SEGURIDAD BÁSICA

- **Propietario real y efectivo:** el propietario real, se emplea para conocer quien ha ejecutado el proceso mientras que el propietario efectivo, se usa para obtener autorización de acceso a los recursos. Por defecto, cuando un usuario ejecuta un archivo, el proceso resultante tiene como propietario efectivo el usuario que lo ha ejecutado.

- **Setuid y setgid:** mecanismo que permite ejecutar ficheros con permisos elevados, normalmente como root. Cuando se ejecuta un fichero com setuid activado, el UID *efectivo* del proceso resultante será el propietario del fichero ejecutable, en lugar del UID del proceso que ejecuta el programa. El UID real no se ve afectado. Lo mismo pasa con setgid pero con el GID.

- Las cuentas de los usuarios del sistema (incluyendo sus contraseñas cifradas con una función hash) se encuentran el el fichero **/etc/passwd.** Formato de línea: *login:passwd:UID:GID:info:home-dir:shell.*

- Cuando se crea un nuevo usuario, los ficheros de inicio se copian en el directorio **/etc/skel.**

- El fichero **/etc/profile** es el primero en leerse automáticamente por bash cuando este es invocado por login. En este fichero, se configuran muchas de las variables de bash (PATH, USER…). En resumen, en él se incluyen las variables generales del sistema.

- En caso de querer **crear una cuenta de usuario** de forma manual, habría que seguir los siguientes pasos: editar el fichero /etc/passwd y añadir una nueva línea (x en la contraseña si usamos shadow y modificar el fichero /etc/shadow), editar /etc/group para añadir un nuevo grupo o añadir el nuevo usuario a un grupo ya existente, crear el directorio del usuario y copiar los ficheros de /etc/skel, usar chown, chgrp y chmod para fijar el usuario, grupo y permisos del directorio, por último fijar la contraseña con passswd.

- Si se quiere **eliminar una cuenta de usuario** de forma manual, hay que inhabilitar la cuenta impidiendo el acceso (temporalmente cambiando el campo contraseña de /etc/passwd a * o definitivamente borrando las entradas del usuario de /etc/passwd y /etc/shadow), hacer backup de los ficheros del usuario y por último, eliminar los ficheros de usuario.

- Diferencia entre **su** y **sudo**: su permite cambiar de una cuenta a otra (solo cambia el UID y GID) y sudo ejecuta un solo comando que se le pasa como parámetro con permisos de superusuario.

- **Criptografía de clave secreta:** se emplea una única clave para cifrar y descifrar. La clave debe ser conocida por ambos, el emisor y receptor, por lo tanto, la mayor dificultad de esta aproximación es la distribución de la clave. Pueden emplearse dos métodos, cifras de Stream y de Bloque.

- **Criptografía de clave pública:** se emplean dos claves matemáticamente relacionadas, aunque el conocimiento de una de ellas no permite determinar fácilmente la otra. Una de las claves se denomina clave pública y puede darse a conocer a todo el mundo. La otra se denomina clave privada, y nunca se da a conocer a nadie. Ejemplo: *Fernando Garcia Vallés* quiere enviar un mensaje a *su puta madre*, *Fernando Garcia Vallés* cifra el mensaje empleando la clave pública de *su puta madre* y *su puta madre* descifra el mensaje empleando su clave privada. (RSA)

- **Cifrado de hash:** son algoritmos que calculan un valor de longitud fija a partir de los datos originales. A partir de dicho valor, es imposible averiguar los datos a partir de los que se generó.

- **SALT**: protección de contraseñas frente a ataques basados en diccionario y "tablas rainbow". Comprende bits aleatorios que se usan como una de las entradas de la función hash.

- **PGP**: se emplea para firmar o cifrar e-mails

- **Métodos de autentificación en SSH:** usar criptografía de clave pública para verificar la identidad de la máquina cliente, usar criptografía de clave pública para verificar la identidad del usuario (método más seguro) ó usar el método normal login/password. El método se especifica en */etc/ssh/sshd_config.*

Importante; la primera vez que se conecta e un servidor, este envía su clave pública y el usuario ha de aceptarla (o no). Las claves/públicas y/o privadas se generan con ssh-keygen (rsa, dsa, ed25519).

- SSH tiene otra utilidad muy interesante que es que pude hacer **conexiones TCP** de forma segura "tunelizandolas" por un canal SSL.

- **Firewall (filtrado de paquetes):** es la herramienta más básica en seguridad a nivel de red. Limita el tipo de tráfico que puede pasar a través de el, utilizando la información de la cabecera del paquete. Se especifican las direcciones de destino, puertos, y protocolos que son aceptables, y el firewall descarta los que no cumplan con la especificación dada

- **Iptables** aplica cadenas de reglas ordenadas a los paquetes. A los conjuntos de cadenas se les denomina tablas, cada una tiene una misión distinta dentro del firewall (filter, nat, raw, mangle).

- La tabla **filter**, es la tabla por defecto. Contiene tres cadenas por defecto: FORWARD, INPUT, OUTPUT, las reglas en FORWARD se aplican a paquetes que llegan a un interface y deben salir por otro y las reglas en INPUT y OUTPUT se aplican a paquetes que entran o salen del host, respectivamente. Cada regla tiene un objetivo (ACCEPT, DROP, REJECT, LOG, QUEUE, RETURN, ULOG) que determina lo que hacer con los paquetes emparejados con una regla de la cadena y cuando un paquete cumple una regla, ya no se chequearán mas reglas.

- **Firewall con estado**, permite hacer seguimiento de conexiones mediante la extension *state.* Mantienen en memoria el estado de cada conexión y determinan para cada paquete si pertenece a una conexión en curso, si es de una nueva conexión, o es erróneo.

- La tabla **NAT** es un mecanismo para remapear direcciones IP. Útil para que varios nodos dentro de una red privada puedan compartir una IP pública, NAT oculta la organización de la red interna y previene el acceso a los nodos desde el exterior. Las cadenas de NAT son; PREROUTING, POSTROUTING, INPUT, OUTPUT y los objetivos DNAT (cambia la dirección destino del paquete p), SNAT (cambia la dirección origen del paquete P) y MASQUERADE (igual que SNAT pero con ips dinámicas).

CONFIGURACIÓN BÁSICA DEL SISTEMA

- A**rranque** de un ordenador (tipo PC): El **iniciador ROM** realiza tres funciones principales: comprueba el sistema, detectando sus características y comprobando su funcionamiento, lee y almacena en memoria el programa cargador del S.O. y por último, pasa el control al cargador del S.O., saltando a la dirección de memoria donde lo ha almacenado. El **programa cargador** está en los primeros sectores del disco (Master Boot Record) y con un tamaño prefijado. Es el encargado de cargar el núcleo (o kernel) del S.O. y pasarle el control. Posteriormente, el kernel hará una comprobación del hardware del sistema y lanzará el proceso Init. El proceso Init termina el proceso de arranque, dejando el sistema en modo multiusuario preparado para que los usuarios trabajen en él.

- **Fichero inittab:** es el fichero de configuración para el proceso Init. Cada línea, es una. Entrada formada por cuatro campos: identificador de 2 letras, nivel de ejecución al que debe ejecutarse el proceso, acciones (cómo y cuando ejecutar el proceso) y path de acceso al programa a ejecutar.

- **Modo monousuario:** Estado del sistema definido para realizar tareas administrativas y de mantenimiento, que requieren un control completo y no compartido del sistema. Sólo realiza el montaje del sistema de ficheros raíz (/),. El problema del modo monousuario no tiene ninguna protección.

- **Systemd,** es un conjunto de bloques básicos para la construcción de un sistema Linux. Tiene tres funciones generales; gestor del sistema y de servicios, plataforma para desarrollo de software y es el "pegamento" entre aplicaciones y el Kernel. Sin embargo, Systemd puede generar "polémica" puesto que viola la "filosofía Unix": hacer una sola cosa, y hacerla bien.

- **Paquete:** archivo comprimido que contiene los archivos de una aplicación. Incluye meta-datos; descripción, versión, requisitos previos, scripts de instalación, etc. Además, gestiona dependencias. Dos tipos principales, RPM y DEB.

- **Repositorios:** Almacenes de paquetes (online, CD/DVD, etc.) donde se pueden buscar e instalar paquetes ofrecidos por los desarrolladores de la distribución o de la aplicación.Casi todas las distribuciones instalan aplicaciones, bibliotecas, etc. desde repositorios. Buscan la simplificación de la instalación de paquetes, descargando e instalando automáticamente dependencias. Tipos; YUM, APT-GET y APT.

- El **kernel** es el núcleo del sistema operativo, abstrae el hardware del sistema mediante una API que permite; manejar dispositivos, Gestión y comunicación de procesos, de memoria y de entrada/Salida. En función del modo de ejecución de los servicios del SO se distinguen distintos tipos, Microkernels (servicios en modo usuario) y Kernel monolítico (servicios en modo supervisor).

- **Módulo:** bloque de software que se puede cargar/descargar del kernel dinámicamente.

- **Driver:** bloque de software que permite al kernel hablar con dispositivos, pueden ser módulos o no.

ALMACENAMIENTO

- **Enlaces:** existen dos tipos de enlaces, duros. Y blandos. Los enlaces duros son referencias que asocian un nombre a un fichero, el sistema de ficheros guarda el número de referencias a cada fichero y no permite su eliminación total hasta que el último enlace se ha borrado (no puede haber enlaces duros entre diferentes sistemas de ficheros). Los enlaces blandos, apuntan a otro fichero por su nombre. Pueden crear bucles entre si (ciclos) y apuntar a ficheros no existentes y pueden establecerse entre sistemas de ficheros distintos.

- Diferencias entre **sockets y tuberías:** ambos métodos son mecanismo de comunicación entre procesos, pero las tuberías son unidireccionales y los sockets bidireccionales. Además, múltiples procesos pueden escribir a la misma tubería pero receptor no podrá distinguirlos, en cambio, los sockets permiten la identificación del proceso origen. Por último, en caso de querer usar alguno de estos dos métodos en un script, las tuberías son más cómodas.

- **Sticky bit:** flag que permite modificar los permisos de acceso según el tipo de fichero. Actualmente, en ficheros regulares ya no tiene ningún efecto pero en directorios, previene a usuarios el borrado/ renombrado de ficheros salvo que sean dueños del fichero o del directorio (se usa en directorios donde todo el mundo puede escribir). Se puede activar con *chmod +t /tmp.*

- **Magic number:** es una firma en forma de cadena de bits que los ficheros suelen contener al principio para representar su tipo. Es usado por el comando **file** para identificar ficheros.

- Un **sistema de ficheros** define la manera de almacenar, recuperar, actualizar y identificar datos en un soporte físico.

- **/etc,** guarda "ficheros de configuración", estos ficheros deben ser estáticos y no binarios ejecutables (fstab, hosts, sytemd…).

- **/dev,** ubicación para los ficheros de dispositivo. En la actualidad hay gestores dinámicos de dispositivos como udev, tiene 3 componentes: biblioteca libudev, demonio de gestión de /dev y el comando udevamd. La identificación de dispositivos realiza mediante **UUID** (números de 32 dígitos), udev permite establecer las correspondencias entre UUID y dispositivos en el directorio /dev.

- **/tmp,** permite a los programas disponer de una carpeta donde almacenar ficheros temporales. Al arrancar una máquina, /tmp es borrado.

- /var, contiene ficheros que varían con frecuencia; incluyendo directorios y ficheros spool (datos impresión), datos de administración, logs y ficheros temporales.

- **/boot,** contiene todo lo necesario para el proceso de arranque excepto algunos ficheros de configuración, se almacena todo los datos que se usan antes que el kernel pueda ejecutar programas en modo usuario.

- **/mnt,** ofrece un directorio donde se pueda montar sistemas de ficheros de manera temporal y **/media,** debe utilizarse como cabecera de punto de montaje para elementos extraíbles.

- El proceso de **montaje/desmontaje** consiste un unir/liberar un sistema de ficheros al árbol de directorios. Para montar, el usuario debe conocer la partición/volúmen lógico donde se encuentra el sistema de ficheros a montar.

- **/etc/fstab,** fichero de configuración que contiene sistemas de ficheros montados habitualmente. Formato: # <file system> <mount point> <type> <options> <dump> <pass>.

- **/etc/mtab,** muestra todos los sistemas de ficheros montados en el sistema. Contiene todos los sistemas de ficheros montados de /etc/fstab y aquellos que el usuario haya montado manualmente.

- **Particionar** es el proceso para dividir un disco en partes lógicas de tamaño conocido, 2 herramientas principales: fdisk y parted (fdisk sólo trabaja con particiones MBR y parted también maneja particiones GUID).
- En un sistema de ficheros convencional, el almacenamiento es muy rígido: capacidad fija, imposible "mover datos" con el sistema en marcha, no es posible repartir los datos entre discos para aumentar el ancho de banda, etc. Como solución, se crea una capa de abstracción (**LVM**) entre las particiones (almacenamiento físico) y los sistemas de ficheros (volúmenes lógicos).

- Las **cuotas de disco**, permiten limitar el espacio disponible para usuarios y grupos en sistemas de ficheros. Existen dos tipos de limitaciones, dura (nunca puede ser excedida) y blanda (puede ser temporalmente excedida). Ambos tipos de limitaciones se aplican de manera independiente sobre bloques e i-nodos.

- **RAID,** es un mecanismo para crear dispositivos lógicos virtuales uniendo 2 o más dispositivos de bloque reales, permite, por ejemplo, combinar varios discos en un único sistema de ficheros. Puede ser implementado mediante hardware o software.

- La **copias de seguridad,** buscan proteger la información de los usuarios es una de las tareas más críticas de todo administrador de sistemas. Principios; cobertura (¿Cuáles son los ficheros que hay que proteger?), periodicidad (¿Cuándo deben hacerse las copias?), separación (¿Dónde deben guardarse las copias?), historia. (¿Dónde deben guardarse las copias?), integridad (¿Son válidos los datos a copiar?), etc.

- Un sistema de **backup** cuenta con cuatro elementos; planificador, comandos de copia/restauración, medios de almacenamiento y datos de origen. Cabe destacar que el planificador establece 2 políticas, cuáles son los datos de origen a copiar y cuándo deben hacerse las copias (cron). Existen dos tipos de backup en primera copia, completo y parcial, y otros dos en siguientes copias, increméntale y diferencial. Existen otros 2 tipos de comandos de copia/restauración, según nivel trabajo: bloques/sistemas (dump y Restore) de fichero y ficheros.

GESTIÓN DE PROCESOS

- Se puede interactuar con procesos vivos enviándoles **señales** mediante el comando kill.

- La prioridad de ejecución, **niceness**, de un proceso determina cuanto tiempo de CPU recibe. El rango de prioridades va entre -20 (máxima prioridad) y +19 (minima prioridad) en linux, al ejecutarse un proceso, este hereda la prioridad del padre. El dueño de un proceso sólo puede reducir la prioridad, root puede reducirla y aumentarla (comando nice).

- Para crear trabajos que se ejecuten periódicamente se utilizan el daemon **cron** y el comando crontab. Crontab permite configurar los procesos periódicos y cron se encarga de su ejecución. **Anacron** se usa para ejecutar comandos periódicamente, con una frecuencia especificada en días. Al contrario que cron, no asume que la máquina está corriendo permanentemente (se puede usar en máquinas que no están corriendo las 24 horas)