Vulnerabilidad Shattered en cifrado SHA-1.

Rafael Leyva, Adrian Orduña.

June 18, 2017

Contents

1 SHA-1.

El SHA, siglas de Secure Hash Algorithm, es un algoritmo criptográfico de cifrado con funciones hash, desarrollado por el Instituto Nacional de Estándares y Tecnología (NIST), ampliamente usado para el cálculo de hash para comprobar la consistencia y cambios en archivos.

2 ¿En qué consiste SHA-1?

SHA-1 se basa en principios similiares a los usados por el profesor Ronald L. Rivest del MIT en el diseño de los algoritmos MD4 y MD5.

Para ello aplica al mensaje de entrada una serie de funciones algebráicas de modo que cada carácter del mensaje original es pareado con uno o más bits del hash final. Estos bits son ajustados para siempre producir una salida de 160 bits.

SHA-1 produce una salida de 160 bits (20 bytes) de un mensaje inicial que puede tener un tamaño máximo de 2 bits.

Aqui podemos ver el pseudocódigo del algoritmo en Wikipedia:

```
// Note 1: All variables are unsigned 32-bit quantities and wrap modulo 232 when calcu
//         ml, the message length, which is a 64-bit quantity, and
//         hh, the message digest, which is a 160-bit quantity.
// Note 2: All constants in this pseudo code are in big endian.
//         Within each word, the most significant byte is stored in the leftmost byte p

Initialize variables:

h0 = 0x67452301
h1 = 0xEFCDAB89
h2 = 0x98BADCFE
h3 = 0x10325476
h4 = 0xC3D2E1F0

ml = message length in bits (always a multiple of the number of bits in a character).

//Pre-processing:
 append the bit '1' to the message e.g. by adding 0x80 if message length is a multiple
 append 0 | k < 512 bits '0', such that the resulting message length in bits
   is congruent to |64 | 448 (mod 512)
append ml, the original message length, as a 64-bit big-endian integer. Thus, the total

//Process the message in successive 512-bit chunks:
break message into 512-bit chunks
for each chunk
    break chunk into sixteen 32-bit big-endian words w[i], 0 | i | 15

    Extend the sixteen 32-bit words into eighty 32-bit words:
```

```
    for i from 16 to 79
        w[i] = (w[i-3] xor w[i-8] xor w[i-14] xor w[i-16]) leftrotate 1

    Initialize hash value for this chunk:
    a = h0
    b = h1
    c = h2
    d = h3
    e = h4

    Main loop:[3][53]
    for i from 0 to 79
        if 0 | i | 19 then
            f = (b and c) or ((not b) and d)
            k = 0x5A827999
        else if 20 | i | 39
            f = b xor c xor d
            k = 0x6ED9EBA1
        else if 40 | i | 59
            f = (b and c) or (b and d) or (c and d)
            k = 0x8F1BBCDC
        else if 60 | i | 79
            f = b xor c xor d
            k = 0xCA62C1D6

        temp = (a leftrotate 5) + f + e + k + w[i]
        e = d
        d = c
        c = b leftrotate 30
        b = a
        a = temp

    Add this chunk's hash to result so far:
    h0 = h0 + a
    h1 = h1 + b
    h2 = h2 + c
    h3 = h3 + d
    h4 = h4 + e

Produce the final hash value (big-endian) as a 160-bit number:
```

```
hh = (h0 leftshift 128) or (h1 leftshift 96) or (h2 leftshift 64) or (h3 leftshift 32)
```

3 Versiones de SHA.

La primera versión del algoritmo fue creada en 1993 con el nombre de SHA, aunque en la actualidad se la conoce como SHA-0 para evitar confusiones con las versiones posteriores.

- La segunda versión del algoritmo, publicada con el nombre de SHA-1, fue publicada dos años más tarde.
- SHA-2 fue anunciado en 2001 y estaba formado por diversas funciones: SHA-224, SHA-256, SHA-384 y SHA-512).
- SHA-3 fue seleccionado en una competición de funciones hash celebrada por el NIST en 2012, esta última versión se caracteriza por ser la más diferente de sus predecesoras.

4 Vulnerabilidades descubiertas.

A lo largo de su vida, a esta familia de algoritmos se le conocen varias vulnerabilidades. En 1998, se encontró una vulnerabilidad para SHA-0, aunque esta no se podía extender a SHA-1. En cualquier caso, la NSA aumentó en ese momento la seguridad del SHA-1. En 2004, se encontró una debilidad matemática den SHA-1, que permitía encontrar colisiones de hash más rápido. Sin embargo, este hallazgo resulta poco relevante, pues la complejidad de búsqueda de colisiones pasaría de 2^{80} a 2^{69}, algo que aún es computacionalmente inviable, requiriendo incluso más trabajo que MD5 (2^{64}).

4.1 Shattered.

En 2017, un equipo formado por Google y CWI Amsterdam, anunciaron en Febrero, la primera colisión de SHA-1, la cual ha sido nombrada como SHAttered, poniendo de manifiesto que las vulnerabilidades matématicas descubiertas para dicho algoritmo son explotables.

El experimento de este equipo consistió en generar un hash idéntico para dos PDF distintos que pueden ser descargados en la web que han dedicado a dicho experimento `https://shattered.it/` en la cual además describen más detalladamente el proceso para simular este experimento.

A grandes rasgos y sin entrar en detalles técnicos las colisiones han de ser diseñadas, es decir el segundo archivo debe ser modificado para generar el hash del primero, en el caso de los PDF se hace añadiendo ciertos parámetros a los metadatos del archivo.

5 Usos de SHA-1.

Los usos más comunes que de SHA-1 que se han visto afectados por las colisiones descubiertas son:

- Certificados de las firmas digitales.
- Firmas PGP/GPG de los emails.
- Actualizaciones software, validando las actualizaciones del siguiente modo:
  - El fabricante calcula el hash del archivo de actualización.
  - El usuario recibe la actualización y calcula su hash.
  - Si el hash coincide la actualización es correcta.
  - En otro caso la actualización se ha visto comprometida y no es recomendable aplicarla.
- Firma de autenticidad de las imagenes de los sistemas operativos,

procediendo de un modo similar al de las actualizaciones.

- Sistemas de backup, permitiendo el uso de funciones hash comprobar

los archivos que han sido modificados (aquellos que han cambiado su hash) para ser modificados en la copia de seguridad. Es el sistema que usa la orden rsync vista en las prácticas por ejemplo.

- GIT

6 ¿A qué afecta esta vulnerabilidad?

Como ya se ha comentado con anteriorida, la posibilidad de ocasionar colisiones en distintos archivos mediante el protocolo SHA-1 tiene varias implicaciones en software y sistemas que son utilizados a diario por miles de usuarios. A continuación vamos a detallar unos cuantos:

6.1 Cifrado PGP/GPG

Este tipo de cifrado de clave público/privada es muy utilizado en servicios de correo por ejemplo. Las implicaciones en este protocolo de la vulnerabilidad descubierta por el equipo de Google tiene un gran impacto ya que es posible que el receptor reciba un correo que no es el que en un inicio se le envió, pudiendo haber sido éste modificado por un ataque de man in the middle, y habiendo alterado el contenido del original sin que el receptor de dicho correo se de cuenta de dicho ataque.

6.2 Validación de archivos .iso con checksum.

En la actualidad cuando nos disponemos a instalar un nuevo servicio operativo en cualquier equipo y descargamos su imagen es ampliamente recomendable comprobar si el hash de dicho archivo con extensión .iso coincide con el que el proveedor de la imagen dice que debe de tener ya que en otro caso el archivo podria estar corrupto o no ser el mismo que deseabamos descargar. En este caso se podría cambiar esa imagen por otra que contenga malware o troyanos y sin darnos cuenta tener dicho software malicioso en el servidor de nuestra empresa por ejemplo.

6.3 Control de versiones con git.

Actualmente, prácticamente la totalidad de proyectos de desarrollo se gestionan usando el protocolo Git, diseñado por Linus Torvalds en la década de 1990.

Dicho protocolo utiliza SHA-1 para tener un identificador único de cada commit, el cual indexa los cambios que han sido realizados desde el ultimo commit de modo que siempre podemos restablecer el proyecto a un estado anterior.

En este caso la vulnerabilidad tiene una implicación clara y es que todo el sistema se podria ver comprometido, ya que en el caso de existir dos commits con el mismo hash, el sistema no sabría como resolver esto.

Cada repositorio tiene un hash asociado con lo cual, ante la existencia de dos repositorios con dos hash iguales, podríamos estar en riesgo de malware ya que sería imposible detectar un repositorio malicioso de uno que no lo es.

Además se usa este tipo de comprobación para la firma de los commits, etc. con lo cual las implicaciones son críticas en este caso.

A pesar de esto, la técnica para crear colisiones es compleja por lo que todavía no es posible generar todo esto con facilidad, aunque es cuestión de tiempo.

6.4 Apache subversion (svn)

Subversion es una alternativa a Git de la fundación Apache, la cual está más comprometida que Git con esta vulnerabilidad, ya que usa SHA-1 para comprobar archivos duplicados, y en caso de tener dos archivos con el mismo hash el sistema se bloquea y cae.

7 ¿Cómo evitar la vulnerabilidad Shattered?

Como ya se ha dicho anteriormente, SHA-1 lleva siendo teóricamente vulnerable muchos años y hay otros protocolos nuevos que cubren dichas vulnerabilidades como SHA-3 y SHA-256 que sí que son inquebrantables, al menos a día de hoy, ya que todas estos algoritmos se basan en la imposibilidad de hacer cálculos por fuerza bruta con la potencia actual.