Unidad 2: Seguridad pasiva

[**Introducción**](#_heading=h.30j0zll) **2**

[**Copias de seguridad**](#_heading=h.1fob9te) **3**

[*Tipos de copias de seguridad*](#_heading=h.3znysh7) *3*

[Recomendación sobre el tipo de copia a efectuar](#_heading=h.2et92p0) 3

[Ejemplo de planificación de copia de seguridad](#_heading=h.tyjcwt) 4

[*Respaldo de la información en GNU/Linux*](#_heading=h.3dy6vkm) *4*

[Copias de seguridad](#_heading=h.1t3h5sf) 4

[TAR](#_heading=h.4d34og8) 4

[CRONTAB](#_heading=h.2s8eyo1) 5

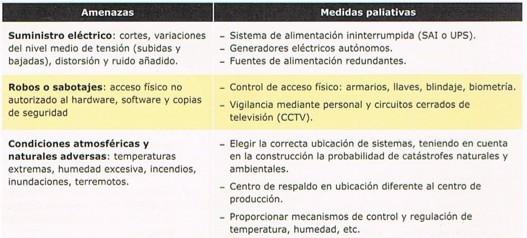
[Ejemplo de script](#_heading=h.17dp8vu) 6

[Enlaces duros y enlaces simbólicos](#_heading=h.y0ntp7fy2zyw) 8

# Introducción

Un sistema seguro es aquel que intenta no dejar ningún cabo suelto, ni siquiera las respuestas a las amenazas menos verosímiles; por eso, el objetivo de las técnicas de seguridad pasiva es minimizar los efectos o desastres causados por un accidente, un usuario o un malware a los sistemas informáticos; es decir, dando por hecho que las amenazas pueden cumplirse, estas técnicas implantan medidas para que el impacto de un incidente —más o menos inesperado— sea el menor posible.

He aquí algunos tipos de amenazas con algunas de las medidas que les podrían hacer frente:



Las consecuencias de que las amenazas se cumplan pueden ser:

* Pérdida o mal funcionamiento del hardware.
* Falta de disponibilidad de servicios.
* Pérdida de información.

# Copias de seguridad

¿Qué ocurre si un miembro de nuestra organización borra datos importantes de ésta? ¿Y si un malware nos borra información relevante? Las copias de seguridad o backups son réplicas de datos que nos permiten recuperar la información original en caso de ser necesario; esas copias podrán ser almacenadas en: dispositivos extraíbles —CD, DVD, pen drive…—; la propia máquina; discos de red…

## Tipos de copias de seguridad

Para llevar a cabo las copias de seguridad, lo adecuado es utilizar métodos estructurados, de tal forma que sea fácil encontrar la copia que nos interesa para ser restaurada. Si se combinan los tipos de copias de seguridad adecuadamente, se conseguirá —además de asegurar en gran medida los datos críticos — que encontrar la restauración a realizar sea una tarea sencilla; en función de la cantidad de archivos que se quieran poner a salvo, podemos dividir las copias en tres tipos:

* Total: copia total de todos los archivos y directorios del sistema.
* Incremental: copia de los archivos que han cambiado desde la última copia —sea ésta del tipo que sea—.
* Diferencial: copia de los archivos que han cambiado desde la última copia total.

### Recomendación sobre el tipo de copia a efectuar

Evidentemente, si tuviéramos un sistema informático con potencia y capacidad ilimitados, haríamos copias totales cada poco tiempo; sin embargo, ese estadio ideal no suele darse y el administrador de la red debe idear un método que consiga la máxima seguridad de los datos, pero teniendo en cuenta las restricciones del sistema; de ahí que el administrador de la red deba hacer una planificación concienzuda.

Si el volumen de datos de la copia de seguridad a realizar no es muy elevado, podría ser una buena opción realizar siempre copias totales; así, si hubiera que hacer una restauración, sólo habría que recuperar la última copia.

En cambio, si el volumen de datos de la copia de seguridad es muy elevado, pero el volumen de datos que se modifica no, una opción recomendable sería hacer una primera copia total y, posteriormente, copias diferenciales; de esta forma, en caso de desastre, habría que recuperar la copia total y la última copia diferencial.

Finalmente, si el volumen de datos de nuestra copia de seguridad es muy elevado y el volumen de datos que se modifica también lo es, las copias diferenciales ocuparán mucho espacio; de ahí que lo más práctico sea realizar una primera copia total y, posteriormente, realizar siempre copias incrementales. Así, si se da una pérdida de información importante, habrá que restaurar la copia total inicial y todas las incrementales realizadas desde ella.

En cualquier caso, se recomienda hacer las copias de seguridad cuando menos actividad haya en el sistema, para interferir lo menos posible en el funcionamiento adecuado de éste.

Las soluciones que hemos aportado tienen tanto ventajas como inconvenientes, como refleja esta tabla:

| **Método de copia** | **Espacio de almacenamiento** | **Velocidad de copia** | **Restauración** | **Copia recomendada** |
| --- | --- | --- | --- | --- |
| ***Total*** | Máximo | Muy lento | Muy simple | Pocos datos a copiar |
| ***Total  + Incremental*** | Mínimo | Rápido | Compleja | Muchos datos que cambian frecuentemente |
| ***Total  +******diferencial*** | Intermedio | Lento | Sencilla | Datos cuya velocidad de cambio es moderada. |

### Ejemplo de planificación de copia de seguridad

Hay organizaciones con gran poder económico que deciden utilizar un método de copias de seguridad mixto; por ejemplo:

* Todos los días 1 de cada mes, a las 23:00 horas: copia de seguridad total.
* Todos los viernes a las 23:00 horas: copia de seguridad diferencial desde la copia del día 1.
* Todos los días —excepto los viernes y el día 1— a las 23:00 horas: copia de seguridad incremental desde la copia del día anterior.

Con este método, en caso de desastre, habría que recuperar la copia total del mes, la última diferencial y todas las incrementales desde la última diferencial.

## Respaldo de la información en GNU/Linux

Una de las formas de hacer copias de seguridad bajo GNU/Linux es usar el empaquetador de archivos *tar* junto con el automatizador de tareas *cron*.

### Copias de seguridad

#### TAR

A continuación, un ejemplo de un comando para empaquetar archivos en GNU/LINUX:

tar -vcf nombre\_archivo.tar nombre\_carpeta\_a\_empaquetar, donde:

* *v (verbose)*: permite obtener una descripción de los archivos empaquetados/desempaquetados
* *c (create)*: crea un archivo *tar*
* *f (file)*: indica que se dará un nombre al archivo *tar*
* *--newer=fecha*: realiza un empaquetado incremental teniendo en cuenta qué archivos han sido modificados desde la fecha que se le indique.

Este otro ejemplo serviría para desempaquetar archivos:

tar -tvxf mi\_archivo.tar, donde:

* *t*: ver el contenido —sin extraer—.
* *x*: (extract/extraer) extrae los archivos en la carpeta que contiene el tar

#### CRONTAB

cron es un administrador regular de procesos en segundo plano (demonio) que ejecuta procesos a intervalos regulares. Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero *crontab*; éste es un simple archivo de texto que guarda una lista de comandos a ejecutar en un tiempo especificado por el usuario; *crontab* verificará la fecha y la hora en que se debe ejecutar el script. Cada usuario puede tener su propio archivo crontab, editando el fichero */etc/crontab*.

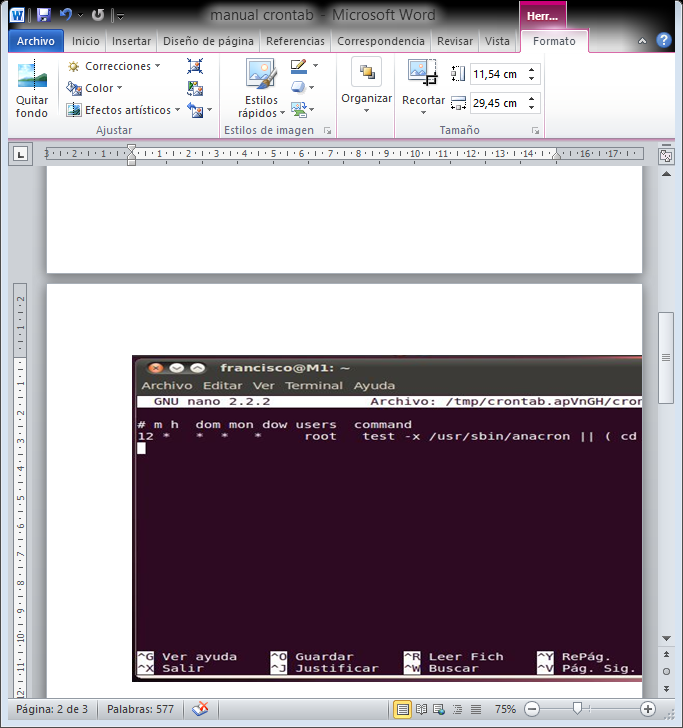
La sintaxis del comando a ejecutar es:

crontab [-e] -l [-r] [usuario], donde:

* *e (edition)*: indica la edición del *cron;* abrirá el archivo */etc/crontab*.
* *l (list)*: ver las tareas programadas en el archivo *cron*.
* *r (remove)*: borrar un archivo cron.

Si no se especifica el usuario, el comando se ejecutará para el usuario cuya sesión está en marcha.

Cuando accedemos al archivo *crontab,* nos encontramos con una pantalla con una línea semejante a ésta:

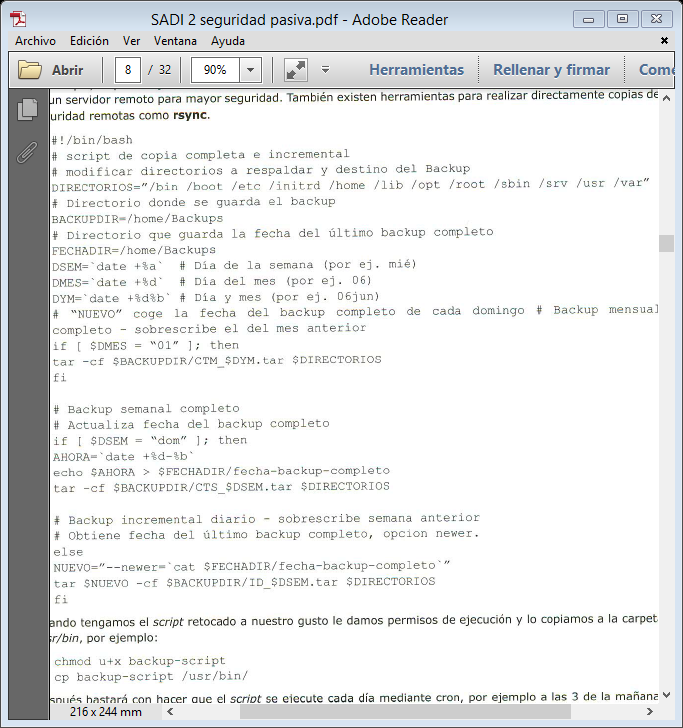


Esa línea, que no deja de ser un comentario, nos indica en qué orden hay que introducir la información; por cada acción que se quiera programar, habrá que añadir una línea atendiendo a lo siguiente:

* *m*: corresponde al minuto en que se va a ejecutar el script, el valor va de 0 a 59. El carácter *\** actúa como comodín, indicando cualquier valor de los posibles.
* *h*: la hora exacta; se maneja el formato de 24 horas y los valores van de 0 a 23, siendo 0 las 12:00 de la medianoche. El carácter *\** actúa como comodín, indicando cualquier valor de los posibles.
* *dom*: hace referencia al día del mes. El carácter *\** actúa como comodín, indicando cualquier valor de los posibles.
* *mon*: hace referencia al mes. El carácter *\** actúa como comodín, indicando cualquier valor de
* *dow*: hace referencia al día de la semana; puede ser numérico —de 0 a 7, donde 0 y 7 son domingo— o pueden indicarse las 3 primeras letras del día en inglés: mon, tue, wed, thu, fri, sat, sun. El carácter *\** actúa como comodín, indicando cualquier valor de los posibles.
* *users*: define el usuario que va a ejecutar el comando; puede ser root u otro usuario diferente siempre y cuando tenga permisos de ejecución del script. El carácter *\** actúa como comodín, indicando cualquier valor de los posibles.
* *command*: se refiere al comando o a la ruta absoluta del script a ejecutar.

#### Ejemplo de script

Lo habitual es que el/los comando(s) *tar* a ejecutar formen parte de uno o varios scripts. Veamos un script de ejemplo en el que se hacen: una copia de seguridad total el día 1 de mes; una copia de seguridad total todos los domingos que sobreescribe la del domingo anterior; copias diferenciales diarias que sobreescriben las de la semana anterior.



Deberemos darle al script permisos de ejecución para los usuarios deseados:

*chmod u+x nombre-script  
cp nombre-script /usr/bin*

Pasos para poner en marcha el script una vez escrito:

* Lanzar el script hasta que se ejecute sin errores de compilación; para ello, ejecutaremos el comando: *./nombre\_del\_script*
* Hacer que el script se ejecute mediante cron a la hora deseada.
* Probar nuestro sistemas de copias de seguridad; al estar en un entorno de prueba, se recomienda jugar con el cambio de la fecha y la hora del sistema.

Si quisiéramos restaurar el sistema, deberíamos extraer el fichero adecuado del directorio */home/Backups* mediante el comando *tar*.

### ***Enlaces duros y enlaces simbólicos***

En GNU/LINUX se utiliza un sistema de archivos basado en inodos (en los sistemas Windows existe algo similar). Veamos con mayor profundidad cómo funciona este sistema: a cada archivo o directorio se le asigna un número llamado inodo. Cada uno de estos números se corresponde con una entrada en la tabla de inodos, en la cual se almacena toda la información representativa del archivo: propietario, permisos, tipo de archivo… Lo que no se guarda en esa tabla es el propio nombre del archivo. Éste se almacena en el directorio al que pertenece y es aquí donde se asocia con el inodo correspondiente. De esta manera, se pueden tener varios nombres haciendo referencia a un mismo archivo, algo útil cuando se quiere trabajar con un mismo archivo en varios directorios diferentes. Cuando hay varios nombres para un archivo, a cada nombre se le llama “enlace duro”. Si tenemos dos enlaces duros de un fichero, veremos lo que parecen dos ficheros; sin embargo, en realidad es uno solo.

Los enlaces simbólicos son otro tipo de enlace que permite que un fichero haga referencia a otro. Se asemejan a los accesos directos de Windows.

La ventaja de los enlaces duros es que permiten un mayor aseguramiento de la información. Así, un archivo no es eliminado hasta que todos sus enlaces son eliminados. Supongamos que somos los administradores de una red y hay un archivo de vital importancia en el servidor. Además de las pertinentes copias de seguridad —que hay que tener planificadas, con el coste que supone—, podría resultar conveniente crear un enlace directo de ese archivo en un directorio al que no se suela acceder —para asegurarnos de que no lo tocamos—. Cada vez que se modifique nuestro importante archivo, la modificación se verá en ambas ubicaciones, sin necesidad de planificar unas copias temporales. Esto no es como una mera copia de un archivo, pues en este último caso, el archivo “copia” no se modificaría automáticamente al modificar el original.

Por su parte, si el archivo al que hace referencia un enlace simbólico es eliminado, el enlace simbólico se romperá. Sin embargo, es muy fácil saber a qué archivo hace referencia un enlace simbólico, pero no a cuál hace referencia un enlace duro —al visualizar los atributos del simbólico, nos aparece explícitamente a qué archivo “apunta”, pero no al visualizar los del duro—. Además, si modificas uno de los enlaces duros, se te modifican todos. Eso —que es una ventaja a la hora de asegurar la integridad de los datos, como se ha explicado en el párrafo anterior— pero puede ser una desventaja si no se conoce bien la estructura de los archivos. Imaginemos que entra una nueva persona como administradora a sustituirnos. Moviéndose por el sistema de archivos, ve dos con diferente nombre y mismo contenido. Le interesa mantener el contenido de uno de ellos, por lo cual modifica uno y el otro lo deja como está. Pero no se había percatado de que eran enlaces duros, con lo cual se habrán modificado ambos ficheros.

La sintaxis para crear enlaces duros es:

*ln nombre\_archivo nombre\_enlace\_duro\_nuevo*

La sintaxis para crear enlaces simbólicos es:

*ln -s nombre\_archivo\_original nombre\_archivo\_apuntador*