Índice de contenido

Información de derechos reservados de esta publicación.

Autor del manual: Rodrigo Mendoza Martinez

TEMA 8. SERVICIO OPENSSH

8.0 El Protocolo SSH

El protocolo SSH (Secure Shell) es una herramienta que nos permite conectarnos a equipos remotos (Servidores en Producción) así mismo, nos da la capacidad de llevar a cabo tareas administrativas dentro del mismo como, activar o apagar servicios,

Además de la conexión a otros equipos, SSH nos permite copiar datos de forma segura, gestionar claves RSA para no escribir claves al conectar a las máquinas y pasar los datos de cualquier otra aplicación por un canal seguro tunelizado mediante SSH.

Una clave RSA (Sistema Criptográfico con Clave Publica) es un algoritmo que genera un par de llaves de autenticación, la publica y la privada. La publica se distribuye en forma autenticada y la privada que generalmente es guardada en secreto por el propietario.

El protocolo SSH (Secure Shell) esta implementado bajo el estándar TCP/IP, el cual a su vez se encuentra dividido en 5 secciones:

1. Nivel Físico
2. Nivel De Enlace
3. Nivel de Internet
4. Nivel de sesión
5. Nivel de Transporte
6. Nivel de Aplicación

Por lo que el protocolo SSH esta ubicado en la quinta capa del modelo TCP/IP, nos referimos a la capa de aplicación

La capa de aplicación es el nivel que los programas más comunes utilizan para comunicarse a través de una red con otros programas. Los procesos que acontecen en este nivel son aplicaciones específicas que pasan los datos al nivel de aplicación en el formato que internamente use el programa y es codificado de acuerdo con un protocolo estándar.

De manera predeterminada, el protocolo SSH atiende peticiones por el puerto 22

En este capitulo haremos uso de OpenSSH la cual es la alternativa libre y abierta al programa propietario SSH

8.1 Acerca de OpenSSH

OpenSSH (Open Secure Shell) es un conjunto de aplicaciones que permiten realizar comunicaciones cifradas a través de una red, usando como base al protocolo SSH. Este proyecto es liderado actualmente por Theo de Raadt quien actualmente es fundador y líder de proyectos como OpenBSD.

Los desarrolladores de OpenSSH aseguran que este es más seguro que el original, lo cual es debido a la conocida reputación de los desarrolladores de OpenBSD por crear código limpio y perfectamente auditado, lo que contribuye a que sea más seguro. Su seguridad también es atribuible al hecho de que su código fuente se distribuya libremente con una licencia BSD. Aunque todo el código fuente del SSH original también está disponible, existen restricciones con respecto a su uso y distribución, lo que convierte a OpenSSH en un proyecto mucho más atractivo a la hora de atraer nuevos desarrolladores.

Además de la conexión a otros equipos, openSSH nos permite copiar datos de forma segura mediante la implementación de dos herramientas proporcionadas por openSSH, estas son:

- SCP
- SFTP
- SSHFS

8.2 Instalación de OpenSSH

A partir de este punto empezaremos a descargar los paquetes necesarios para el perfecto funcionamiento de openSSH, de esta manera si usted se encuentra trabajando bajo algún ambiente gráfico, sea KDE o GNOME le pedimos abra una terminal de BASH, por otra parte si usted se encuentra trabajando bajo linea de comandos no tendrá que hacer nada.

Los paquetes a descargar son los siguientes:

- ssh

La forma en que se instalaran estos paquetes sera tecleando en consola lo siguiente:

```
root@server1:~$apt-get install ssh
```

Al instalar el servicio ssh, el también instala el servidor y cliente.

8.3 Configuración de OpenSSH

OpenSSH dispone de dos conjuntos diferentes de ficheros de configuración: uno completamente dedicado al cliente (ssh, scp y sftp) y otro orientado completamente al servidor.

8.3.1 Archivos de configuración del servidor

La ubicación de los ficheros referentes al servidor se encuentran en la ruta:

```
/etc/ssh/
```

Dentro del directorio podemos encontrar los siguientes ficheros de configuración:

moduli	Contiene grupos Diffie-Hellman usados para el intercambio de la clave Diffie-Hellman que es imprescindible para la construcción de una capa de transporte seguro. Cuando se intercambian las claves al inicio de una sesión SSH, se crea un valor secreto y compartido que no puede ser determinado por ninguna de las partes individualmente. Este valor se usa para proporcionar la autenticación del host.
ssh_config	El archivo de configuración del sistema cliente SSH por defecto. Este archivo se sobrescribe si hay alguno ya presente en el directorio principal del usuario
sshd_config	El archivo de configuración para el demonio sshd

ssh_host_dsa_key	La clave privada DSA usada por el demonio sshd
ssh_host_dsa_key.pub	La clave pública DSA usada por el demonio sshd
ssh_host_rsa_key	La clave privada RSA usada por el demonio sshd para la versión 2 del protocolo SSH.
ssh_host_rsa_key.pub	La clave pública RSA usada por el demonio sshd para la versión 2 del protocolo SSH.

8.3.2 Archivos de configuración del cliente

La ubicación de los ficheros referentes al cliente se encuentran almacenados en el directorio de trabajo de cada usuario:

Ejemplo: "/home/curso/.ssh/"

Dentro del directorio podemos encontrar los siguientes ficheros de configuración:

identity.pub	La clave pública RSA usada por ssh para la versión 1 del protocolo SSH.
known_hosts	Este archivo contiene las claves de host DSA de los servidores SSH a los cuales el usuario ha accedido. Este archivo es muy importante para asegurar que el cliente SSH está conectado al servidor SSH correcto.

8.4 Configuración de fichero sshd_config

La función que desempeñan los ficheros de configuración de openSSH son de vital importancia para la seguridad de nuestro servidor , ya que si no se llegaran a configurar apropiadamente estos ficheros la vulnerabilidad de nuestro servidor seria demasiado sensible a ataques informáticos, es por ello que le enseñaremos la manera apropiada en la que deberá ser configurado este vital fichero.

8.4.1 Blindando el fichero sshd_config

Este fichero lo podrá localizar en en la siguiente ruta

```
/etc/ssh/
```

El siguiente paso sera abrir el fichero con la ayuda del editor de textos VI

```
[root@server1:~$ vim /etc/ssh/sshd_config
```

A partir de este punto comenzaremos a blindar SSH.

8.4.2 Cambiando el puerto

SSH tiene asignado por defecto el puerto 22, esto es algo que conocen todos nuestros posibles atacantes , por lo que es una buena idea cambiarlo.

Para modificar esta opción y las siguientes que iremos mencionando editaremos el fichero de configuración **sshd_config**, que por defecto se encuentra en el directorio **/etc/ssh/**.

Se recomienda usar un puerto cualquiera por encima del 1024, así que usted puede elegir el que quiera. En este ejemplo usaremos el 59345, por lo que tendrá que editar el parámetro ***Port*** del fichero de configuración el cual deberá quedar así:

```
Port 59345
```

8.4.3 Protocolos ssh.

Hay dos versiones de ssh en cuanto a su protocolo de comunicación, estas son:

- Versión 1
- Versión 2.

La versión 1 de openSSH hace uso de varios algoritmos de cifrado de datos mas sin embargo, algunos de estos algoritmos han dejado de ser mantenidos por sus creadores y por lo tanto presenta serios huecos de seguridad que potencialmente permite a un intruso insertar datos en el canal de comunicación. Para evitar el uso del protocolo 1 y sus posibles ataques a este, basta con indicar que solo admita comunicaciones de ssh basadas en el protocolo 2.

Por default en debian/ubuntu ya viene configurado en el versión 2 del protocolo.

8.4.4 Parámetro PermitRootLogin.

Este es quizá el parámetro mas importante de seguridad que podemos indicar para blindar nuestro servidor. Prácticamente la mayoría de sistemas operativos Linux crean por defecto al usuario root , es por ello que la mayoría de los ataques informáticos se concentran en atacar al equipo a través de la cuenta de root y mucho mas si la cuenta tiene asignada una contraseña débil, una manera de deshabilitar el logeo al sistema a través de la cuenta de root es poner en '**no**' la variable **PermitRootLogin** , con esto el usuario root no tendrá permiso de acceder mediante ssh y por lo tanto cualquier intento de ataque directo a root será inútil. Con esto siempre tendremos que ingresar como un usuario normal y ya estando adentro entonces mediante un **su –** cambiarnos a la cuenta de root.

Para llevar a cabo estos cambios tendrá que editar el parámetro **PermitRootLogin** del fichero de configuración el cual deberá quedar de la siguiente manera:

```
PermitRootLogin no
```

8.4.5 Parámetro X11Forwarding.

Si nuestro servidor no tienen entorno gráfico instalado, o no queremos que los usuarios se conecten a él, definiremos esta opción en el fichero de configuración:

Para llevar a cabo estos cambios tendrá que editar el parámetro **X11Forwarding** del fichero de configuración el cual deberá quedar de la siguiente manera:

```
X11Forwarding no
```

8.4.6 Parámetro LoginGraceTime.

El número indica la cantidad de segundos en que la pantalla de login estará disponible para que el usuario capture su nombre de usuario y contraseña, si no lo hace, el login se cerrará, evitando así dejar por tiempo indeterminado pantallas de login sin que nadie las use, o peor aun, que alguien este intentando mediante un script varias veces el adivinar un usuario y contraseña. Si somos el único usuario del sistema considero que con 20 o 30 segundos es mas que suficiente.

Para llevar a cabo estos cambios tendrá que editar el parámetro **LoginGraceTime** del fichero de configuración el cual deberá quedar de la siguiente manera:

```
LoginGraceTime 30
```

8.4.7 Permitiendo AllowUsers.

Este parámetro solo permitirá la conexión a los usuario que este registrados del archivo de configuración del openssh, el parámetro no existe nosotros lo podemos agregar el parámetro al final del archivo.

```
AllowUsers curso
```

Guardamos la nueva configuración y con esto ya tendremos configurado nuestro servicio OpenSSH.

8.4.8 Iniciando el servicio SSH

Llegado a este punto usted ya deberá contar con las configuraciones de seguridad apropiadas, por lo que solo faltaría iniciar el servicio de SSH.

Para iniciar el servicio de SSH tendrá que teclear en consola y como **root** lo siguiente:

```
root@server1:~# /etc/init.d/ssh restart
*Restarting OpenBSD Secure Shell server sshd            [ OK ]
```

Igualmente existen otras opciones:

start	Inicia el servicio
stop	Detiene el servicio
restart	Reinicia el servicio.-La diferencia con **reload** radica en que al ejecutar un **restart** este mata todos los procesos relacionado con el servicio y los vuelve a generar de nueva cuenta
reload	Recarga el servicio.-La diferencia con restart radica en que al ejecutar un reload este solamente carga las actualizaciones hechas al fichero de configuración del servicio sin necesidad de matar los procesos relacionados con el mismo, por lo que podría entenderse que hace el cambio en caliente.

8.5 Herramienta ssh

Para establecer una conexión con un servidor SSH remoto, haremos uso del Bash, o también conocido como Terminal.

La sintaxis para llevar a cabo esta operación es la siguiente:

```
[root@localhost]# ssh usuarioRemoto@ipDelServidorRemoto
```

En caso de haber establecido un puerto de escucha distinto al puerto 22, solo deberá especificar el puerto por el cual requiere autenticarse al servidor. Ejemplo

```
[root@localhost ]# ssh -p[puerto] usuarioRemoto@ipDelServidorRemoto
```

Ejemplo 1: La empresa Factor Integración para la cual trabajamos, nos ha pedido reiniciar el servicio de apache , para ello nos ha proporcionado los siguientes datos:

- IP del servidor remoto -> **192.168.1.243**
- Nombre del usuario remoto -> **curso**
- Puerto de autenticación -> **59345**

Para conectarnos al servidor remoto habrá que especificar el puerto de escucha, el usuario remoto y la IP del servidor remoto.

```
rodmen@laptop:~$ ssh -p 59345 curso@192.168.1.243
curso@192.168.1.243's password:
```

Una vez dentro del servidor remoto nos logearemos ahora si como **“root”**

```
curso@server1:~$ su -
Password:
```

Por ultimo, solo bastara reiniciar el servidor de apache

```
root@server1:~# /etc/init.d/apache2 restart
```

Para salir del SSH solo basta teclear “exit”

```
root@server1:~# exit
logout
curso@server1:~$ exit
logout
Connection to 192.168.1.243 closed.
lucifer@lucifer:~$
```

Pero también tiene otra función podemos ejecutar comandos remotamente sin estar conectados al servidor:

```
rodmen@laptop:~$ ssh -p 59345 curso@192.168.1.243 "df -h"
curso@192.168.1.243's password:
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda2             9.3G  413M  8.4G   5% /
varrun                756M   48K  756M   1% /var/run
varlock               756M     0  756M   0% /var/lock
udev                  756M   60K  756M   1% /dev
devshm                756M     0  756M   0% /dev/shm
/dev/sda1             950M   35M  867M   4% /boot
/dev/sda6              19G  173M   18G   1% /home
/dev/sda3             950M   18M  885M   2% /tmp
/dev/sda5              14G  264M   13G   2% /var
lucifer@lucifer:~$
```

Como pueden ver la información desplegada es del servidor que se encuentra remotamente.

8.6 Herramienta scp

Es un medio de transferencia segura de archivos entre un equipo local y uno remoto haciendo uso del protocolo Open Secure Shell (openSSH).

La sintaxis de SCP para llevar a cabo esta operación es la siguiente:

```
rodmen@laptop:~$scp usuario@IP:rutaDelRecursoRemoto
```

En caso de haber establecido un puerto de escucha distinto al puerto 22, solo deberá especificar el puerto por el cual requiere autenticarse al servidor. Ejemplo

```
rodmen@laptop:~$scp -P[puerto] usuario@IP:rutaDelRecursoRemoto
```

El scp tiene varias opciones para su uso:

OPCION.	DESCRIPCION.
-P	Este parámetro se ocupa cuando el servicio OpenSSh esta escuchando por otro puerto.
-r	Este parámetro permite la copia recursiva de directorios.
-p	Conservar los permisos y datos de la creación, modificación de los archivos y carpetas.
-C	Permite la compresión al vuelo de la información cuando pasa de un equipo a otro.
-q	Desactiva el historial de envía/recepción de información.
-l	Limitamos el ancho de banda para el envío/recepción de información.

Vamos a ver como funciona muestra herramienta, ejemplos:

Mandaremos un solo archivo a través del puerto 59345.

```
rodmen@laptop:~$ scp -P 59345 wndw3-es-ebook.pdf curso@192.168.1.243:/home/curso

curso@192.168.1.243's password:

wndw3-es-ebook.pdf                                    100% 6607KB   6.5MB/s   00:01
rodmen@laptop:~$
```

Ahora mandaremos una carpeta, subcarpetas y archivos dentro de las mismas sin perder ningún permiso y comprimido al vuelo.

```
rodmen@laptop:Documentos$ scp -P 59345 -rpC Bash/ curso@192.168.1.243:/home/curso

curso@192.168.1.243's password:

bash-intro.pdf                                        100%  139KB 138.7KB/s   00:00
guia_reglas_bash.pdf                                  100%   39KB  38.7KB/s   00:00
taller_sh.pdf                                         100%   64KB  63.7KB/s   00:00
array.txt                                             100%  103     0.1KB/s   00:00
bash-script-2.0.pdf                                   100%  455KB 455.3KB/s   00:00
ejemplos-Bash3.doc                                    100%   66KB  65.5KB/s   00:00
Bash-Prog-Intro-COMO.pdf                              100%   93KB  92.5KB/s   00:00
macprogramadores.org.BASH.pdf                         100% 1010KB   1.0MB/s   00:00
ejemplos-Bash2.doc                                    100%  179KB 178.5KB/s   00:00
recursos1_ison.pdf                                    100%  314KB 314.2KB/s   00:00
ejemplos-Bash1.doc                                    100%  184KB 183.5KB/s   00:00
capitulo_2_shell.pdf                                  100%  116KB 115.8KB/s   00:00

rodmen@laptop:Documentos$
```

También podemos descargar información desde el servidor solicitando desde el cliente a esto se le llama scp a la inversa. La única controversia es que debemos conocer en donde se localiza la información.

```
rodmen@laptop:~$scp -P 59345 curso@192.168.1.243:/home/curso/pmagic-3.5.iso.zip .
curso@192.168.1.243's password:
pmagic-3.5.iso.zip                                    100%   73MB  10.5MB/s   00:07
rodmen@laptop:~$
```

Como vemos el ejemplo estamos descargando un fichero del servidor remoto, pero hay un punto al final de la linea, esto significa que en donde se encuentra ubicados en su sistema sera donde guarde el archivo.

8.7 Herramienta sftp

El protocolo de transferencia de archivos SFTP es un protocolo que proporciona la transferencia de archivos y la funcionalidad de manipulación de los mismos Se utiliza normalmente con SSH a fin de asegurar la transferencia de archivos.

En comparación de capacidades con el anterior protocolo SCP, que únicamente permite la transferencia de archivos, el protocolo SFTP permite una serie de operaciones sobre archivos, ficheros, o carpetas remotos, en pocas palabras, nos permite navegar directamente en el servidor remoto con el fin de localizar el recurso que deseamos descargar.

La sintaxis de SFTP para llevar a cabo esta operación es la siguiente:

```
rodmen@laptop:~$ sftp usuarioRemoto@ipDelServidorRemoto
```

En caso de haber establecido un puerto de escucha distinto al puerto 22, solo deberá especificar el puerto por el cual requiere autenticarse al servidor. Ejemplo.

```
rodmen@laptop:~$ sftp -o Port=[Puerto] usuario@IP
```

El siguiente paso sera autenticarnos con la contraseña del usuario remoto

```
rodmen@laptop:~$ sftp -o Port=59345 curso@192.168.1.243
Connecting to 192.168.1.243...
curso@192.168.1.243's password:
sftp>
```

Se manejan los mismo comandos del servicio ftp.

8.8 Herramienta sshfs

Nosotros podemos acceder a un sistema de ficheros remoto usando **sshfs** el cual nos permite montar un sistema de ficheros remotos cifrados mediante la implementación del protocolo OpenSSH.

De esta manera nosotros podemos acceder a los archivos remotos como si estuvieran dentro de nuestra maquina, solo debemos recordar que la conexión entre las computadoras sera un tanto lenta.

sshfs (Secure Shell File System) es un sistema de ficheros de Linux que tiene como funcionalidad montar sistemas de ficheros remotos en nuestro equipo.

Los efectos prácticos de esto es que el usuario final puede interactuar amigablemente con archivos remotos estando en un servidor SSH, viéndolos como si estuvieran en su computadora local.

8.8.1 Instalando sshfs y fuse

Los paquetes necesarios para la instalación serán:

- **sshfs.**
- **fuse-utils.**

Teclee la siguiente instrucción desde una terminal BASH para instalar los paquetes antes descritos:

```
laptop:~# apt-get install sshfs fuse-utils
```

8.8.2 Configuración sshfs y fuse

Una vez instalado, establece los permisos adecuados para que lo puedas montar como usuario mortal.

```
laptop:~# chmod 4755 /usr/bin/fusermount
```

El siguiente paso sera crear un punto de montaje para el sistema de ficheros remoto, esto lo haremos tecleando el siguiente comando:

```
laptop:~# mkdir  /mnt/sshfs
```

Cargamos el modulo fuse para que permita sshfs funcione.

```
laptop:~# echo "fuse" >> /etc/modules && modprobe fuse
```

8.8.3 Montar unidad remota sshfs

Por ultimo solo deberá montar el sistema de ficheros remoto, para llevar a cabo esta operación deberá ejecutarlo como root del sistema.

```
laptop:~# sshfs curso@192.168.1.243:/home/curso /mnt/sshfs
```

Si se tiene especificado algún puerto de escucha para el servidor de SSH solo deberá especificare. mediante el parámetro “-p”

```
laptop:~# sshfs -p 59345 curso@192.168.1.243:/home/curso /mnt/sshfs
curso@192.168.1.243's password:
laptop:~#
```

8.8.4 Montar unidad vía fstab.

Montar particion desde el fichero fstab.

```
laptop:~# vim /etc/fstab
```

Agregamos la siguiente al final del archivo la siguiente linea.

```
sshfs#curso@192.168.1.243:/home/curso /mnt/sshfs fuse port=59345,user,noauto 0 0
```

Ya podremos montarlo de una forma mas simple,

```
laptop:~# mount /mnt/sshfs
curso@192.168.1.243's password:
laptop:~#
```

Para desmontarlo solamente debemos hacer los siguiente.

```
laptop:~# umount /mnt/sshfs
```

8.8.5 Montar unidad como usuario normal.

Como hemos visto hasta el momento solamente el usuario administrador o root, solo puede montar las unidades remotas sshfs, pero no siempre el administrador del sistema va estar siempre montando la unidades remotas cada dia, por eso le vamos a dar permisos a los usuarios mortales para poder montar esta unidades.

Agregaremos al usuario rodmen al grupo de fuse, para poder tener permisos a las unidades.

```
laptop:~# usermod -G fuse -a rodmen
```

Ahora debemos cambiar los propietario y grupo de la carpeta en donde se van a montar la unidad remota.

```
laptop:~# chown rodmen.fuse /mnt/sshfs
```

Si estamos conectados dentro de la sesión del usuario que estamos habilitando con permisos al grupo fuse, tendremos que salir de sesión y entrar nuevamente. Casos muy extremos tenemos que reiniciar nuestro SO pero no es muy común.

Ya dentro de la sesión solamente tenemos que abrir una terminal y ejecutar lo siguiente:

```
rodmen@laptop:~$ mount /mnt/sshfs
curso@192.168.1.243's password:
rodmen@laptop:~#
```

Para poder verificar que esta montada ejecutamos:

```
rodmen@laptop:~$ mount | grep fuse
fusectl on /sys/fs/fuse/connections type fusectl (rw)
curso@192.168.1.243:/home/curso on /mnt/sshfs type fuse.sshfs
(rw,noexec,nosuid,nodev,max_read=65536,user=lucifer)
rodmen@laptop:~$
```

Como vemos ya tenemos montada nuestra unidad remota como usuario mortal. Pero ahora como desmontamos la unidad.

```
rodmen@laptop:~$fusermount -u /mnt/sshfs
rodmen@laptop:~#
```

Listo la unidad ya fue desmontada.

8.9 Bitácoras Openssh.

En openssh tiene otras opciones interesantes que se pueden ocupar para el.

- Este comando nos permite ejecutar un test a la configuración del servicio de openssh.

```
root@server1:~# /usr/sbin/sshd -d -d -d
```

- La opción “-vvv” a nuestro comando básicos de openssh, muestra la información de lo que pasando detrás de la ejecución de los mismos.

```
rodmen@laptop:~# ssh -p59345  -vvv curso@192.168.1.243
```

- Este archivo almacena los log del openssh del servidor.

```
root@server1:~# tail -f /var/log/auth.log
```