

Una breve introducción al *cracking* de contraseñas

Una contraseña es un dato secreto que permite acceder a un recurso cuyo acceso está limitado. Este se mantiene en privado de tal manera que cuando alguien desea acceder al recurso, se le somete a una prueba para ver si es poseedor del secreto, garantizando el acceso en caso afirmativo o denegándolo en caso contrario.

Hoy en día las contraseñas son ampliamente utilizadas en el mundo de la informática para restringir el acceso a sistemas operativos, servicios en red, documentos, interfaces de control, etc, pero, por regla general, en la mayoría de los sistemas, la gestión de passwords no se realiza de maneras lo suficientemente seguras.

Veamos los sistemas que implementan los dos sistemas operativos más extendidos, MS Windows y UNIX.

1 UNIX.

1.1 Dónde y cómo se guardan las contraseñas.

La mayor parte de sabores UNIX (freeBSD, AIX, Linux, Solaris, HP-UX, Tru64. . .) guardan toda la información relativa a las contraseñas en dos ficheros: el */etc/passwd* y el */etc/shadow*, aunque este último varía su nombre y localización dependiendo del UNIX particular.

Las contraseñas no se almacenan en texto plano sino que se codifican aplicando una función hash, típicamente MD5 o SHA. Esto genera un “*hash*” de la contraseña, que es como su DNI, una cadena de caracteres que la identifica de forma única. Un hash se puede comparar también con el ADN humano, que en teoría es algo único para cada individuo. Por tanto, una contraseña como puede ser “*qwerty*”, se almacenaría como “d8578edf8458ce06fbc5bb76a58c5ca4”. Esto sucede por dos motivos principalmente:

1. Evitar que el administrador de la máquina pueda ver los passwords de los usuarios. Esto tentaría a más de uno, que podría intentar probar si la

contraseña del usuario para ese sistema coincide con la de su buzón de correo electrónico, su cuenta de Blogger o su Banco On-Line...

2. Evitar el desastre total si el fichero `shadow` es robado. De esta manera si dicho fichero cayese en malas manos, la única información que, en principio, se sacaría en claro sería una lista de logins.

Las entradas del fichero `passwd` tienen esta pinta:

```
root:x:0:0:root:/root:/bin/bash
michael:x:561:561:Michael Knight:/home/michael:/bin/bash
```

De izquierda a derecha, los campos significan:

- Login del usuario
- La X significa que el hash se guarda en el fichero `/etc/shadow`
- UserID, el identificador de usuario.
- GroupID, el identificador del grupo al que pertenece el usuario.
- Nombre real.
- Directorio inicial.
- Interpretador de comandos a utilizar.

Las entradas del fichero `shadow` son así:

```
root:$1$8tldVM3U$R/b8Zuc97O/.oDuJfWKdZ3:13214:0:99999:7:::
michael:$1$6ptdlGeX$IPdux8frIty3yrP3EPs7/:10063:0:99999:7:::
```

De izquierda a derecha, los campos significan:

- Login del usuario.
- Hash MD5¹ del password. Si está en blanco significa que esa cuenta no necesita password. Si contiene un asterisco * significa que la cuenta está deshabilitada.
- El número de días desde el 1 de Enero de 1970² que han pasado desde la última vez que se cambió el password.
- El número de días que ha de esperar el usuario para poder cambiar su password. Un 0 significa que el password puede ser cambiado en cualquier momento.

¹El tipo de función *hash* utilizada varía de unos UNIX a otros.

²Este sistema de medición se conoce como *epoch time*.

- El numero de días restantes para que el usuario sea obligado a cambiar el password. Si ese valor es 99999, el usuario no esta obligado a cambiar su contraseña.
- El numero de días de antelacion con que se avisará al usuario de que su password está a punto de expirar.
- El numero de días que tienen que pasar para que la cuenta sea deshabilitada despues de que el password caduque.
- El numero de días desde el 1 de Enero de 1970 que han pasado desde que se deshabilitó la cuenta.
- Un campo reservado para uso futuro.

Merece la pena destacar, que UNIX por defecto utiliza “*salts*”. A grandes rasgos esto significa que al password original se han añadido algunos bits aleatorios (típicamente 12 bits) de tal manera que para cada posible password, hay 4096 hashes posibles. Esto implica que, por ejemplo, sea imposible ver a simple vista si dos usuarios utilizan el mismo password, o que la creacion de diccionarios *hash* precomputados sea mucho más difícil.

1.2 El proceso de autenticación.

El proceso de autenticación ante una máquina UNIX podría simplificarse como:

- Cliente: Quiero conectarme.
- Servidor: OK, ¿quien eres?
- Cliente: Soy el usuario “Vicentin”.
- Servidor: OK, Vicentín, ¿cual es tu contraseña?
- Cliente: Mi contraseña es “SiempreAtOpE-5.0”

Ahora el servidor lee el hash que corresponde al usuario “Vicentin” y ve que es “3497db22a28ce7cccf392ed4ba1c99c4”. Aplica el algoritmo MD5 al password recibido desde la conexion con Vicentin y sale “3497db22a28ce7cccf392ed4ba1c99c4”. Comprueba si son iguales y como lo son:

- Servidor: Acceso Concedido: Bienvenido Vicentín.

1.3 Crackeando las contraseñas

Bueno, todo esto es muy bonito, pero, ¿Cómo de seguro es este sistema? ¿Puedo estar tranquilo? La respuesta es complicada.

Si tenemos acceso físico a la máquina UNIX, facilmente podremos arrancar desde un LiveCD, montar la particion correspondiente (con el LiveCD tenemos privilegios de root) y robar los ficheros passwd y shadow. O peor aún, podríamos editar dichos ficheros, añadir un par de lineas y tener lista una cuenta de usuario con privilegios para nosotros solos.

Si no tenemos acceso físico la cosa se vuelve muchisimo más complicada. Tendríamos que encontrar alguna vulnerabilidad que se pudiera explotar para hacer una escalada de privilegios y hacernos con ambos ficheros y eso es algo que requiere bastantes más conocimientos.

En cualquier caso, supongamos que ya tenemos la lista de logins con sus correspondientes contraseñas *hasheadas* con MD5. Ahora vamos a utilizar John The Ripper, uno de los crackeadores de contraseñas más populares, que además es de código abierto. Un crackeador de contraseñas es un programa que prueba diferentes passwords, uno tras otro, hasta dar con el correcto.

A la hora de probar combinaciones tenemos dos opciones, probar palabras de diccionario o probar todas las combinaciones posibles.

A lo primero se le llama un ataque de diccionario y suele ser más efectivo en terminos de tiempo, ya que es más probable que un usuario utilize passwords como “guitarra”, antes que “4xs\$dfer”. Un ataque de diccionario suele incorporar también algo de heurística, de tal manera que no solo se prueban las palabras del diccionario, sino que por ejemplo se pueden tratar de combinar palabras, añadirles números, etc. De esta forma se probarían passwords como “Marta1981” o “España82”, que son muy comunes entre el usuario medio.

La segunda opcion es el ataque por fuerza bruta y consiste en probar todas las combinaciones posibles hasta dar con la correcta. A primera vista puede parecer que esta solución es infalible, ya que si probamos todas las combinaciones, al final daremos con la correcta. Esto es totalmente cierto pero tiene un inconveniente muy grave: el tiempo. Veamos un ejemplo:

Tenemos un teclado que nos permite introducir cualquiera de los siguientes caracteres imprimibles:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ  
abcdefghijklmnopqrstuvwxyz  
1234567890  
.-_+`~:;@*^[]\}{!~$%&'()/=?&|'<>
```

Aproximadamente 97 caracteres. Supongamos que nuestra contraseña tiene una longitud de 12 caracteres.

Por tanto. Base 97. Longitud: 12

Combinaciones posibles: $97^{12} = 693.842.360.995.438.000.295.041$ (¡Joder!)

Eso que parece el saldo de la cuenta de Bill Gates, es un numero extremadamente enorme de posibilidades diferentes, algo que hace que computacionalmente sea imposible dar con la clave en un tiempo razonable.

Vamos a ser optimistas y considerar que en nuestra máquina, John the Ripper es capaz de probar medio millón de passwords por segundo. Por tanto $500.000 * 60$ segundos * 60 minutos * 24 horas * 365 dias = 15.768.000.000.000 combinaciones por año. No son pocas, pero ni de lejos las suficientes. Podemos concluir por tanto que con la potencia de calculo actual de un ordenador corriente, para romper por fuerza bruta un password de 12 caracteres se tardaría más tiempo del que podemos vivir, por lo que el problema se vuelve impracticable.

De todas formas, podeis probar. La forma de utilizar John The Ripper es muy sencilla. Descargais los ficheros fuente, lo compilais y dentro del directorio `./run` ejecutad:

```
$ : ./john /ruta/hacia/el/fichero/shadow (fuerza bruta sobre los hashes del fichero)
```

```
$ : ./john -WORDLIST=ficheroeldiccionario /ruta/hacia/el/fichero/shadow (ataque basado en diccionario)
```

Si necesitais interrumpir el proceso, podeis hacerlo pulsando *Control+C*. Para reanudar basta con ejecutar:

```
$ : ./john -restore
```

2 WINDOWS

2.1 Dónde y cómo se guardan las contraseñas.

Windows NT/2000/XP/2003 funciona de manera algo distinta y como es costumbre en Redmond, menos segura. Windows tampoco guarda las contraseñas en texto plano sino que lo hace utilizando dos sistemas diferentes: LAN Manager hash y NTLM Hash. Los hashes se guardan en la base de datos local SAM o en el Active Directory, dependiendo del tipo de Windows instalado. A grandes rasgos, LM Hash es un sistema inseguro y NTLM es un sistema más o menos seguro. Lo que ocurre es que en Windows podemos aplicar el principio de la cadena “la resistencia de una cadena es la resistencia de su eslabón más debil”. Es decir, que mientras esté habilitado el uso de LMHashes (por defecto lo están), no vamos a tener que hacer grandes esfuerzos para romper las contraseñas de un sistema Windows. Para entender todo un poco mejor veamos como funciona el LM Hashing.

2.2 LAN Manager Hash

LM Hash fue desarrollado a principios de los 70 por IBM para usarlo con sus flamantes (por aquel entonces, claro) ordenadores 360/370. En los 80, este algoritmo fue adoptado por el LAN Manager, un experimento conjunto de IBM y Microsoft hasta que este último decidió romper la alianza en vista del gran éxito que había tenido su Windows 3.11. Desde entonces, por razones de compatibilidad hacia atrás, todos los sistemas Windows almacenan sus contraseñas utilizando hashes LM. El inconveniente de esta compatibilidad hacia atrás es que se acaba utilizando un sistema que provee un grado de seguridad extremadamente débil, ya que es posible romper las contraseñas del sistema por fuerza bruta en un tiempo más que aceptable.

El algoritmo funciona más o menos así:

- Todas las minúsculas que contenga la contraseña son convertidas a mayúsculas, lo que reduce en 27 caracteres la base sobre la que trabajamos. A menor base, menores combinaciones posibles. . .
- La contraseña se rellena con NULLs por la derecha hasta que tiene una longitud exacta de 14 caracteres, que es la máxima longitud posible.
- La contraseña se separa en dos bloques de 7 caracteres cada uno.
- Se le aplica el algoritmo DES a cada uno de los bloques.
- Se concatenan los dos hashes obtenidos, consiguiendo una cadena de 128bits.

Como habéis podido deducir, esto es sencillísimo de romper, ya que además de que la base sobre la que trabajamos es menor, el hecho de que la clave se parta en bloques de 7 caracteres hace posible crackear ambos por separado. Por tanto, en el peor de los casos tendríamos:

Base: $97-27=70$; Longitud=7 ; Combinaciones $70^7=8.235.430.000.000$.

Suponiendo que pudiésemos probar medio millón de combinaciones por segundo, tardaríamos, en el peor de los casos, 190 días en dar con la clave. Creedme, esto es un tiempo de risa.

2.3 Crackeando las contraseñas

A diferencia de UNIX, los hashes no se guardan en un simple fichero de texto plano sino que hace falta una herramienta especial para poder obtenerlos. Por contra, mientras que en UNIX solo el root puede acceder al fichero, en Windows, cualquier usuario sin privilegios puede hacerse con los hashes, basta con tener una cuenta en la máquina, que permita ejecutar una aplicación, en concreto `pwdump2`.

Hay muchos crackeadores de contraseñas para Windows. Hay algunos que extraen los hashes de forma automática. Entre los más importantes hay que mencionar L0pthCrack, muy bueno pero comercial y Ophcrack2, que ahora se distribuye en formato LiveCD listo para arrancar-y-crackear.

¿Quiere esto decir que cualquier usuario va a romper el password del administrador? No si deshabilitamos los hashes LM. Esto se puede conseguir de dos formas, siguiendo los pasos que Microsoft detalla en su documento *“How to prevent Windows from storing a LAN manager hash of your password in Active Directory and local SAM databases”* o utilizar passwords de longitud mayor a 15 caracteres. Dada la limitación de longitud del sistema LM, cualquier password que supere los 14 caracteres debería ser truncado, por lo que sabiamente Windows opta por utilizar exclusivamente NTLM.

NOTA: Este documento solo pretende ser una introducción a los conceptos relacionados con la seguridad de las contraseñas. Se han quedado muchas cosas importantes en el tintero, como por ejemplo el uso de las Rainbow Tables o el tema de las colisiones. Además, el texto no ha sido sometido a revisión por experto alguno por lo que no debe tomarse como fuente de información fiable pues ni siquiera el autor garantiza su total veracidad y precisión.

Referencias

- Pwdump2
<http://www.bindview.com/Services/razor/Utilities/Windows/pwdump2_readme.cfm>
- Ophcrack
<<http://ophcrack.sourceforge.net/>>
- How to prevent Windows from storing a LAN manager hash of your password in Active Directory and local SAM databases.
<<http://support.microsoft.com/default.aspx?scid=kb;en-us;299656>>

© Luis Martín García. Salamanca, {luis.mgarc@gmail.com} v.0.2. Salamanca. Julio, 2006. Some rights reserved. Texto licenciado bajo licencia Creative Commons Attribution-ShareAlike 2.1 Spain.