educación IT
CODE YOUR FUTURE

Introducción a la Ciberseguridad

Módulo 5

Algoritmo Hash

Algoritmos de Hash

Se refiere a una función o método para generar claves que representen de manera casi unívoca a un documento, registro, archivo, etc. Un *hash* es el resultado de dicha función o algoritmo.

Estos algoritmos son **unidireccionales**, lo que quiere decir que es posible obtener, en base a un dato, su hash resultante pero, en base a este hash, es imposible obtener el dato original. Esto los diferencia con respecto a los algoritmos de cifrado, debido a que, lo que vemos en el hash, no es el contenido cifrado, sino el resultado de cálculos matemáticos, que no representan a la información original, sino a una *firma* que identifica a la misma. Un hash es, básicamente, como una **huella dactilar**. Nos permite identificar algo, pero no nos permite reconstruir la información completa (en base a una huella dactilar, no podemos conocer los detalles de una persona, como su color de pelo, ojos, altura, etc.).

Una propiedad fundamental del *hashing* es que **si dos resultados de una misma función son diferentes, entonces las dos entradas que generaron dichos resultados también lo son**.

Es posible que existan claves resultantes iguales para objetos diferentes, ya que el rango de posibles claves es mucho menor que el de posibles objetos a resumir (las claves suelen tener en torno al centenar de bits, pero los ficheros no tienen un tamaño límite). A estos casos se los llama ***colisiones***. **Una buena función de hash debe experimentar pocas colisiones**.

Algunos de los algoritmos de Hash más utilizados son ***LM***, ***NTLM***, ***MD5*** y ***SHA***.

Abajo podemos ver los resultados de varias funciones de hash aplicadas a la palabra *cripto*:

```
MD5: ebea37120a7c6f155e0c50f9c92ce351
```

```
SHA1: 7dd5de71de03fc27ea218d0d0bd479c713a39fec
```

Almacenamiento de contraseñas

En los sistemas informáticos, **el almacenamiento seguro de la base de datos de usuarios y contraseñas suele ser un gran desafío**. Imaginemos que debemos contar con una pequeña tabla, que tenga solamente dos campos: *Usuario* y *Contraseña*, como muestra la tabla de la derecha.

Usuario	Contraseña
Admin	ADM123inistrador
Fabian	pepe2020
Manuel	emedemama

Esta base de datos debería estar almacenada en algún lugar lógico (un archivo de texto plano, una base de datos relacional, etc).

En primer lugar, si almacenamos las contraseñas en **texto claro**, debemos saber que, **si alguna persona pudiera leer esta información, sería el poseedor de todas las credenciales de acceso al sistema**. Es importante tener en cuenta que leer los contenidos de un archivo suele ser una tarea relativamente sencilla y que, de una forma u otra, por lo menos, un usuario va a tener que poder leerlos (el que sea el encargado del proceso de autenticación). Por otro lado, los usuarios administradores del sistema van a poder (en la mayoría de los casos) leer todos los archivos del mismo, por lo que conocerían totalmente las claves de acceso. Esto les permitiría hacerse pasar por cualquier usuario, sin que estos lo noten.

Una buena forma de evitar todos estos problemas, es **usar funciones de *hash***, para **almacenar los *hashes* resultantes de las contraseñas**, en lugar de las contraseñas mismas.

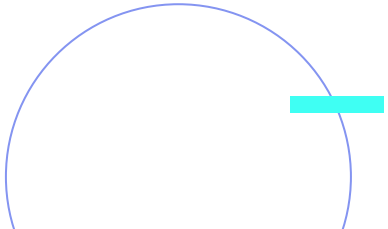
Podríamos formular una nueva tabla, que almacene los hashes, de la siguiente forma:

Usuario	Contraseña
Admin	52e71b10074c9bfae5dee9652c86217d
Fabian	cb13af1d5b72b5466608ee61e67a030a
Juan	d052a70b81c33146b5b456f88bcb37c2
Manuel	65ce8c51bbd6e35527092146d9fbc7ca

(La tabla está almacenando los hashes producidos por el algoritmo ***MD5***).

Podemos observar que, ahora, el **hecho de poder leer la base de datos no nos permite conocer las contraseñas de acceso**.

También podemos notar que, si bien las contraseñas originales tenían diferentes longitudes, los hashes resultantes poseen todos una **longitud de 32 caracteres** (esto es lo que produce el algoritmo MD5).



educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!