Sesión 1. (SSH Ubuntu)

telnet: no encripta por lo que se creó

ssh: (Secure Shell)

SSH	Instalado		Arch config	Root ?	FW
Ubuntu	NO	ssh/sshd	No comentario	NO	NO
CentOS	SI	sshd	comentario	SI	SI

Comandos:

- **ps -xf**: muestra todos los procesos en ejecución.
  - **x**: procesos que no están siendo ejecutados por ninguna terminal (también).
  - **f:** formato "forest" ("bosque") de familias en forma de árbol.
  - **Af**: todos los procesos
- **ssh <ipservidor> -l <usuario> -v**: conecta con un servidor remoto a través de ssh.
  - **l**: usuario.
  - **v**: verbose (muestra información extra).
- **apt shearch <cadena>**: busca los paquetes que se pueden instalar con apt.
- **apt install <paquete>**: instala el paquete indicado.
- **tasksel:** abre la interfaz de instalación de servicios que lanza Ubuntu durante su instalación.
- **ssh-keygen**: genera un par de claves pública y privada.
- **ssh-copy-id <usuario>@<ipservidor> -p <puerto>**: copia la clave publica en el servidor destino. (necesario usuario y contraseña a no ser que ya haya sido configurado para acceder con llaves previamente).

Ubuntu:

- Comprobamos que ssh no está instalado con **ps -xf |grep ssh**
- **sudo tasksel** ó con **apt install openssh-server** (es mejor porque vemos las dependencias).
- Como queremos configurar ssh vamos a ver la config por defecto con **ps -Af | grep ssh**
- Vemos que hay un servicio sshd, el demonio de ssh que siempre está escuchando.
- Tendremos que conectarmos a ese servidor sshd con el cliente ssh.
- Los archivos de configuración están en ***/etc/ssh/ssh_ config*** *y* ***etc/ssh/sshd_ config***
- Probamos usando **ssh localhost**
- Ahora provamos conexión con ssh desde la máquina anfitriona
- **shh <ip> -l <nombre user>**

CentOSt:

- En **systemctl status sshd** para ver si funciona
- Copiamos la config antigua **cp /etc/ssh/sshd_config /etc/ssh/sshd_config_orig**
- Editamos el archivo con **vi la linea de PermitRootLogin**
- Restauramos el servicio **systemctl restart sshd**
- Ahora vemos que no podemos acceder con root
- Ahora **cambiamos el puerto editando el archivo de configuración** a 22022, reiniciamos servicio y probamos.
- Ahora vamos a iniciar sesión sin contraseña.

Claves publicas en el cliente:

- Generamos la llave pueblica en Centos o en la anfitriona para ubuntu con ssh-kaygen (sin contraseña)
- **ssh-keygen**
- Copiamos la clave con **ssh-copy-id<usuario>@192,168,56,105 -p 22022** pregunta usuario
- Comprobamos que se copió bien y nos intentamos conectar
- **ssh -p 22022 <usuario>@192,168,56,105** y no pide contraseña.

Claves publicas en el servidor:

- Modificamos el acceso para que solo se puedan conectar si tienen nuestra clave privada.
- **vi /etc/ssh/sshd_config** en **passwordAuthentication = no,** lo descomentamos
- Reiniciamos el servicio con **systemctl restar ssh.service**
- Comprobamos el estado del servicio con staus.
- Probamos desde el cliente con **ssh -p 22022 usuario@192,168,56,105**

Sesión 2. (SSH CentOS)

Comandos de la practica:

- **Sed**
  - **-i** modifica una linea
  - **-e** sustituye una cadena por otra
- **Journalctl**
  - **-x** muestra explicación de los errores producidos
  - **-f** follow, hace un seguimiento de los errores en el tiempo
- **yum provides**
- **Semanage**
  - -l muestra una lista
  - -a añadir puerto nuevo
  - -p puerto
  - -t tipo de puerto
- **Ufw status/enable/allow**
- **Firewall-cmd**
  - --add-port=
  - --permanente
  - –reload
- **Rsync <parametros> <ruta1> <ruta2>**
  - -a guarda metadatos, permisos, privilegios...
  - -v vervose
  - -i informe final
  - -z comprime
  - --delete
- **git init** (inicia el repositorio en la carpeta en la que estamos)
- **git log** (visualiza el log)
- **git status** (muestra el estado de git)
- **git config** (configura el usuario de git)
- **git add**
- **git commit** (-m messange, --ammend modifica mensaje, -ammend -a add un archivo new
- **git diff (**muestra las diferencias entre el work directory y el staged area, con –staged, con HEAD dif entre dir work y repositorio)
- **git revert** <commit> en vez de perder el trabajo como con git reset, desaplicará al directorio de trabajo los cambios hechos en el commit especificado haciendo un nuevo commit
- **git branch** muestra la rama
- **git checkout** cambia de rama, con -b creamos rama y nos cambiamos
- **git stash** salva el directorio sin hacer commit, git stash apply aplica el ultimo stash y con list los mostramos

- **git merge** une una rama
- **git pull**, Nos traemos (git fetch) los cambios del repositorio y unimos (git merge) con los del directorio de trabajo actua

Clase CentOs:

- editamos vi **/etc/sshd/config quitamos el root de login**
- reiniciamos el servicio con **systemclt restart sshd**
- probamos desde el anfitrión si hay conexión con root
- vemos que no hay
- vamos a usar el sed para hacer el cambio de puerto
- man sed
- **sed -e "s/#Port 22/Port 22022/" -i /etc/ssh/ssh_config**
- **cat /*etc*/ssh/sshd_config | grep Port**
- sino funciona volvemos a sustituir
- reiniciamos servicio **systemclt restart sshd**
- como falla llamamos **journalclt -ex**
- el error es que el puerto 22022 que no se puede utilizar
- leemos el archivo de configuración sshd_config y en el ultimo parrafo dice que hay que decirle a Selinux que nos deje el puerto:
- semanage no lo tenemos por lo que **sudo yum install semanage** y no está buscamos con la opcion search pero no sirve porque salen solo librerías tenemos otra opción $ **yum provides semanage**
- antes había que levantar la targeta de red
- vemos que tenemos que hacer **yum install policycoreutils-python -y**
- man semanage
- **semanage port -l ssh**
- buscamos el de ssh con grep
- sustituimos el puerto **semanage port -a -t ssh_port_t -p tcp 22022** así añadimos el puerto
- **semanage port -l | grep ssh**
- **systemctl restart sshd**
- **systemclt status sshd**
- funciona los ssh localhost pero desde fuera no debido a que el FW fireworks está cortando el paso lo configuramos
- firewall-cm sustituye a la complegidad de iptables
- miramos el manual man firewall-cmd
- añadimos el puerto **firewall-cm --add-port=22022/tcp**
- funciona ya desde fuera
- pero al reinicial se va por lo que hay que añadir al comando anterior otra directiva

- **firewall-cm --add-port=22022/tcp --permanent**
- pero hay que hacerlo en orden sino no se inicia, es decir lo ponemos y lo añadimos permanente
    - En Ubuntu:
        - **ufw status**
        - **ufw enable**
        - **ufw allow 22022**

Copias de seguridad:

- ¿Que diferencia una copia de archivos de una copia de seguridad?
    - Los metadatos (como la fecha).
- Con estos comandos podemos crear una copia de seguridad añadiendo la fecha al final del nombre del archivo.
- La mejor forma de seguridad es en anillos, de fuera hacia adentro: CPDs distribuidos, maquina en el mismo edificio, RAID.
- Hay varios tipos de copias de seguridad:
    - **Total**: hace copia de todo el sistema.
    - **Parcial**: hace copia de una parte del sistema
    - **Incremental**: compara las diferencias y las guarda con respecto a la última versión
- Al hacer la copia tenemos problemas con el espacio en disco y el tiempo. Con LVM evitamos poner en modo monousuario porque tiene snapshots (siempre que haya espacio suficiente)

- Comandos de copia:
    - **dd:** bit bit dd if=/... of=/.. o con .img de of que crea una magen de sda
    - **cpio**: copia archivos a y desde
    - **tar**: crea un empaquetado
    - **rsync**: Sincroniza dos una fuente y destino incluso através de ssh,c opia enlaces, dispositivos, propietarios, grupos y permisos, permite excluir archivos transfiere los bloques modificados de un archivo, puede agrupar todos los cambios de todos los archivos en un único archivo, puede borrar archivos.
        - -e “ssh” si es el destino es name@maquina:/ruta
        - uso común : -aviz
        - copia total: añadimos –delete

- Comandos BackUps
    - Rsnapshot

Sesión 3. (Servidor Web)

- **HTML**: HyperText Mark-up Lenguaje
- **Etiqueta**: un editificador
- **Servidores Web**: proceso que está escuchando que recibe peticiones y devuelve html
  - Apache
  - Engine
  - Lightbox
- **Solución al html statico**: javascript, despues la web dinamica, la suma de base de datos y un lenguaje interpretado.

Almacenamiento	Lenguaje
MySQL → MariaDB	PHP
MongoDB	Python
Postgress	Node JS

- Vamos a instalar un servidor **LAMP: Linux Apache MySQL PHP**

Ubuntu:

- **sudo taskserver** e instalamos lamp server.
- Comprobamos que funciona **sudo systemctl apache2, mysq**...

CentOs:

- buscamos con **yum search apache | grep server**
- **yum install httpd**, d de demonio y **http es hypertext transfer protocol**
- **systemctl status httpd**, como ponse disable lo ponemos enable,
- **systemctl enable httpd**, ahora lo iniciamos y vemos como están con status
- ahora instalamos sudi **yum install php**
- y ahora instalamos sudo **yum install mariadb**
- e instalamos mariadb-server **yum install mariadb-server**
- habilitamos y lanzamos
- hacemos **mysql -u root -p** (user y password pero p en minuscula par aliana de ordenes)
- y vemos que no hay contraseña, vamos a ponerselo
- **mysql_secure_instalation** es un script de configuración para la contraseña y no dejamos que root se conecte a localhost
- con el anfitrion por ssh por el 22022 y **pegamos el script php del manual** en /var/www/html
- **modificamos el script poniendo la ip, el user, el pass y el nombre de bd**

- vamos a **$ mysql** y creamos la base de datos,
- **CREATE DATABASE my_db**
- **CREATE TABLE tabla1,**
- Con anfitrion **probamos conexión con curl** pero lo vemos que el problema es el firewall, entonces,
- **firewall-cmd –add-port=80/tcp –permaner**
- **firewall-cmd –reload**
- **cd /etc/httpd/conf** editamos **vi httpd.conf**
- en module dir module añadimos *.php y reiniciamos el servicio
- instalamos php-mysql **yum install php-mysql**
- ejecutamos el script que está en var/www/html con **php myscript.php**
- SELinux está bloqueando para verlo, **getsebool -a | grep httpd**
- **setsebool -P httpd_can_network_connect_db on**
- reiniciamos httpd **systemctl restart httpd**
- todo funciona con curl

Para seguridad en nuestro servidor:

1. fail2ban
2. rmux y scream
3. RootKit Hunter → nkhunter. Es como un antivirus que analiza lo que se ha instalado

- CentOS y el anfitrion
- nos conectamos a 192.168.56..
- instalamos scream **sudo sym install scream** (sirve para que un proceso no se cierre cuando nos deconectemos de la máquina)
- **fail2ban** si un user tiene muchos fallos de login banea al usuario, mete la conexión en un jail.
- **sudo yum install epel-release**
- **sudo yum install fail2ban**
- **systemctl status fail2ban**, hacemos el **systemctl enable fail2ban** y lo iniciamos con start, **systemctl start fail2ban**
- **fail2ban-client** status es el cliente que interactua con el servicio
- configuramos un jail, el archivo está en **cd /etc/fail2ban**
- **cp jail.conf jain.local** y editamos este nuevo en sshd ponemos enable = true
- **systemctl restart fail2ban** y preguntmamos como ha quedado con **fail2ban-client status** y vemos que hay una problema con añadir sshd para más info.
- editamos el archivo de conf otra vez y en sshd sustituimos **port** = por **port 22022**
- ¿como desbaneamos? **fail2ban-client set sshd unbannip 192...**