CÓMO COMPROBAR LA INTEGRIDAD DE LOS FICHEROS

Comprobar la integridad de un fichero consiste en averiguar si algún dato del archivo ha variado desde su creación. El archivo puede haber sido modificado por error, por cortes en la comunicación o, en el peor de los casos, porque un atacante haya inyectado código malicioso.

En entornos de código abierto es posible contrastar y compilar el código fuente del programa, revisándolo para comprobar que no contiene código no deseado. De esta forma se puede garantizar que el software no ocasionará ningún daño, aunque por el volumen de algunos programas (y por tanto de su código fuente), no es un proceso práctico en la realidad.

Aun así, lo habitual, tanto para Windows como para Linux es descargar programas ya compilados en binario de los que no se dispone del código fuente. Es posible analizarlos, descompilarlos, desemsamblarlos, estudiarlos en entornos de prueba, etc, pero resulta ineficiente y complejo conocer qué hace realmente un programa sin disponer del código fuente. Además implica muchos conocimientos relativos al análisis de comportamiento.

Ante esta situación, ¿quién puede garantizar que los ficheros que son descargados no contienen algún tipo de código no deseado? ¿Cómo se puede estar seguro de que lo que se descarga el usuario es lo que el autor legítimo ha colgado en su web y que no ha sido alterado por un tercero? Se repasan a continuación los conceptos básicos sobre la integridad y origen de ficheros y métodos prácticos para comprobarlos.

I Introducción

Cuando un desarrollador hace público un programa, suele almacenarlo en un servidor para que sea descargado. Este servidor no siempre está bajo el control del autor del programa o es administrado por la misma persona que ha realizado la herramienta, por tanto, puede ser vulnerable a ataques. Incluso si el programador aloja el software en un servidor administrado por él, es posible que pueda ser comprometido sin su conocimiento y otro atacante tenga la posibilidad de modificar o sustituir el programa que pone a disposición de todos los usuarios.

Esta situación en la que un atacante se hace con un servidor FTP o web y tiene la posibilidad de modificar un programa que más tarde será descargado por otros, es lo que se conoce como una troyanización, y puede suponer un riesgo a nivel global si ocurre sobre aplicaciones populares. Este tipo de ataque se ha dado en muchas ocasiones. Sin las herramientas y precauciones adecuadas, nada garantiza que un programa esté libre de malware, ni siquiera el hecho de descargarlo de una página oficial del fabricante.

En general, los axiomas que hay tener en cuenta siempre que se implemente un control de seguridad o se considere un aspecto de la seguridad, son los siguientes:

- Confidencialidad. Nadie debe poder acceder a los datos privados de un usuario, excepto el propio usuario y las personas autorizadas.
- Disponibilidad. El sistema y los datos tienen que ser accesibles por los usuarios autorizados en todo momento.
- Integridad. Nadie puede cambiar, recortar o falsificar ilegítimamente los datos.

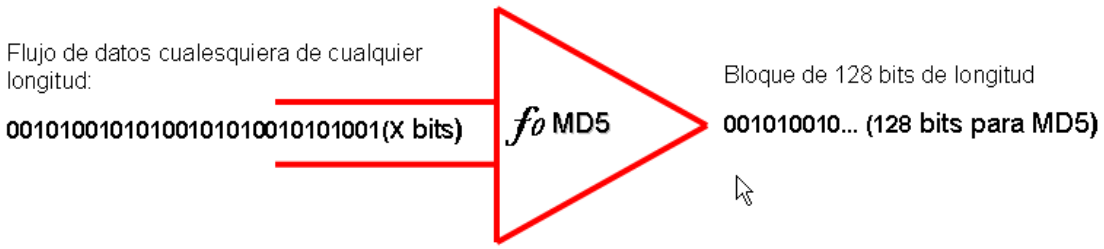
Comprobar la integridad de los ficheros es la tarea que permite saber a ciencia cierta si éstos han sido modificados desde su creación. Esto es lo que se estudiará en lo siguientes epígrafes.

II Funciones *hash* criptográficas

Las funciones *hash* son estructuras de datos muy conocidas en matemáticas y ciencias de la computación y se encuentran ligadas muy estrechamente con la criptografía en general y la integridad de los datos en particular.

Las funciones *hash* criptográficas convierten un mensaje de cualquier tamaño en un mensaje de una longitud constante. Lo que se obtiene al aplicar una función *hash* criptográfica a un mensaje (flujo de datos, o más usualmente, un archivo) se llama resumen criptográfico, huella digital o *message digest*. Es decir, a partir de un número indeterminado de bits, siempre se obtiene un número constante y diferente que identifica de forma unívoca a ese flujo de datos.

Ilustración 1: Representación de la función *hash* MD5



Fuente: INTECO

En seguridad de la información, se utilizan funciones *hash* criptográficas en procesos de autenticación, o de comprobación de integridad de datos.

Características de las funciones *hash*

- Las funciones deben ser conocidas, públicas y su código abierto.
- Existe un número infinito de conjunto de datos que pueden procesar estas funciones. Sin embargo, el resultado es limitado (siempre un tamaño fijo).

> Por ejemplo, en el caso de la función *hash* MD5 el número de posibles resultados es de 2^{128}. Por tanto, no existe una relación uno a uno, y a nivel teórico existirían dos flujos de datos diferentes con un mismo resultado *hash*. Esto se llama colisión. Las funciones *hash* más conocidas y usadas están diseñadas para que, en el uso real, la probabilidad de una colisión sea muy baja.

- La función no puede ser invertible, es decir, no se puede encontrar una función o un algoritmo que sea capaz de computar el mensaje original a partir del resumen criptográfico.
- No debe ser posible generar un mensaje con un resumen criptográfico determinado a no ser que se utilice un método de fuerza bruta, es decir, probando con mensajes arbitrarios hasta obtener el resumen criptográfico deseado.
- No debe existir un método (que no sea de fuerza bruta) para producir un colisión, es decir, dos mensajes con el mismo resumen.
- Si se cambia un bit del mensaje, tiene que cambiar el resumen criptográfico en un 50%.
- Siempre que se aplique la función a un mensaje, se debe obtener el mismo resultado.

Aplicaciones

Con estas premisas, las aplicaciones de las funciones *hash* son claras:

- Protección de contraseñas. Estas funciones permiten almacenar un resumen criptográfico de las contraseñas en vez de las contraseñas en texto claro. Así, en lugar de comparar las contraseñas en texto claro, se calcula su *hash* con una función y se comparan los resultados. De esta manera, el sistema no tiene por qué almacenar el texto claro en ningún momento para comprobar si alguien conoce una contraseña almacenada.
- Comprobación de integridad. Si se calcula un *hash* de dos flujos de datos y dan un mismo resultado, se puede confirmar que los flujos de datos son idénticos.

- Garantizar la integridad de un flujo de datos (archivos, por ejemplo) a diferentes niveles:

    - Al descargar u obtener un fichero por cualquier medio, se puede comprobar que se trata del original o que no tiene defectos si el proveedor proporciona un resumen criptográfico para comparar. Si hubiese cambiado un solo bit del archivo, el resumen sería muy distinto.

    - Se puede comprobar si se han producido cambios no controlados en los datos de un sistema de almacenamiento calculando y comprobando de forma periódica los resúmenes criptográficos de los datos.

    - Otra utilidad es la de ahorro de "coste" computacional en criptografía. Por ejemplo en la firma electrónica (de la que se hablará más adelante). Firmar un mensaje es computacionalmente costoso. Sin embargo, que un *hash* represente a ese mensaje es mucho más liviano computacionalmente. Por tanto, se suele firmar un *hash* (que representa unívocamente a un flujo de datos o mensaje) en lugar de al mensaje en sí.

Algoritmos más utilizados

Los algoritmos más utilizados para calcular el *hash* son:

- MD5. Ante la entrada de cualquier flujo de datos, devuelve un bloque de 128 bits. En 2006 se publicó un método capaz de encontrar colisiones en unos minutos y por tanto, aunque muy usado, no se considera totalmente seguro hoy día.

- SHA256 (y sus sucesores: SHA512, por ejemplo). Ante la entrada de cualquier flujo de datos, devuelve un bloque de 256 bits. Al aumentar los bits de salida (hasta 2^{256} frente a 2^{128} del MD5), la posibilidad de colisión es menor y por tanto es más seguro. Se utiliza en los principales protocolos de cifrado: SSL, SSH, PGP o IPSec. Los métodos para encontrar colisiones, aunque existen, en SHA no tienen suficiente potencia como para poder proporcionar un ataque práctico, por tanto se considera relativamente seguro hoy día.

III Firma digital (funciones *hash* más criptografía asimétrica)

En criptografía asimétrica existen dos claves distintas que se complementan para cifrar y descifrar los mensajes. Funcionalmente tienen las mismas capacidades, pero una de ellas permanece secreta y la otra se hace pública.

En criptografía asimétrica se cifra con la clave pública y se descifra con la privada. Análogamente, un texto se puede cifrar mediante la clave privada y descifrar mediante la pública. Este es el fundamento de la firma digital, que se basa en la firma del hash de un

mensaje (puesto que, como se ha mencionado anteriormente, firmar todo un mensaje es computacionalmente más costoso). Dado que el *hash* es siempre de un tamaño fijo y menor que el mensaje original, pero lo representa unívocamente, firmar el *hash* es equivalente a firmar un mensaje o archivo cualquiera.

El usuario A desea enviar un mensaje al usuario B y quiere tener la certeza de que un usuario C (atacante) no modifica su contenido, aunque le es indiferente si lo lee (en el mundo físico, sería equivalente a enviar una postal pero asegurándose de que nadie modifica el mensaje original).

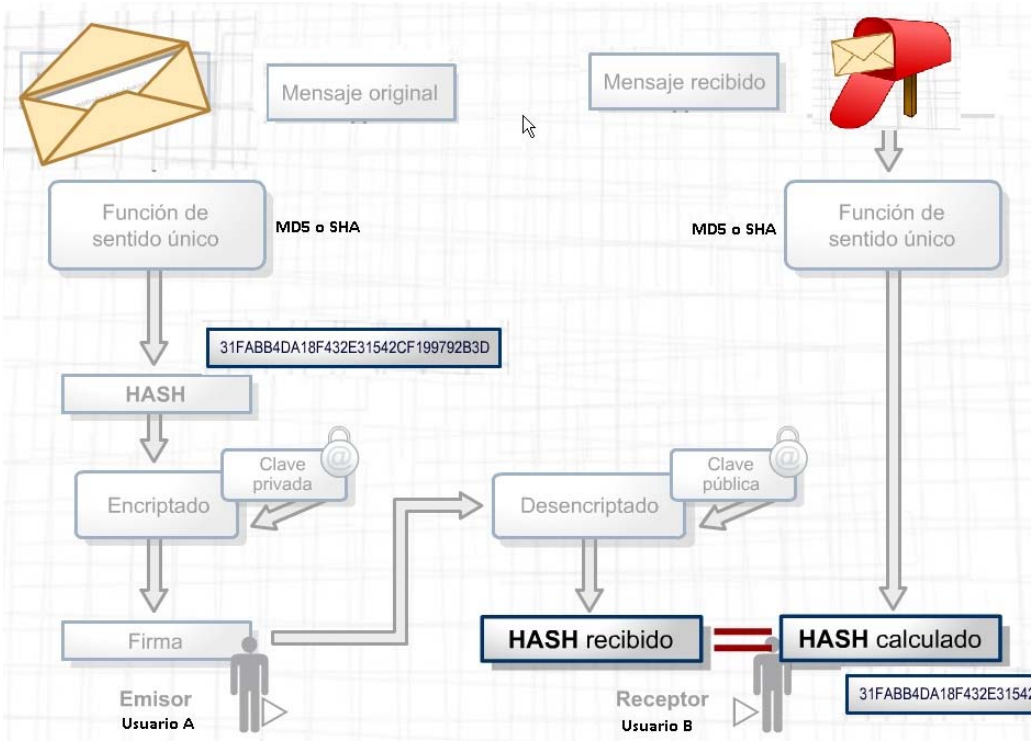
El usuario A calcula un resumen criptográfico del mensaje (esto es, el *hash* mediante un algoritmo seguro que ha acordado con el usuario B, podría ser MD5 o SHA, por ejemplo) y lo cifra con su clave privada. Envía al usuario B los datos en claro y el *hash* cifrado a su vez.

El usuario B utiliza la clave pública de usuario A para descifrar el resumen criptográfico.

Por otra parte calcula el resumen criptográfico del mensaje en claro que le ha llegado.

Al comparar los dos resúmenes puede deducir si un hipotético usuario C ha manipulado o no el contenido de la carta.

Ilustración 2: Esquema de uso de firma digital y funciones *hash*



Fuente: Hispasec

Si el usuario C manipula la carta, los dos resúmenes criptográficos no coinciden. Para llevar a cabo el ataque, por tanto, el usuario C tiene que manipular también el resumen criptográfico que ha enviado el usuario A. Pero si los algoritmos utilizados (el de cifrado asimétrico y el de *hash*) son fuertes, el usuario C no puede falsificar el resumen. Ni puede cifrar el resumen del mensaje falsificado ni puede construir un mensaje cuyo resumen sea exactamente el del original.

Si la firma digital está asociada al usuario A a través de una autoridad de certificación, no solo se garantiza la integridad sino que se garantiza que A y sólo A, ha podido realizar esa acción. Esto es lo que se conoce como un certificado digital.

IV Métodos prácticos de comprobación de integridad

A continuación se estudia de forma práctica lo expuesto anteriormente y se muestran algunos métodos de comprobación de la integridad de los archivos. A modo de resumen, los métodos que se exponen son los que ilustra la Tabla 1:

Tabla 1: Comparativa de métodos de comprobación de identidad

Método	Ventajas
Hash	Garantía de que el archivo no ha cambiado
Firma digital	Garantía de que el archivo no ha cambiado y proviene de una firma digital concreta, aunque nada garantiza que pertenezca a esa persona física determinada.
Certificado	Garantía de que el archivo no ha cambiado y proviene de una firma concreta, garantizada su identidad por una tercera parte confiable (Autoridad Certificadora)
Virustotal	Ofrece una idea (no definitiva) sobre si el archivo contiene malware o no.

Fuente: INTECO

Hash

Una vez que se conocen las características de las funciones *hash*, se pueden utilizar como método para garantizar la integridad de los archivos que se descargan. Así, por ejemplo, un programador que hace público un archivo ejecutable de un programa, puede calcular su *hash* (bien MD5, bien SHA) y publicarlo también. De esta forma, se sabe que cualquier otro archivo que no sea exactamente ese que ha publicado el autor, tendrá un resultado *hash* diferente.

Para calcular el *hash* MD5 en Windows se pueden utilizar diferentes herramientas gratuitas:

- md5sum.exe, disponible desde http://www.etree.org/cgi-bin/counter.cgi/software/md5sum.exe
- md5.exe disponible desde http://www.fourmilab.ch/md5/
- fsum.exe disponible desde http://www.slavasoft.com/fsum/

También, Microsoft facilita una herramienta por línea de comandos. Está disponible desde http://support.microsoft.com/kb/841290/en-us. Se llama fciv, (*File Checksum Integrity Verifier*). Permite además del cálculo del MD5, el cálculo del *hash* más seguro SHA.

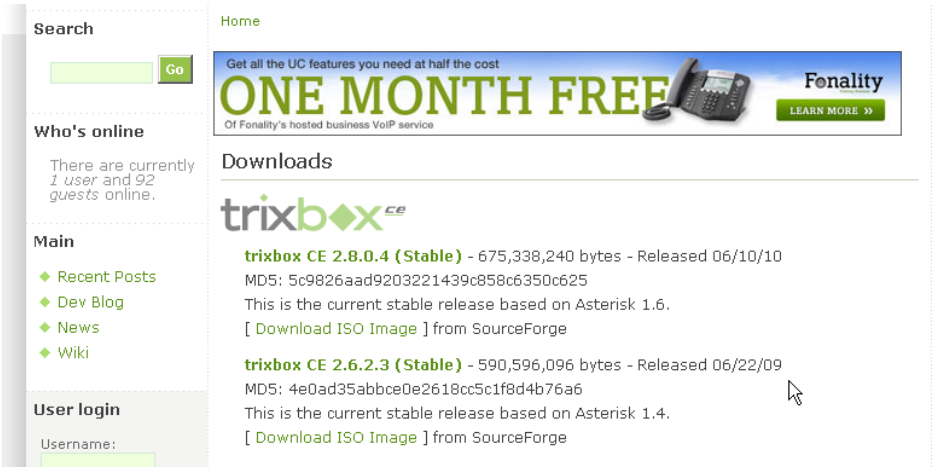
Con las herramientas adecuadas, se puede comprobar la integridad de los ficheros en los que el autor ha hecho público el *hash* oficial. Por ejemplo, se pueden observar diferentes páginas en los que se hace público el *hash* del fichero junto con el propio archivo de descarga.

Ilustración 3: Ejemplo de página que publica los *hashes* junto con el programa para descarga

Fuente: INTECO

Siguiendo el ejemplo mostrado en la Ilustración 4, una vez descargado el archivo trixbox-2.8.0.4.iso desde "Download ISO Image", se puede comprobar la integridad de éste de la siguiente forma con la herramienta md5sum:

Ilustración 4: Ejemplo de página que publica los hashes junto con el programa para descarga



Fuente: INTECO

c:\>md5sum trixbox-2.8.0.4.iso

y devolvería el siguiente resultado:

5c9826aad9203221439c858c6350c625 * trixbox-2.8.0.4.iso

Como los dos *hashes* coinciden (el publicado en la página y el resultado de la operación de cálculo que se ha realizado) podemos decir que el archivo mantiene su integridad.

Este método no es infalible. Si un atacante consigue tener acceso a un servidor web como para modificar un fichero que puede ser descargado, es posible que también pueda modificar la página web y cambiar el *hash* para que coincida con el del programa troyanizado. Para evitarlo, se debe comprobar el *hash* en otros servidores diferentes (espejos) desde donde pueda realizarse la descarga. El cálculo del *hash* también sirve no sólo para descargas sino también para programas que llegan desde otras fuentes (un CD, por ejemplo). Obviamente no es un sistema infalible pero permite elevar el nivel de confianza del software que se utiliza, aumentando las posibilidades de evitar o detectar que el archivo se encuentra modificado.

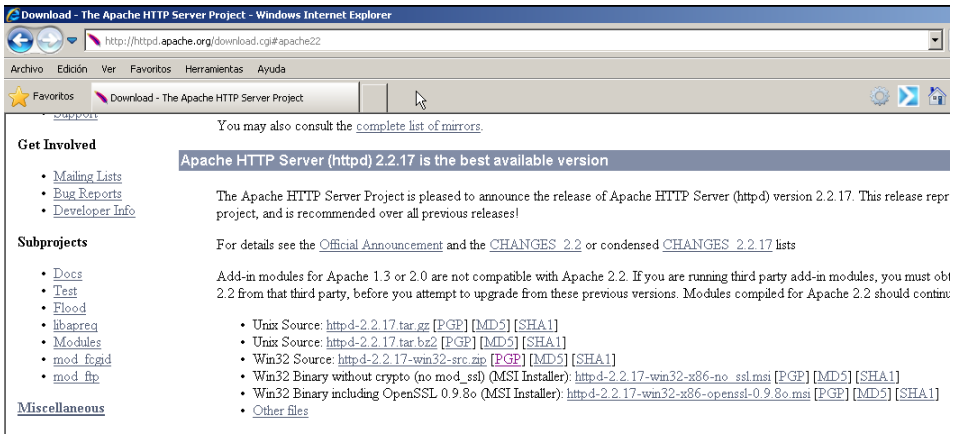
Otra desventaja es que muchos fabricantes no publican el *hash* MD5 ni ningún otro método para comprobar que el software que se descarga es fiable más que la descarga desde una página oficial.

Firma digital

Además de la integridad, gracias a la criptografía simétrica es posible comprobar no sólo que un archivo no ha sido alterado, sino que ha sido realizado por la persona u organización que afirma haberlo creado. Esto se consigue a través de las firmas criptográficas que acompañan a ciertos archivos y, como se ha mencionado, corresponden a la firma del *hash* del archivo.

Las firmas suelen ser archivos con extensión SIG o ASC que resultan de firmar criptográficamente con la clave privada del autor el *hash* de un fichero. Si posteriormente se comprueba, a través de la clave pública, que el fichero firmado concuerda con la firma, es que se está ante un fichero realmente creado por quien dice haberlo hecho, y no modificado desde que se firmó. Una vez más, esto no garantiza en ningún modo las intenciones del archivo, sólo su origen.

Ilustración 5: Ejemplo de página que publica las firmas PGP de los ficheros



Fuente: INTECO

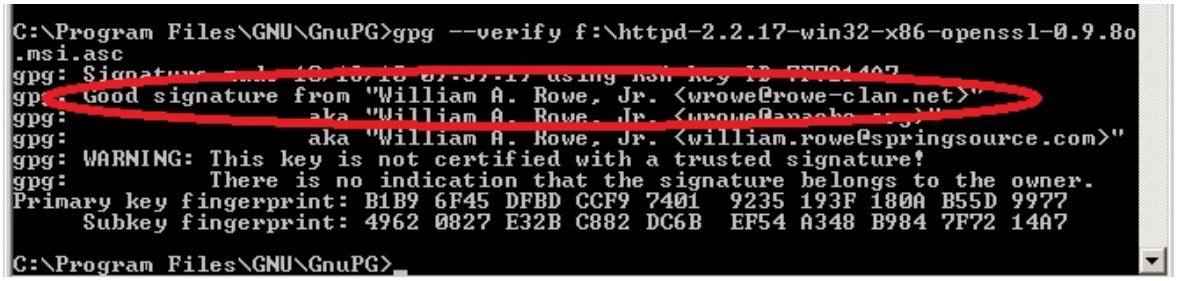
Para comprobar esta firma, entre otras opciones, es posible descargar las herramientas GnuPG (libres y gratuitas). Normalmente el archivo de firma tiene el mismo nombre que el fichero original, sólo que a la firma se le añade por convención la extensión .SIG o ASC. También es necesario disponer de la clave pública del firmante. Esta puede ser encontrada en el mismo servidor de descarga del fichero o a través de los repositorios mundiales de claves públicas. Por ejemplo, para comprobar la firma del servidor Apache para Windows, se debe importar primer el fichero de claves públicas que el propio servidor aloja:

gpg --import KEYS

y después verificar el fichero en cuestión con el comando:

gpg --verify d:\httpd-2.2.3-win32-src.zip.asc

Ilustración 6: Resultado de la comprobación de firma sobre un archivo



Fuente: INTECO

El procedimiento puede variar, pero básicamente consiste en importar y verificar firmas, una vez se dispone de:

a) La firma pública.

b) El archivo a verificar.

c) La firma del archivo.

Como se indica en la Ilustración 6 (*This key is not certified with a trusted signature!, There is no indication that the signature belongs to the owner*), nada asegura que la firma pertenezca realmente al dueño, puesto que no está firmada por una entidad de confianza. Aquí es donde entran en juego los certificados. Con ellos, se garantiza la integridad del archivo (que no haya sido alterado), que ha sido creado por un usuario con una firma concreta y además, que esa firma corresponde físicamente a él (una tercera parte confiable lo garantiza).

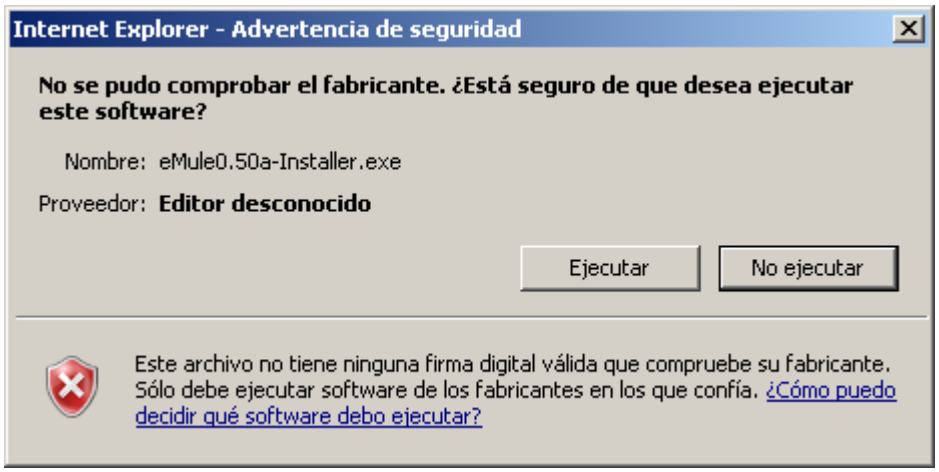
Certificados

Un certificado consiste en la asociación entre una entidad física y una firma, realizado por una entidad confiable.

Es lo equivalente a un DNI en el mundo físico. Un certificado digital certifica que una firma criptográfica pertenece a una persona, y una entidad lo ha comprobado: le ha pedido a esa persona sus datos y pruebas fehacientes de que la firma le pertenece. En ese momento, ha firmado a su vez el certificado, que no es más que ponerle un "sello" de confianza: la firma pertenece a su autor.

En Windows, el explorador de sistema realiza un buen trabajo a la hora de comprobar certificados. Cada vez que se descarga un archivo firmado, muestra un cuadro de diálogo para poder observar detenidamente el certificado, y aceptar la ejecución o no en caso de duda.

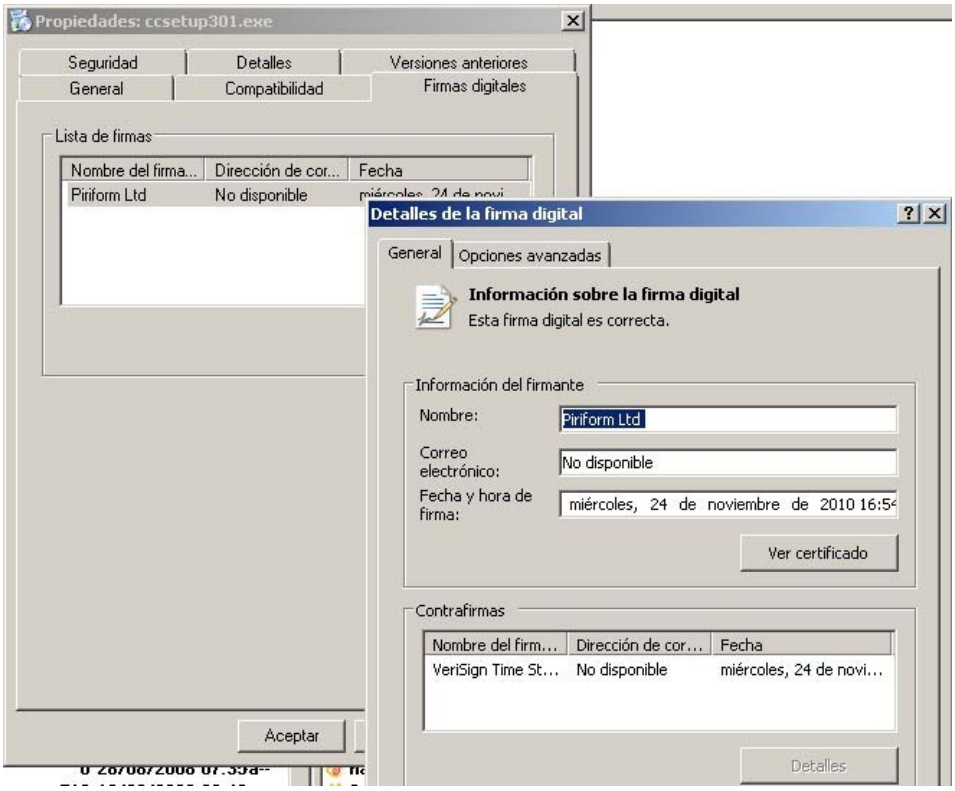
Ilustración 7: Ejecución en Windows de un software no firmado (editor desconocido)



Fuente: INTECO

Si un archivo ya ha sido descargado, también es posible observar si está firmado o no a través de las propiedades. En el ejemplo la Ilustración 8, el certificado de la empresa *Piriform* está "certificado" por *VeriSign*, que ha comprobado que esa firma pertenece realmente a esa empresa.

Ilustración 8: Examinar desde Windows el certificado de un programa



Fuente: INTECO

Los detalles del certificado indican si se puede confiar en el archivo. Si la ruta de firmas culmina en una entidad de confianza (en este caso *VeriSign*), entonces es más que probable que el archivo no sólo no haya sido modificado desde que se creó, sino que además viene de la empresa que dice venir.

Esta comprobación también se puede realizar a través de línea de comando, con la utilidad de Microsoft sigcheck http://www.microsoft.com/technet/sysinternals/FileAndDisk/Sigcheck.mspx que permite comprobar qué ficheros en una ruta concreta no están firmados.

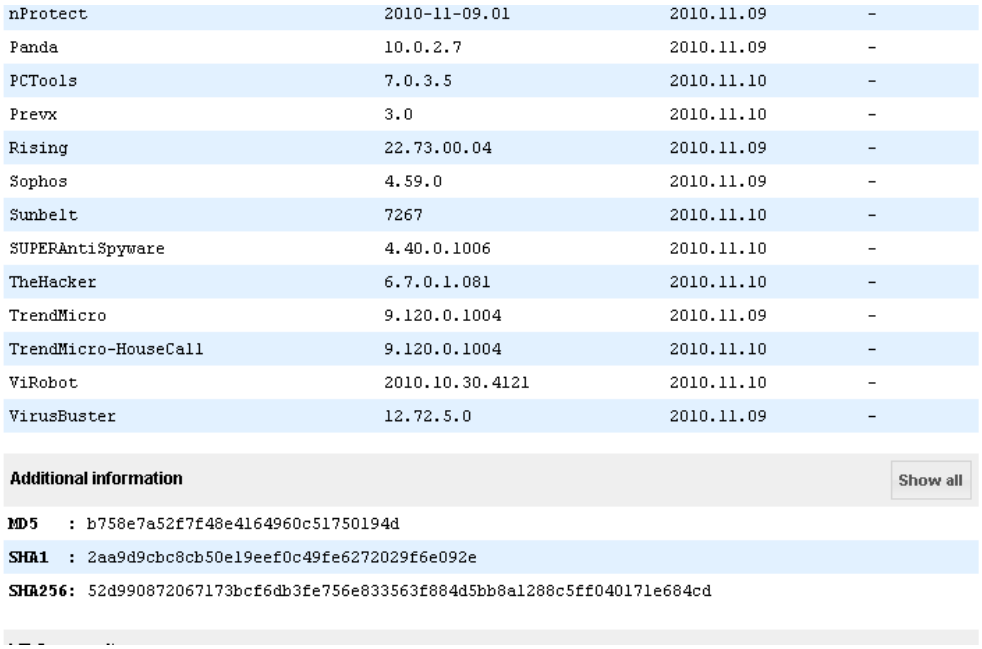
Virustotal

En cualquier caso, además, existe la posibilidad de analizar concienzudamente el archivo en busca de virus o troyanos.

Es aconsejable utilizar sistemas de análisis múltiple como virustotal.com para asegurar al menos que algunas casas antivirus reconocen o no el programa como peligroso. Virustotal.com ofrece la posibilidad de conocer la opinión de múltiples motores antivirus sobre un programa o archivo concreto.

Virustotal.com además, ofrece información de los *hashes*, como se indica en Ilustración 9.

Ilustración 9: Resultado del envío de un archivo a VirusTotal.com



nProtect	2010-11-09.01	2010.11.09	-
Panda	10.0.2.7	2010.11.09	-
PCTools	7.0.3.5	2010.11.10	-
Prevx	3.0	2010.11.10	-
Rising	22.73.00.04	2010.11.09	-
Sophos	4.59.0	2010.11.09	-
Sunbelt	7267	2010.11.10	-
SUPERAntiSpyware	4.40.0.1006	2010.11.10	-
TheHacker	6.7.0.1.081	2010.11.10	-
TrendMicro	9.120.0.1004	2010.11.09	-
TrendMicro-HouseCall	9.120.0.1004	2010.11.10	-
ViRobot	2010.10.30.4121	2010.11.10	-
VirusBuster	12.72.5.0	2010.11.09	-

Fuente: INTECO