Trabajo Práctico Nº 2

Hash

1. Los algoritmos de hash (md5, sha-x, etc.) no se utilizan para cifrar mensajes. ¿Por qué?

   Porque una de las propiedas (requisito) de estas funciones resumen es que sea computacionalmente imposible obtener el texto plano original (entrada de la función hash) a partir del hash generado. La función es *unidireccional*. Por ende no sirve para cifrar un mensaje ya que no se podría descifrar para obtener el mensaje original.

2. Explique conceptualmente la utilidad de algoritmos de hash para:

   a) Autenticación de usuarios.

      Las funciones hash tiene un doble uso en cuanto a la autenticación de usuarios. Por un lado, existen ciertos sistemas criptográficos asimétricos que utilizan una función de hash para generar, a partir del hash del mensaje a enviar, una firma (o rúbrica digital) que será enviada junto con el mensaje cifrado y luego en el destino se descifra esta firma con la clave pública del emisor, obteniendose el hash que luego sera comparado con el hash generado en el emisor al aplicar la misma función hash al mensaje (previamente descifrado). Si los valores hash son iguales, la firma es auténtica y el mensaje integro. Suele utilizarse este método ya que generar una firma digital a partir del mensaje entero suele ser muy lento en los sistemas criptográficos asimétricos.

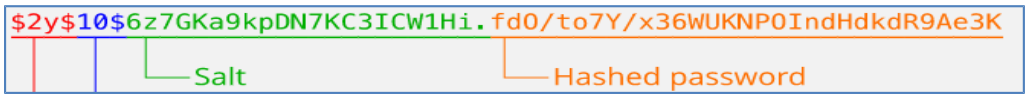
      Otro de los usos de las funciones hash en la autenticación de usuarios es para el almacenamiento cifrado de contraseñas convirtiendolas en texto ilegible, manteniendo de esta manera la confidencialidad de la misma en caso de que alguna persona no autorizada acceda a la base de datos o archivo que contiene las contraseñas. Cuando un usuario intenta acceder al sistema introduciendo su contraseña se genera el hash de esa entrada y se compara con el valor hash almacenado en la base de datos o archivos, si coincide, entonces el usuario esta autenticado.

   b) Comprobación de integridad de archivos.

      Antes de envíar o descargar un archivo se puede generar un hash del mismo en el nodo origen, que sera enviado (ó publicado) junto con el archivo para que pueda ser comparado con el hash que se genere en el nodo destino y determinar la integridad del archivo recibido, esto es, verificar que el archivo no ha sido alterado en la transmisió. Si los valores hash son iguales la integridad del archivo es correcta, si no, el archivo el archivo a sido alterado.

3. ¿Qué es **salt**? ¿Para que se utiliza?

   El salt ("sal") es un plus de seguridad que se le agrega a las funciones hash cuando se utilizan, por ejemplo, para cifrar contraseñas. **Es un número de dígitos alfanuméricos aleatorios que se le agrega al valor hash resultante, ya sea al principio o al final**. De esta manera, los valores hash ya no son los normales, dificultando su traducción (ó decodificación) por medio de la utilización de ***rainbow tables*** (se explica en el próximo ejercicio), ya que se deberá probar no solo con cada hash, sino también con cada *salt* y sus combinaciones.



4. Explique brevemente qué es una ***rainbow table***.

   Una **rainbow table** (ó tabla *arco iris*) es una tabla de consulta (de gran tamaño, del orden de los gigabyte) que almacena combinaciones de palabras y valores hash precalculados, y que sirve para descifrar una contraseña cifrada con alguna función hash. Existen tablas rainbow publicadas en internet para la mayoria de los algoritmos más populares de hash (MD5, SHA-1, SHA-256, etc.). Aunque es normal creer que estas tablas se componen de pares texto plano/hash, en realidad no es así, debido a que existen mayúsculas, minúsculas, números, caracteres especiales, etc. Esto da como resultado muchísimas combinaciones y la tabla ocuparía mucho espacio (hasta el orden de los terabytes) e incluso sería inviable desde el punto de vista de la busqueda en estas tablas. Por eso, estas se generan a partir de pares “palabra inicial” y “palabra final” de un proceso que realizan las tablas rainbow justamente para crear su contenido. Dicho proceso se lleva a cabo utilizando un algoritmo de resumen y otro de reducción; el algoritmo de resumen es simplemente la función hash soportada por la tabla, las tablas son específicas, si queremos crackear un hash md5 tendremos que usar una tabla que se haya creado con ese algoritmo de resumen. Por otro lado, el algoritmo de reducción se aplica sobre hashes para obtener nuevas palabras o combinaciones de caracteres.

   El proceso se inicia aplicando la función hash a una palabra, que se denomina *palabra inicial*. El resultado lógicamente será su representación en formato hash, al cual le aplicaremos el algoritmo de reducción para generar una nueva palabra y continuar el proceso. Esto mismo se repetirá unas 40.000 veces. Al final, se guardará en la tabla rainbow únicamente la palabra inicial y la última que se ha generado, es decir, la palabra final. A esto se lo conoce como dupla, por cada una se realiza todo este proceso, y como se menciono anteriormente, las tablas están compuestas por muchísimas duplas (pares).

5. Programa de login/logout. Almacenamiento de contraseñas en hash. Implementación en PHP.

   Implementación de un programa muy simple para demostrar el uso de *funciones hash* para el almacenamiento seguro de contraseñas en un sistema de autenticación de usuarios. El programa esta hecho con PHP y se utilizaron dos funciones, que forman parte de la librería estandar del lenguaje, para la generación y validación de valores hash y contraseñas:

   - password_hash (...): para generar un hash a partir de una contraseña.
   - password_verify (...): para comprobar que una constraseña coincida con un hash dado.

   *Git: jldevia/**ARS2018** – **TP2/***

   El ingreso a la aplicación se efectua por medio de la pagina ***login.php***.