Universidad Nacional Autónoma de México

Facultad de ingeniería

Sistemas operativos

Hashes, colisiones de hashes y actualidad

Profesor: Wolf Iszaevich Gunnar Eyal

Alumno: Velasco Pachuca Bryan

Grupo: 06

Contenido

¡¿Hash?!

¿Qué es una función hash?

Una función hash o función digestora es una función que te permite transformar cualquier cadena arbitraria de información en una cadena de bits con longitud fija, sin importar el tamaño de la cadena de entrada. [1]

Dicho en otras palabras, es una caja negra cuya entrada puede ser cualquier cadena de bits y la salida es otra cadena de bits de tamaño fijo.

Ejemplo:

Convertimos las cadenas ‘Sistemas operativos’ y ‘Sistemas operativ0s’ usando el hash SHA-1 cuya salida es una cadena de 160 bits de longitud:

‘Sistemas operativos’ ⇒ (SHA-1) $24651F4BC0FBC5F705F2429B7521BCD0673CAE9D_{hex}$

‘Sistemas operativ0s’ ⇒ (SHA-1) $7684C1749D5E15022C58B95AE47DB8B60D555CBE_{hex}$

La longitud de salida de la cadena de bits del hash dependerá de la complejidad técnica y la función de compresión utilizada, por ello, las longitudes de salida más comunes van de los 32 bits a los 512 bits.

A continuación, se muestra la función hash conocida como ELF [2], la cual es una función hash no criptográfica utilizada para almacenar valores en una tabla de hashes:

Código de operación de función Hash ELF en C#, se puede observar la utilización de corrimiento de bit (<< o >>), los operadores AND (&), XOR (^=) y NOT (~).

```
public static uint ELFHash(string str)
{
    uint hash = 0;
    uint x;
    uint i;

    for (i = 0; i < str.Length; i++)
    {
        hash = (hash << 4) + ((byte)str[(int)i]);

        if ((x = hash & 0xF0000000) != 0)
        {
            hash ^= (x >> 24);
        }

        hash &= ~x;
    }

    return hash;
}
```

La función hash toma como entrada un arreglo de bits (*string*) y da como resultado un arreglo de 32 bits (*Unsigned 32-bit integer [uint]*), por lo tanto, el número posible de combinaciones que se pueden obtener es $2^{32} = 4{,}294{,}967{,}296$.

Ejemplo:

Convertimos las cadenas ‘Espero aprobar SO y ‘Espero @probar SO’ en hash ELF, la salida es un arreglo de 32 bits :

‘Espero aprobar SO’ ⇒ (ELF) $CD2642F_{hex}$

‘Espero @probar SO’ ⇒ (ELF) $CD0942F_{\text{hex}}$

Para fines didácticos nótese: En cada iteración se lee un solo carácter del *string* de entrada, por lo que el tamaño del bloque de entrada del hash es de 8 bits y la sección que realiza la retroalimentación:

```
hash = (hash << 4) + ((byte)str[(int)i]);

if ((x = hash & 0xF0000000) != 0)
{
    hash ^= (x >> 24);
}

hash &= ~x;
```

Solo se ejecuta una sola vez, por lo tanto, es de una sola ronda; cuando en una función hash la retroalimentación sucede n-veces por bloque, se dice que tiene n-rondas.

Si se realizase una segunda ronda en donde se retroalimente con el resultado de la primera:

```
hash = (hash << 4) + hash/2;

if ((x = hash & 0xF0000000) != 0)
{
    hash ^= (x >> 24);
}

hash &= ~x;
```

Se obtienen las siguientes salidas:

‘Espero aprobar SO’ ⇒ (ELF) $\mathrm{F99084F}_{hex}$

‘Espero @probar SO’ ⇒ (ELF) $\mathrm{B447827}_{\mathrm{hex}}$

Características de un buen hash a nivel teórico

Una función hash deberá cumplir con las siguientes características teóricas [3]:

1. No reversible (one-way function).
2. Con efecto avalancha o difuso.
3. Determinista.
4. Resistente a colisiones.
5. No predecible.

Características de un buen hash a nivel técnico

Una función hash también deberá cumplir con las siguientes características técnicas [4]:

1. First preimage resistance: Dado un hash resultado de una función, deberá ser difícil encontrar la entrada que lo produjo.
2. Collision resistance: Deberá ser “difícil” obtener/encontrar dos entradas diferentes con el mismo hash.
3. Second preimage resistance: Dada una entrada con su respectivo hash, deberá resultar “imposible” encontrar otra entrada que comparta el mismo hash.

Tipos de hashes

Existen múltiples tipos de hashes, con tamaños de salida variados y estados internos completamente diferentes, no obstante, nos enfocaremos en el conjunto de hashes relevante para este trabajo, el cual es la familia SHA (*Secure Hash Algorithm*).

Familia SHA

La familia SHA (*Secure Hash Algorithm*) es una familia de funciones hashes criptográficas que inicialmente fue creada por el NIST (*National Institute of Standards and Technology*) en conjunto con la NSA (*National Security Agency*) cuando se creó el primer hash, el SHA-0.

Posteriormente, se introdujo el SHA-1 como sucesor del SHA-0, no obstante, debido a la poca complejidad algorítmica presentada, el NIST publicó un conjunto nuevo de funciones hash conocidas como SHA-2.

Finalmente, se adoptó el hash SHA-3 (también conocido como Keccak [‘ketʃaʔ’]) tras una competición organizada por el NIST para encontrar un nuevo estándar de hash.

SHA-0

Publicado en 1993 como el primer SHA. En 1998 fueron encontradas las primeras colisiones y desde el 2004 se confirmaron con certeza varias colisiones en la salida de un hash SHA-0 reducido de 65 rondas y colisiones en 142 de los 160 bits de salida para el SHA-0 completo de 80 rondas [5].

SHA-1

Publicado en 1995 y basado en SHA-0, también emplea 80 rondas y devuelve una cadena de 160 bits.

Desde el 2005 se encontraron colisiones en una versión reducida de SHA-1 con 53 rondas y más tarde en 2010 en una versión de 73 rondas, por ello en 2011 el NIST decidió marcar al SHA-1 como obsoleto [6].

Más adelante, en 2017 se creó el ataque SHAttered en el que se demostró que era posible generar colisiones en SHA-1, especialmente en documentos PDF, obteniendo así la primera colisión registrada para el SHA-1 completo de 80 rondas [7].

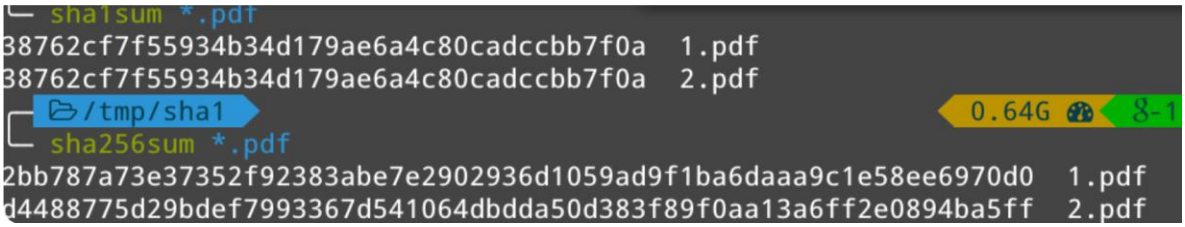

Imagen proporcionada por shattered.io, en donde se pueden observar primero 2 PDF's distintos con el mismo hash SHA-1 pero con diferentes hashes SHA-256.

En 2020 se realizó una implementación práctica en la que era redituable producir colisiones en la herramienta de firmas digitales GnuPG a partir de añadir al inicio de dos firmas diferentes, dos valores calculados que lograsen producir el mismo hash (*Chosen-prefix collision attack*) con un costo de $45,000 dólares por colisión (aprox. $890,000 pesos mexicanos) [8].

SHA-2 (SHA2-224, SHA2-256, SHA2-384 y SHA2-512)

Es un conjunto de funciones hash presentadas por primera vez en 2001 como los verdaderos sucesores de SHA-0 y SHA-1 [9], mostrando cambios más significativos en su estructura, como un tamaño de bloque de 512 a 1024 bits, operaciones de desplazamiento de bit y de 64 a 80 rondas según el tipo de SHA-2.

En 2008 se encontraron colisiones para un SHA2-256 de 24 rondas y en 2016 para un SHA2-512 reducido de 27 rondas [10].

A pesar de no estar registrada aún alguna una colisión en un SHA-2 completo de 80 rondas, se continua investigando su seguridad en búsqueda de alguna vulnerabilidad.

SHA-3/Keccak (SHA3-224, SHA3-256, SHA3-384 y SHA3-512).

Es el último miembro de la familia SHA adoptado oficialmente el 5 de agosto de 2015 por ser el ganador de la *NIST hash function competition* con notables diferencias respecto al SHA-2.

Al igual que SHA-2, cuenta con diferentes formatos donde el tamaño de su bloque de entrada varía de 576 a 1152 bits empleando 24 rondas por bloque para todos los formatos [11].

En 2012 se pudieron encontrar colisiones en un SHA3-256 de 4 rondas [12] y en 2019 para uno de 5 rondas [13], lo cual está bastante lejos de comprometer la seguridad del SHA-3.

La actualidad del SHA

Cuando que el NIST declaró al SHA-1 como obsoleto e inseguro en 2011, múltiples empresas, organizaciones e instituciones que lo utilizaban migraron a SHA-2, dejando interesantes secuelas en múltiples ámbitos.

Protocolo de encriptación IPsec

En 2007 el protocolo de encriptación IPsec (*Internet Protocol Security*) comenzó a utilizar SHA-2 en lugar de SHA-1 como base para la autenticación del origen de datos y la verificación de su integridad, lo cual se sigue utilizando hasta el día de hoy [14] [15].

Algoritmo de encriptación HMAC

El algoritmo de encriptación HMAC (*Keyed-Hashing for Message Authentication Code*) no se vio comprometido con las colisiones demostradas en SHA-1, pero se recomienda usar su versión basada en SHA-2 (HMAC-SHA2) o SHA-3 (HMAC-SHA3) [16].

Certificados SSL

Desde 2017 Google anunció que dejaría de dar soporte a certificados SSL (*Secure Sockets Layer*) basados en SHA-1, mostrando advertencias sobre las seguridad del sitio cuando se desease ingresar, esto alentó a todos los operadores de servidores a migrar al SSL basado en SHA-2 [17].

Procesadores Intel Xeon

Desde septiembre de 2019 la gama de procesadores Intel Xeon [Silver, gold, Platinum y W] basada en la microarquitectura Ice Lake, cuenta con instrucciones de aceleración de hardware para operaciones de hashing con SHA [18].

Microsoft Download Center

El 3 de agosto de 2020, todo software descargable de Microsoft firmado por SHA-1 fue removido del *Microsoft Download Center* [19].

Git

Git ampliamente ha utilizado al SHA-1 como forma de identificación y aseguramiento de la integridad de cualquier objeto o *commit* y tras el ataque SHAttered de 2017 para SHA-1, toda la infraestructura de Git se vio comprometida.

En 2018 se hizo un análisis exhaustivo de diferentes funciones hash que podrían reemplazar al SHA-1, quedando el SHA2-256 como sucesor (se ignoró fuertemente al SHA-3 debido a su falta de compatibilidad) [20].

Desde la versión 2.29 en 2020 se comenzó a dar soporte experimental para SHA-2, configurando el archivo *.repo* del repositorio de esta manera [21]:

```
[core]
        repositoryFormatVersion = 1
[extensions]
        objectFormat = sha256
        compatObjectFormat = sha1
```

Migrar a SHA-2 requerirá clonar los repositorios deseado y reescribirlos para que fuesen compatible con SHA-2 (sin excepción).

Viendo más a futuro, se espera crear la infraestructura adecuada para llegar a implementar hashes más largos (>256 bits) que estén preparados para la seguridad cuántica.

Hasta la última versión (2.35.1) lanzada este año no se cuenta aún con una implementación por default del SHA-256 en Git [22].

Conclusiones

###

Referencias

1. HTTPS://EDPS.EUROPA.EU/SITES/EDP/FILES/PUBLICATION/19-10-30_AEPD-EDPS_PAPER_HASH_FINAL_EN.PDF, OVERVIEW DE FUNCIONES HASH.
2. https://www.programmingalgorithms.com/algorithm/elf-hash/, CÓDIGO DE ELF HASH.
3. HTTPS://WWW.SYNOPSYS.COM/BLOGS/SOFTWARE-SECURITY/CRYPTOGRAPHIC-HASH-FUNCTIONS/, CARACTERÍSTICAS TEÓRICAS DE UNA FUNCIÓN HASH.
4. HTTPS://WWW.COALFIRE.COM/THE-COALFIRE-BLOG/PROPERTIES-OF-SECURE-HASH-FUNCTIONS, CARACTERÍSTICAS TÉCNICAS DE UNA FUNCIÓN HASH.
5. HTTPS://LINK.SPRINGER.COM/CONTENT/PDF/10.1007/978-3-540-28628-8_18.PDF, COLISIONES EN HASH SHA-0 EN 2004.
6. HTTPS://EPRINT.IACR.ORG/2010/413.PDF, COLISIONES EN SHA-1 DESDE 2010
7. HTTPS://MARC-STEVENS.NL/RESEARCH/PAPERS/SBKAM17-SHATTERED.PDF, THE FIRST COLLISION FOR FULL SHA-1.
8. HTTPS://EPRINT.IACR.ORG/2020/014.PDF , COLISIONES EN SHA-1 EN 2020.
9. HTTPS://WWW.FEDERALREGISTER.GOV/DOCUMENTS/2002/08/26/02-21599/ANNOUNCING-APPROVAL-OF-FEDERAL-INFORMATION-PROCESSING-STANDARD-FIPS-180-2-SECURE-HASH-STANDARD-A, APROBACIÓN DE SHA-2 COMO FIPS
10. HTTPS://WWW.COMPARITECH.COM/BLOG/INFORMATION-SECURITY/WHAT-IS-A-COLLISION-ATTACK/, COLISIONES EN LOS DIFERENTES HASHES.
11. HTTP://WWW.CRIPTORED.UPM.ES/DESCARGA/CLASS4CRYPTC4C7.5_SHA-2_SHA-3_RESUMEN_HASH.PDF, SHA-3
12. HTTPS://EPRINT.IACR.ORG/2012/672.PDF, COLISIONES EN SHA-3 CON ESTADOS INTERNOS ALTERADOS.
13. HTTPS://EPRINT.IACR.ORG/2019/147.PDF, COLISIONES EN SHA-3
14. HTTPS://DATATRACKER.IETF.ORG/DOC/HTML/RFC4868, MEMO DE IPSEC A CERCA DE LA UTILIZACIÓN DE SHA-2.
15. HTTPS://WWW.WATCHGUARD.COM/HELP/DOCS/FIREWARE/12/ES-419/CONTENT/ES-419/MVPN/GENERAL/IPSEC_ALGORITHMS_PROTOCOLS_C.HTML, PROTOCOLO IPSEC.
16. HTTPS://EPRINT.IACR.ORG/2006/187.PDF, SEGURIDAD DEL ALGORITMO DE ENCRIPTACIÓN HMAC.
17. HTTPS://WWW.BITDEFENDER.COM/BLOG/HOTFORSECURITY/GOOGLE-FOLLOWS-MICROSOFT-FIREFOX-IN-BLOCKING-SHA-1-CERTIFICATES, GOOGLE DEJA DE DAR SOPORTE A SSL BASADO EN SHA-1.

18. https://en.wikichip.org/wiki/intel/microarchitectures/ice_lake_(client), microarquitectura Ice Lake.
19. https://docs.microsoft.com/en-us/lifecycle/announcements/sha-1-signed-content-retired#:~:text=On%20August%203%2C%202020%2C%20SHA,from%20the%20Microsoft%20Download%20Center, UPDATE: SHA-1 signed content to be retired.
20. https://lore.kernel.org/git/20180609224913.GC38834@genre.crustytoothpaste.net/, discusión de los candidatos para reemplazar al SHA-1
21. https://www.infoq.com/news/2020/10/git-2-29-sha-256/, soporte experimental de git para SHA-2 en la versión 2.29
22. https://git-scm.com/docs/hash-function-transition/, documentación de la transición a SHA-2 en la versión 2.35.1

Fuentes:

1. https://perso.uclouvain.be/fstandae/source_codes/hash_atmel/specs/jh.pdf
2. https://www.researchgate.net/profile/Geetha-Ganesan/publication/267422045_Cryptographic_Hash_Functions_A_Review/links/549cf6d10cf2b8037138c35c/Cryptographic-Hash-Functions-A-Review.pdf
3. https://hash.online-convert.com/
4. https://en.wikipedia.org/wiki/Comparison_of_cryptographic_hash_functions
5. https://crypto.stackexchange.com/questions/16219/cryptographic-hash-function-for-32-bit-length-input-keys
6. https://en.wikipedia.org/wiki/Cryptographic_hash_function.
7. https://www.drdobbs.com/database/hashing-rehashed/184409859
8. https://crypto.stackexchange.com/questions/95592/264-versions-of-the-same-message
9. https://blog.infocruncher.com/resources/ethereum-whitepaper-annotated/On%20the%20Secure%20Hash%20Algorithm%20family%20(2008).pdf
10. https://en.wikipedia.org/wiki/Collision_attack
11. https://en.wikipedia.org/wiki/Secure_Hash_Algorithms
12. https://lwn.net/Articles/823352/