs to be able to sign the latest build, and release for download. However, allowing multiple developer access to the private key kills the security of the key. Every time a secret key is shared, the confidentiality provided by that key gets halved. Some questions that come to mind:

1. What are the best practices for managing Code Signing Digital IDs and private keys?
2. What can be done to secure the private key for code signing?
3. Who should have the possession of the private key? Multiple people or just the project manager?
4. What key escrow (recovery) techniques can be used, if the private key holder is not available?
5. Who should be allowed to digitally sign the build?

If one person is responsible for signing all binaries, that person becomes the single point-of-failure. So it is recommended to give several people the ability to sign builds. However, this needs to be done in a way that several developers DO NOT end up with the private key for the SPC.

A secret sharing scheme allows you to distribute a secret (such as a private key) over some number of people such that a specified number of people (the threshold) must work together to recover the secret

Code signing system: option one

One option is to use a code signing system (let's call it CSS1), which uses a shared secret scheme (Shamir, Blakeley or Trivial). A secret sharing scheme allows you to distribute a secret (such as a private key) over some number of people such that a specified number of people (the threshold) must work together to recover the secret. For more information on secret sharing, have a look at this Wikipedia entry (`http://en.wikipedia.org/wiki/Secret_sharing`). In its simplest form, this option would involve giving out "parts" of keys to each developer, and the CSS1 would have the knowledge on how to reconstruct the private key. If at least three developers are able to provide parts of the whole key, the CSS1 would go ahead reconstruct the key and sign the binary. The developers will never get to see the whole private key. This way, you can avoid a single person being able to sign, while at the same time making sure that no single person is critical for the signing. To further secure the system Shamir's or Blakeley's scheme can be used.
Secret sharing is a good start, but it doesn't get you all the way there. Suppose its time to sign a new build: then some

parties reconstruct the secret key, and *poof* all of a sudden CSS1 knows the secret key, and you are back to a single point (or even multiple points) of (security) failure. Now you are dependent on the security implemented in the CSS1 to safe guard the privacy of the key. An attacker might be able to make use of the window in time between the key being reconstructed and the key being destroyed, to retrieve the key from the CSS1.

Code signing system: option two

Another low-tech option is to use an isolated computer dedicated for code signing. Let's call this system CSS2. CSS2 should have no hard drive or network connection of any type. The operating system and the code signing tools, including the PVK and SPC, should reside on a read-only DVD. The following are security considerations:

1. The operating system contained on the DVD should be very minimal. It should restrict any file copying or displaying functionality. It should also restrict any access to the console. This way an attacker cannot copy the PVK and the SPC from the DVD.
2. The DVD should be encrypted, and the decryption key should be embedded into CSS2. This way any copies of the DVD will be useless, outside CSS2.
3. The DVD should be placed in a safe box, which requires three more or keys to open it.

At the time of signing, at least 3 developers must get together to perform the code signing. They must use their keys to open the safe box, retrieve the DVD, boot the computer using the DVD, sign their code, and place the DVD back into the safe.

Inevitably, there is still the potential for collusion, i.e. three or more developers with malicious intent can work together to create and sign malicious software

This system will prevent any attacker from gaining access to the PVK file. Even if an attacker gets hold of a copy of the DVD, it will be useless since it is encrypted. Also, since there is no file copy or display mechanism on CSS2, the PVK file can not copied. Inevitably, there is still the potential for collusion, i.e. three or more developers with malicious intent can work together to create and sign malicious software.

Conclusion

Code signing can provide a very good security tool for verifying the authenticity of a released binary, and also guarantee that the code was not tampered with during transmission. However, management of the certificate and private key for code signing is very critical. If these private keys and certificates are misused, both the customer and software vendor can be adversely affected. The customer might end up with malicious software on their computer, while the vendor may lose money and reputation. If proper key management techniques are utilized, any misuse can be avoided.

Copyright information

About the author

Saqib Ali is Senior Systems Engineer at Seagate Technology. He also manages the free software XML Validator and Transformer available at http://validate.sf.net (`http://validate.sf.net`)