

Ciencia forense informática

Algoritmos de "hashing"

Por: Román Ramírez <rramirez@chasesun.es>
Jesús Arnáiz <jarnaiz@chasesun.es>
Madrid, 20 de Abril de 2004

Introducción

Desde que las comunicaciones digitales aparecieron, los métodos de comprobación de la transmisión de la información han ido evolucionando para garantizar la integridad (recordemos las tres claves de la seguridad informática: disponibilidad, confidencialidad e integridad).

Partiendo del código Morse, hasta las transmisiones más potentes a través de enlaces LMDS o satélite, los algoritmos de verificación de la integridad son de uso común.

Debido a la alta fiabilidad de estos, así como la facilidad y velocidad de uso y proceso (prácticamente todos los sistemas modernos cuentan con herramientas de verificación de integridad), la evolución lógica de este tipo de algoritmos ha sido paralela a la evolución de ciencias como la criptografía o la forense.

La aparición de sistemas avanzados de cifrado como PGP o el protocolo X.500 en su variante X.509, o la mismísima "firma digital", han requerido métodos de verificación rápida de la integridad (autenticidad) que ayuden a sistematizar mecanismos de no repudio.

Algoritmos como MD2, MD4, MD5, SHA-1, SHA-2 (256, 384, 512), RIPEMD-160, PANAMA, TIGER, ADLER32, CRC32, y algunos otros, son ejemplos de sistemas de "hashing" empleados en la comprobación de la integridad.

Más adelante hablaremos sobre algoritmos como CRC, MD5 y SHA1 (los más usados), y sus actuales y potenciales aplicaciones en el ámbito de la tecnología, y en la ciencia forense en particular.

Seguridad informática en general

Dentro del mundo de la seguridad, podemos encontrar miles de aplicaciones de los algoritmos de "hashing", desde en la ciencia forense (como veremos más adelante), hasta en los sistemas de verificación de integridad, pasando por las ya comentadas herramientas de firma electrónica y sistemas de cifrado.

Herramientas como Tripwire, AIDE, md5sum y md5deep, y otras, nos ayudan a obtener los valores de MD5/SHA1 de todos y cada uno de los ficheros dentro de un sistema operativo, de manera que siempre a partir de una gran base de datos que almacene de forma segura estos valores, podemos realizar comprobaciones de las modificaciones realizadas.

De esta manera, podemos localizar ficheros manipulados, substituidos por troyanos o rootkits, o corrompidos por virusos o herramientas de DoS (negación de servicio) *ipso facto*.

Y páginas Web como Known Goods, o Nist, gestionan bases de datos de firmas MD5/SHA1 de prácticamente todos los ficheros de todos los sistemas operativo conocidos.

En Known Goods, podemos conectarnos mediante un navegador a una página Web, <http://www.knowngoods.org/>, donde podemos verificar de forma manual el valor "hash" de ficheros de nuestro sistema (podemos ver más abajo una captura de una petición de prueba con la orden "ls" de un sistema Solaris8). Desde esta página podemos descargar ficheros que compilan todas esas firmas, para verificación local.

En NIST, podemos encontrar a la venta un CD/DVD con todas esas firmas, que además es integrable dentro herramientas como SleuthKit/Autopsy, Encase, FTK, iLook,...), lo que es una gran ventaja desde el punto de vista del investigador forense, ya que se pueden hacer comprobaciones directas de la integridad de los archivos originales dentro de un sistema comprometido.

Seremos capaces de localizar "Caballos de Troya", "rootkits", ficheros manipulados, y todo tipo de cambios que se hayan realizado sobre ficheros conocidos, que se encuentren dentro de estas bases de datos.

known goods

Executables

[Source Releases](#)

[ISO images](#)

Solaris 8.0 sparc

/usr/bin/l^s

- [Advanced Search](#)
- [Mac OS X Search](#)
- [Download](#)
- [Tools](#)

Search

Results 1 - 1 of 1

[previous](#) | [next](#)

Platform: Solaris 8.0 sparc

File:	/usr/bin/l^s
MD5:	351f5eab0baa6eddae391f84d0a6c192
SHA-1:	37c4c703d48470ca5f2e87c45be4303df9535486
Size:	18844 (bytes)

[home](#) | [news](#) | [information](#) | [contact](#)

©2004 The Shmoo Group - Serving 891014 files.

Imagen 1. Buscamos la orden "ls" dentro de Known Goods.

Ciencia forense informática

Esta disciplina moderna, dentro del ámbito de la seguridad informática, se está convirtiendo en una de las más importantes en proyectos de todo tipo; desde proyectos de auditoría y consultorías, hasta proyectos de refutación laboralista (en casos de despidos procedentes o improcedentes) y en casos de investigaciones criminales.

El hecho es que se ha hecho necesario contar con mecanismos y herramientas que garanticen la autenticidad de los datos y evidencias recopilados durante una investigación.

Si sumamos este tipo de sistemas de no repudio, a una investigación sistematizada, y al control estricto de la cadena de custodia y sus responsables (los traspasos de custodia siempre con la aceptación expresa y su recibo correspondiente), podemos llevar nuestras evidencias al entorno judicial con las máximas garantía jurídicas y periciales.

Podemos encontrar el uso de herramientas de MD5 y SHA1 en prácticamente todas las aplicaciones de la ciencia forense informática, desde en la validación de la integridad de evidencias obtenidas de una escena del crimen determinada (por ejemplo, imágenes físicas de discos duros analizadas por el experto forense), hasta en la verificación de la identidad de nuestros interlocutores, mediante firma digital y verificación de la "firma" (en este caso el "hashing") de los mensajes.

En un caso de investigación forense, ya sea de un incidente de seguridad o de un equipo de un empleado, nuestro procedimiento de acceso a las evidencias debe ser siempre el mismo:

1. Acotación de la escena del crimen
2. Localización de la evidencia
3. Cálculo del HASH de la evidencia
4. Obtención de la evidencia
5. Verificación del HASH original y de nuestra copia
6. Almacenamiento seguro de la evidencia

En cualquier momento, y siempre a partir de ese valor de "Hashing", podemos verificar la integridad de las evidencias de forma inequívoca, aparte de que mediante sistemas como los enunciados antes, Tripwire, KnownGoods, Nist, podemos descubrir rápidamente modificaciones y manipulaciones a ficheros, en caso de que nuestra evidencia sea un archivo o sistema de archivos.

CRC

Este algoritmo es ubicuo y estándar a todos los sistemas de comunicaciones modernas. Podemos encontrarlo dentro de las comunicaciones vía TCP/IP, podemos encontrarlo en sistemas de transmisión basados en X.25, en "Frame Relay", y en prácticamente cualquier método empleado por el ser humano para comunicarse mediante ondas.

Tradicionalmente llamado CRC-32, debido a que la longitud del "hash" resultante es de **32bits**, la base de este algoritmo es obtener un bloque de datos de un tamaño determinado (generalmente 32 o 64 kilobytes) y calcular la "firma" resultante, que generalmente se añade al final o al principio del bloque y se transmite con este. El extremo remoto obtiene tanto el bloque de datos como la firma, y puede verificar la integridad de los datos de forma prácticamente instantánea.

Teniendo en cuenta que hablamos de **32bits** de longitud de la cadena resultante, tenemos un conjunto de valores de 2^{32} , es decir, 4.294.967.296 valores posibles en el resultado.

La repetición de los valores del CRC-32, en una misma transmisión y para dos bloques de datos diferentes, es tan improbable que se considera imposible, lo que permite utilizarlo en todos los sistemas de transmisión modernos.

Pero, a la hora de obtener firmas únicas de otro tipo de datos se muestra absolutamente insuficiente, y no es la herramienta adecuada ni para la verificación de la integridad, ni para sistemas de cifrado fiables (ya que sencillos ataques estadísticos contra el cifrado, funcionarán de manera inmediata).

Podemos encontrar información ampliada sobre el algoritmo de CRC-32 en el capítulo 6.4.1 del RFC1510, <http://www.faqs.org/rfcs/rfc1510.html>.

MD5

Este algoritmo nació de la mente del profesor Ron Rivest (la "R" del algoritmo RSA), y se publicó por primera vez en el RFC1321, <http://www.faqs.org/rfcs/rfc1321.html>.

Se desarrolló como una versión ampliada del algoritmo MD4 (RFC1320) que estaba pensando para la velocidad, y no con el objetivo de la seguridad más alta.

El resultante del cálculo de MD5 es un valor de **128bits**, o una serie de **16** caracteres (32 dígitos hexadecimales), lo que da un conjunto de valores de 2^{128} , que en notación científica sería aproximadamente el valor **3,4028236692093846346337460743177e+38**.

Con ese conjunto de valores no es de extrañar que se considere imposible encontrar dos valores de MD5 iguales, y que empresas como Microsoft, Sun Microsystems, IBM y otras, lo empleen para controlar las versiones de sus aplicaciones, objetos ActiveX (COM y .NET), aplicaciones o/y objetos en Java, en protocolos como RMI, Corba...

Esto ha hecho que el número de operaciones realizadas sobre datos que tengan como resultado valores de MD5 haya crecido de forma exponencial en los últimos años.

De hecho, hace unos años, se dio a conocer un famoso caso de dos empresas diferentes que intentaron registrar dos objetos ActiveX, que por casualidades del destino, tenían el mismo valor de MD5 (siendo dos ficheros diferentes).

Si es leyenda urbana o una realidad no es objeto de este artículo, pero el hecho es, que múltiples voces de la comunidad de seguridad abogan por implantar el uso de algoritmos con un conjunto más amplio de elementos, como SHA1.

SHA1

Este algoritmo nace dentro del proyecto Capstone, iniciado por el gobierno de los Estados Unidos como un proyecto de criptografía accesible para todo el mundo.

Este algoritmo es muy potente desde el punto de vista del amplio conjunto de elementos con el que cuenta, ya que se basa en resultados de 160bits de longitud.

En este caso, la longitud del conjunto de elementos resultante es de 2^{160} , es decir un total de **1,4615016373309029182036848327163e+48** posibles en notación científica.

Observamos de forma evidente la práctica imposibilidad de encontrar dos valores iguales de este "hash" para dos entradas diferentes.

Son cada vez más los fabricantes y desarrolladores que optan por la base en SHA1, aunque hay divergencias en el propósito de uso, ya que algunos grupos de especialistas comentan que todos estos algoritmos deberían mantenerse al margen de intereses gubernamentales (apuntando claramente al proyecto Capstone, en este caso).

Anexo I: ejemplos de cálculos con diferentes algoritmos

A partir de la cadena de texto "La seguridad es un tema muy importante", hemos obtenido los diferentes valores de varios algoritmos que ilustren los tipos de resultados:

CADENA:		<i>La seguridad es un tema muy importante</i>
No	Algoritmo	Valor
1	CRC-32	e44c1be6
3	MD4	8be578f00908e505f2497cfdad9a0f69
4	MD5	ca6cf096386381bd968554fab181abf7
5	SHA1	fc232dc1645592d6958cf64623a5dda9189a1402
6	SHA2-256	5e9205980bcb01815442beebc3d7c290008685a9ccb6e9844c27522669545856
7	SHA2-384	3477cc8bbb9fea67905fa827a73d08807eaedad624f49a06 991cbf0d0d8f4190cd393799721f9596aec206505fe71234
8	SHA2-512	f6561739ccbe5b3c6fc889cc09ed9e748dcc41bfea2a687175a93e1381e6a1bac 5b5123aab727591230131ff49c4316df855ed2435b924679aa268e920489d15

Anexo II: glosario

Hash: algoritmos de “hashing”. Es una transformación matemática que toma como entrada un grupo de datos de tamaño variable, y devuelve un resultado de tamaño fijo (ya sea una cadena de texto o un número). Normalmente se representa como $h = H(m)$, donde h es el valor “hash” resultante, H la operación o algoritmo, y m los datos de entrada.

MD5: Message Digest Algorithm. Este algoritmo es una evolución del algoritmo MD4 (ambos desarrollados por Ron Rivest), y que se ha convertido en un estándar mundial para la generación de firmas y verificación de integridad.

PGP: Pretty Good Privacy. Originalmente fue una herramienta desarrollada por Philip Zimmerman con el objetivo de dar acceso a sistemas de cifrado de alta potencia a todo el mundo. Actualmente, es una herramienta comercial (con versión gratuita) que puede integrarse directamente en aplicaciones de correo electrónico, e incluso dentro del propio sistema de ficheros.

RFC: Request For Comments. Documentos que se pueden encontrar en Internet, y que al margen de documentos oficiales, suelen ser el estándar *de facto*, donde podemos encontrar las recomendaciones sobre como implementar una aplicación o sobre el funcionamiento de un determinado protocolo.

SHA1: Secure Hash Algorithm 1. Algoritmo desarrollado por el gobierno de los Estados Unidos para el proyecto Capstone, que tiene como propósito el desarrollo de sistemas criptográficos accesibles al público.

Anexo III: referencias y enlaces

- [1] HashCalc, SlavaSoft Software, <http://www.slavasoft.com/?source=HashCalc.exe>
- [2] RFC.1321, sobre MD5, <http://www.faqs.org/rfcs/rfc1321.html>
- [3] RFC.1510, sobre CRC-32, <http://www.faqs.org/rfcs/rfc1510.html>
- [4] RFC.3174, sobre SHA1, <http://www.faqs.org/rfcs/rfc3174.html>
- [5] SHA1, http://www.w3.org/PICS/DSig/SHA1_1_0.html
- [6] Proyecto Capstone, <http://www.x5.net/faqs/crypto/q150.html>
- [7] Página Web de Ron Rivest, <http://theory.lcs.mit.edu/~rivest/homepage.html>
- [8] La página no oficial de MD5, <http://userpages.umbc.edu/~mabzug1/cs/md5/md5.html>
- [9] The SleuthKit, <http://www.sleuthkit.org>
- [10] Tripwire, <http://www.tripwire.com>
- [11] Known Goods, <http://www.knowngoods.org>
- [12] NIST, <http://www.nist.gov>
- [13] AIDE, <http://aide.sourceforge.net>
- [14] PGP, <http://www.pgp.com>