Práctica 2, Sesión 1 – SSHD

- ssh y telnet

Son protocolos a nivel de aplicación. Con telnet(texto plano) se tiene una comunicación abierta y sin cifrado, por lo que todo puede ser observado por terceras personas. SSH fue creado para que no sucediera lo anterior. SSH lo cifra todo, de manera que nadie externo lo puede ver. Ssh funciona con cliente-servidor.

/etc/services → archivo que contiene alias locales para los puertos más usados.
Telnet → puerto 23
SSH → puerto 22

- Cambiar puerto (ubuntu y centOs)

- UBUNTU

Primero instalamos los servicios de ssh en ubuntu, para ello es mejor estar en modo superusuario.

1. Buscamos el paquete con “apt-cache search sshd”
2. Instalamos el paquete con “apt-get install openssh-server”

Ahora comprobamos que el servicio está activo, para ello usamos “systemctl status sshd”. Este servicio normalmente se activa cuando se inicia ubuntu. En caso de que no estuviera activo el servicio ssh usaríamos “systemctl enable ssh” y lo arrancaríamos con “systemctl start ssh”.

Para cambiar el puerto, sólo tenemos que cambiar el archivo ssh_config que se encuentra en “cd /etc/ssh/”. Sólo tendríamos que cambiar el puerto al deseado (por defecto es 22) con “nano -w ssh_config”.

Ahora sólo tendríamos que comprobar la conexión desde nuestro terminal a la máquina virtual con ssh, para ello usamos “ssh elena@192.168.56.105”, donde elena es el nombre del usuario en la máquina virtual y la otra parte es la IP (ifconfig). Para saber si lo hemos hecho bien, creamos un archivo en blanco con “touch archivo” y se mira en la otra terminal si se ha creado.

Salimos con “exit” y ahora queremos hacer que el root nos deje el acceso. Para ellos sólo tenemos que modificar el archivo “/etc/ssh/sshd_config” y ponemos “PermitRootLogin yes”, guardamos y restauramos el servicio con “systemctl restart sshd”.

Ahora desde la terminal del ordenador probamos a acceder al root de ubuntu con “ssh -p 22 root@192.168.56.105”.

Fail2band es una herramienta que nos permite controlar el número de accesos a un máquina además de impedirle el acceso después de una serie de intentos junto a un tiempo de baneo. Esto se configura en el archivo “/etc/fail2band/jail.conf” con las variables *bantime* y *maxretry*.

- CENTOS

Primero instalamos ssh en CentOs y luego hacemos lo mismo que hicimos para ubuntu
1. yum search sshd
2. yum install openssh-server

Comprobamos que está arrancado “systemctl status sshd” y seguimos con los pasos de ubuntu.

- Criptografía Llave Automática

Antes se usaba un cifrado simétrico en el que ambas partes compartían llave/código para cifrar y descifrar. La ventaja es que una persona en medio no puede leer y el destinatario confía en la identidad del emisor. Sin embargo, el problema es, que si hay muchas personas que tienen la llave, se pierde la identidad del emisor.

Para solucionar este problema aparece la llave automática, donde se cifra el contenido y se identifica al emisor, garantizando la integridad del mensaje. Ahora se usan llaves a pares, pero tiene el problema de que puede llegar a haber muchas llaves.

En la criptografía de llaves asimétrica hay una llave de cifrado y descifrado que son distintas. Tecnológicamente son iguales (cuánto más grande es la llave, más seguridad). Una de ellas es pública y la otra privada (sólo la conoce el propietario).

La llave pública no sirva para cifrar, sirve para identificar al emisor. Privada → Pública (Ineficiente) y sólo funciona en esa dirección. Para garantizar la integridad se usa la firma digital.

El Digest tiene que ser impredecible. A partir de él, es imposible recuperar la información inicial. Un único cambio, por insignificante que sea, produce un nuevo digest. Éste se cifra con tu llave privada para que no se produzcan modificaciones externas.

FIRMA DIGITAL → DIGEST + LLAVE PÚBLICA

Si nos conectamos de una máquina virtual a otra, podemos ver la llave pública. Si queremos verificar que es correcta, la buscamos en internet.

El certificado es un documento firmado por el notario en el que está la llave pública

-Conexión sin contraseña (ssh, con certificados)

Hacemos “ssh-keygen” (Si no queremos poner contraseña no hace falta, pero es conveniente) y vemos como se ha generado la llave pública. Esta llave se encuentra en “/root/ssh/id_rsa.pub o id_rsa”.

Ahora hacemos “ssh-copy-id elena@192.168.56.105” si estoy en centOs y “ssh-copy-id elena@192.168.56.25” si estoy en ubuntu. Y ya estaría sin contraseñas, sólo tenemos que poner la contraseña de la llave creada anteriormente.

Práctica 2, Sesión 2 – Copias, Backups y Control de Versiones

En esta sesión vamos a ver “dd” que funciona a nivel de filesystem y de bloques, además veremos “cpio”, “tar” y “rsync” que funcionan sólo a nivel de filesystem.

- DD (copia binaria)

Tiene dos parámetros obligatorios y se suele usar para acceso por bloques.
Con “dd if=/etc/fstab of=./miFstab” copiamos el archivo fstab en otro.

Si queremos copiar la partición sda1 hacemos:
1. cp -r /boot/ .miBoot (sólo copia ficheros no bloques)
2. sudo dd if=/dev/sda1 of=./miBoot.img bs=1k(1M) (imagen de toda la partición de arranque)

“dd if=/dev/zero of=./zeroNumbers.dat bs=1k count=56” crea un archivo de 0’s.

- CPIO

Para hacer una copia de seguridad de los archivos conf almacenados en /etc hacemos:
1. cd /etc
2. find . -iname ‘*.config’ | cpio -ov > /home/elena/etcConf.cpio
3. ls * | cpio -ov > /home/elena/otro.cpio
4. nano -w pam.conf
5. cpio -ivu “pam.conf” < /home/elena/etcConf.cpio

- TAR

Comprimir → tar cfz /home/elena/etc.tgz /etc
Descomprimir → tar xfz /home/elena/etc.tgz

- RSYNC

Este comando sirve para sincronizar dos directorios. Normalmente se usa para sincronizar ordenadores remotos. Permite la copia de enlaces, la gestión de dispositivos, usuarios y permisos.

Creamos en UBUNTU (creo) un directorio ISE en el que hay una carpeta A2, compuesto por dos carpetas P1, P2 dentro de los cuales se encuentran un archivo prueba1.dat, prueba2.dat, respectivamente. Y creamos dos archivos calificaciones.txt, datos.txt donde queramos.

Nos conectamos a CentOs desde Ubuntu (acceso sin contraseña) y nos salimos.
Definimos RSYNC_RSH=ssh en el archivo “nano -w profile” poniéndolo al final de éste. Y hacemos “source profile”.

1. echo $RSYNC_RSH → ls → rm -rf etc → rm etc.tgz
2. rsync -a --delete ise (máquina centos)elena@192.168.56.25:/home/elena

Comprobamos que todo se ha copiado bien.
Cambiamos un archivo cualquiera y creamos una carpeta nueva. Al hacer esto, para ver los cambios con el ssh, tenemos que hacer de nuevo rsync, ya que no se hace automáticamente.

- Copia de versiones (Git)

Ventajas → Registro de cambio (Cuando cambias líneas o información, guarda las anteriores versiones con el commit)

<Directorio Proyecto>
→ ise2018A2, todos los archivos que cuelgan, están controlados por el CV y cada vez que se haga cualquier cambio se va a crea un snaphoost de todo el directorio.

Ventajas → Auditoría, Ramas de cambio, …

Single Branch → Extreme programme
Feature Branch → Sólo los cambios van a producción

Ejemplos →
1. CVS(Control version system)
   Problema → centralizado y ocupa mucho espacio
2. SVN(Subversion)
   Problema → centralizado pero más rápido
3. GIT
   Distribuido

Centralizado – repositorio central donde todos los programadores suben sus cambios.
Distribuido – no hay repositorio central sino que cada ordenador tiene una copia "completa" de todo el repositorio

- GIT

Estándar:
- Working Space
- Stage
- Repositorio

- Remoto

Pasos realizados por el profesor de prácticas para la realización de la práctica a realizar por los alumnos:

1. Se crea el siguiente directorio "git init ise2018A2" y vemos con "ls -a" que hay un archivo llamado .gitignore el cual vamos a modificar para que no se tenga en cuenta esos archivos a la hora de subir nuestras prácticas a github.

2. Entramos en la carpeta "cd ise2018A2" y modificamos el archivo antes mencionado "nano -w .gitignore":

```
#Evito ficheros objeto
*.o *.class *.lib
#Archivos copia de seguridad
~*   *.bak
#Contraseñas
.htaccess  psswd
```

3. Creamos el siguiente archivo "nano README.md" para hacer una descripción de lo que va a tener nuestro repositorio

4. Ahora añadimos todos los archivos “git add *” para subirlos a la nube y antes de escribir lo que hemos hecho tenemos que configurar el correo y el usuario que tenemos en github con “git config --global user.email” y “git config --global user.name”

5. Pasamos a comentar lo hecho hasta ahora con “git commit” y vemos la historia de cambios realizados con “git log”.

6. Si queremos ver un archivo en formato diff usamos “git show archivo” y hacemos “git commit -a” y cambiamos el commit.

7. Creamos el repositorio desde github, hacemos un remote y por último push, pero con ssh.

Trabajo a realizar por el alumno: Crear una rama distinta a la master y subir los cambios. Para ello tenemos que enviar el usuario de github al profesor de prácticas por swad.

Ponemos nuestro ssh en github usando “ssh-keygen” en la terminal. Cogemos la llave pública y la copiamos en nuestro perfil. Y ahora podemos clonar por ssh. Creamos un branch “git checkout -n ElenaCanteroMolina”. Si queremos ver en la rama que nos encontramos usamos “git branch” y subimos la rama creada con “git push origin ElenaCanteroMolina”. Nos cambiamos a la rama master “git checkout master” y hacemos “git merge ElenaCanteroMolina”.

Para añadir nuestra fila al archivo de github hacemos:
1. git checkout -b <nombre rama>
2. editamos en fichero
3. git commit -a
4. git push origin <nombre rama>

Para resumir el funcionamiento de git, lo que hace es:
Tenemos nuestro directorio de trabajo y queremos realizar un seguimiento de cambios, para ello creamos el repositorio con el comando git init. Para comprobar el estado de los archivos ejecutamos git status.

Una vez creado tendríamos que añadir los cambios y los ficheros para que realice un seguimiento de estos con git add archivo1. En el caso de que ya se haya añadido el seguimiento y solo haya sido modificado se expondrá en el siguiente punto

Ahora se realizaría un commit que en resumen es ponerle una etiqueta al trabajo realizado.
Esto sería con git commit -m "Breve resumen del cambio". Si se le añade -a se aplicará a los archivos seguidos que hayan sido modificados.

Nota: Al realizar la primera vez un commit hay que modificar el archivo ~/.gitconfigañadiendo: [user] name = Nombre email = correo@ausar.com

Práctica 2, Sesión 3 – LAMP

Esta práctica se puede realizar en nuestro propio ordenador. Si lo hacemos desde ubuntu va a ser automático, pero para hacerlo en CentOs es manual.

- Ubuntu

Vamos a usar tasksel que es un programa que sirve para instalar paquetes. Cuando lo ejecutamos, marcamos LAMP server y ya está. Nos ponemos en modo superusuario y empezamos a instalar lo necesario.

Primero miramos si tenemos instalados los paquetes con “apt list --installed | grep apache”, “ apt list --installed | grep mysql” y “apt list --installed | grep php”

Usamos “systemctl list-unit-files” que es más limpio que “systemctl list-units”

-**Apache2**

Miramos si el siguiente servicio está activo “systemctl status apache2” y podemos ver que tiene 6 hijos corriendo. Para ver si apache está de verdad funcionando, usamos un navegador y ponemos la dirección ip de nuestra máquina.

El directorio por defecto donde está la página web es “/var/www/html”, si no especificamos ningún archivo .html coge por defecto el index.html

/etc/apache2/apache2.conf

Para configurar cómo funciona php dentro de apache podemos usar el directorio “/etc/php/7.0”
Encontramos dos ficheros, el primero tiene la configuración estándar y el segundo es un command line interface (cli).

Para cambiar el comportamiento → apache2
Si modificamos algo de la configuración debemos volver a lanzar el servidor de apache2.

“nano -w hola.php” (/var/www/html)

```
<html>
    <head>
        <title>PruebadePHP</title>
    </head>
    <body>
        <?phphpinfo()?>
    </body>
</html>
```

Formato de un archivo en php y ponemos que nos muestre la información de php con “php_info()”

“cd etc/php/7.0/apache2” y modificamos “short_open_tag = on”. Esto sirve para que se sepa que es php siempre el archivo a no ser que pongamos otra cosa y hacemos “systemctl restart apache2” También podemos modificar “upload_max_filesize = 20M”.

Por defecto, los errores van a “/var/log/apache2/err.log” y en “/var/log/apache2/access.log” se registran todos los accesos. Por ejemplo se puede usar para hacer estadísticas de acceso, …

-**MySql**
Primero miramos el estado del servcio “systemctl status mysql”
mysql

-**Firewall (ufw)**
ufw status → systemctl status ufw
system enable ufw / ufw enable
ufw status verbose
ufw disable
ufw enable
ufw allow 80 → acceso al puerto 80
ufw allow apache
ufw allow https
ufw allow ‘Apache Full’
ufw status verbose
ufw deny 80

- CentOs

En CentOs no tenemos paquete para instalar LAMP server directamente como en Ubuntu, por lo tanto tenemos que instalarlo uno por uno.

-**Https**
Buscamos el paquete a instalar “yum search httpd” y lo instalamos “yum install httpd”. Este servicio no se activa solo, así que tenemos que hacer que se active automáticamente con “systemctl list-unit-files | grep httpd”, “systemctl enable httpd”, “systemctl start httpd” y “systemctl status httpd”. Modificamos “nano -w /etc/httpd/conf/httpd.conf”.

Comprobamos que funciona igual que lo hicimos en ubuntu. Y vemos que no podemos entrar y hacemos “ps aux | grep httpd”

curl → llama a servidores web ///// wget()

Ahora usamos **lynx** que es un browser en modo texto y por lo tanto tenemos que instalarlo.
Vamos a ver el **firewall** de CentOs. Comprobamos el estado con “firewall-cmd --state”, lo desactivamos en caso de que esté corriendo y comprobamos el browser. Después lo arrancamos de nuevo y habilitamos el puerto 80.

systemctl stop firewalld → firewall-cmd --state
systemctl start firewalld
firewall-cmd --zone=public --add-service=http
firewall-cmd --list-all (Aquí debe salir http, y comprobamos en firefox)
systemctl restart firewalld

Si comprobamos http se pierde, para hacerlo permanente hay dos maneras:
1º - firewall-cmd --zone=public --add-service=http --permanent
Esto no lo añade a, lo pone sólo en la configuración, tendríamos que ponerlo en php.conf o algo así y hacer restart para que lo haga bien.

2º - firewall-cmd --zone=public --remove-port=80/tcp
- firewall-cmd --zone=public --add-port=80/tcp --permanent
- firewall-cmd --reload
- firewall-cmd --runtime-to-permanent

-**MySql**
Buscamos el servicio “yum search mysql” y lo instalamos “yum install mariadb” y también hay que instalar el server de este servicio sino, no funciona.

systemctl list-unit-files | grep -i maria
systemctl status mariadb
systemctl enable mariadb
systemctl start mariadb
systemctl status mariadb

mysql → quit

mysql_secure_installation (borramos el usuario anónimo y deshabilitamos el acceso remoto)
mysql -uroot -p test

-**PHP**
Buscamos el servicio a instalar “yum search php” y lo instalamos “yum install php”

cd /etc/httpd/conf.d/
/var/www/html → crear index.php (Igual que en ubuntu)

systemctl restart httpd(php/apache)

php no admite mysql, por lo que tenemos que instalar un driver “yum search mysql” y “yum install php-mysql”

systemctl restart httpd

mysql connect → necesitamos un ejemplo y lo ponemos en index.php

mysql grant new user
mysql -uroot -p
    grant all privileges on test.* to ‘david@localhost’ identified by ‘ise2018’;
    quit

mysql -udavid -p<contraseña> test