Control de integridad

Departamento de Sistemas Telemáticos y Computación (GSyC)

http://gsyc.urjc.es

Marzo de 2012

Función hash

Técnica básica para garantizar integridad de un mensaje

```
hash:
a mess, jumble, or muddled.
to chop into small pieces; make into hash; mince.
to muddle or mess up.
```

- Función que, a partir de un bloque arbitrario de datos (*message*), genera de forma determinista un *valor hash*, aka *message digest*, aka *digest*.
  Este valor hash identifica de forma prácticamente unívoca al mensaje, de forma que un cambio en el mensaje, aunque sea pequeño, provoque un cambio en el valor hash
- Es posible que dos mensajes distintos generen el mismo valor hash, aunque muy difícil

Función hash ideal:

- Fácil de generar
- Muy difícil generar el mensaje a partir del hash
- Muy difícil modificar el mensaje manteniendo el hash
- Muy difícil encontrar dos mensajes con el mismo hash

Ejemplos de funciones hash: MD2, MD4, MD5, SHA-1, SHA-2

- En el año 1996 aparecieron vulnerabilidades en MD5. En 2005, en SHA-1
- SHA-2 se considera seguro actualmente
- Se espera que SHA-3 sea definido a finales de 2012

```
koji@mazinger:~$ md5sum ubuntu-11.10.iso
c396dd0f97bd122691bdb92d7e68fde5  ubuntu-11.10.iso

koji@mazinger:~$ sha1sum ubuntu-11.10.iso
8492d3daf0c89907c4301cb2c72094fe59037c76  ubuntu-11.10.iso

koji@mazinger:~$ sha224sum ubuntu-11.10.iso
b070d093af7e933cedf6c21cf5a5b71ff7eb29946b9fc2a21d609ca0  ubuntu-11.10.iso
```

cmp

- Con el *hash* podemos saber si un fichero ha cambiado. Pero no qué ha cambiado
- La orden *cmp* indica el primer byte que ha cambiado en dos ficheros

```
koji@mazinger:~$ cmp a.pdf b.pdf
a.pdf b.pdf son distintos: byte 63401, línea 716
```

diff

En caso de que los ficheros estén en formato de texto plano, *diff* indica cuáles son los cambios, línea a línea
Supongamos los ficheros `a.txt` y `b.txt`

```
a.txt:          b:txt
-----------------------
rojo            colorado
amarillo        ámbar
verde           verde
```

- Las líneas que estén en `a.txt`, pero no en `b.txt` se muestran precedidas del signo de menor
- Las líneas que estén en `b.txt` pero no en `a.txt`, se muestran precedidas del signo de mayor
- Las que no cambian, no se muestran

```
koji@mazinger:~$ diff a.txt b.txt
1,2c1,2
< rojo
< amarillo
---
> colorado
> ámbar
```

- `1,2c1,2` representa el número de línea donde sucedió el cambio, podemos ignorarlo
- En entornos de programación, esto sirve para ver cambios en programas y distribuir *parches* (la orden *patch* puede construir `b.txt` a partir de `a.txt` y la salida de `diff`)

En administración de sistemas, nos sirve para saber qué ha cambiado entre dos instantes de tiempo
P.e.

- En el instante t_0 ejecutamos `ls /directorio > t0.txt`
- En el instante t_1 ejecutamos `ls /directorio > t1.txt`

Comparamos con `diff t0.txt t1.txt`

- Los ficheros desaparecidos se muestran precedidos del signo de menor
- Los ficheros nuevos se muestran precedidos del signo de mayor

Podemos aplicar esto mismo a `ps`, `netstat`, `who`, ...
Es útil para

- Automatizar mediante scripts
- Observar *a ojo* un número elevado de elementos