REDES:

CONFIGURACIÓN DE SSH

CONFIGURACIÓN DE SSH

ÍNDICE

CONFIGURACIÓN DE SSH

Muy breves conceptos respecto del cifrado asimétrico

Previo a explicar el servicio de SSH debemos comprender los conceptos básicos del cifrado asimétrico que es una de las herramientas fundamentales que le dan a SSH el ser lo que es. No es la intención explicar en detalle las bondades de este tipo de cifrado, sino apuntar específicamente a su uso dentro de este servicio.

SSH utiliza el cifrado asimétrico para proteger el canal de comunicación del cliente al servidor y del servidor al cliente; pero ¿cómo lo hace?.

El cliente crea dos claves, una pública y otra privada con las siguientes condiciones:
1.- Con la pública yo solo puedo cifrar y con la privada solo puedo descifrar.
2.- Cada clave pública se corresponde con solo una privada y viceversa.
3.- A partir de la clave pública no se puede deducir la clave privada.
4.- La clave privada JAMÁS se divulga o comparte; pero la clave pública se entrega a quien sea que me la solicite.

Por lo que en el momento de iniciar una conexión, tanto cliente como servidor intercambian sus claves públicas. Cuando el cliente quiera enviar un mensaje al servidor lo cifra con la clave pública del server, que solo podrá ser descifrado con la privada que solo está en poder del servidor. Cuando el servidor por el contrario quiera enviarle un mensaje al cliente, lo cifrará con la clave pública del cliente quien lo descifrará con su propia clave privada.

De esta manera nadie en el medio podrá ver nuestra conversación con el servidor.

El protocolo SSH

El **protocolo SSH** (**Secure Shell**) es una herramienta que nos permite conectarnos a equipos remotos 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 o DSA (sistema de encriptación por medio de llaves) es un algoritmo que genera un par de llaves de autenticación, la pública y la privada. La pública 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:

- Nivel físico
- Nivel de enlace
- Nivel de internet
- Nivel de transporte
- Nivel 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.

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

Estas herramientas en realidad tienen la misma función de copiado, solo se diferencian en la forma en como son aplicadas, tema del cual hablaremos más adelante.

REQUERIMIENTOS PARA LA 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 línea de comandos no tendrá que hacer nada. Los paquetes a descargar son los siguientes:

- OpenSSH
- OpenSSH-clients
- OpenSSH-server

La forma en que se instalarán estos paquetes será tecleando en consola lo siguiente:

```
# aptitude install openssh openssh-clients openssh-server
```

Una vez finalizado el proceso de instalación pasaremos con las configuraciones propias de OpenSSH, nos referimos a los archivos de configuración.

Archivos de configuración de OpenSSH

OpenSSH dispone de dos conjuntos diferentes de archivos de configuración:

- uno completamente dedicado al cliente (ssh, scp y sftp),
- y otro orientado completamente al servidor.

ARCHIVOS DE CONFIGURACIÓN DEL LADO DEL SERVIDOR

La ubicación de los archivos referentes al servidor se encuentra en la ruta:

```
/etc/ssh/
```

Dentro del directorio podemos encontrar los siguientes archivos 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_key	La clave privada RSA usada por el demonio sshd para la versión 1 del protocolo SSH.
ssh_host_key.pub	La clave pública RSA usada por el demonio sshd para la versión 1 del protocolo SSH.
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.

ARCHIVOS DE CONFIGURACIÓN DEL LADO DEL CLIENTE

La ubicación de los archivos referentes al cliente se encuentran almacenados en el directorio de trabajo de cada usuario, en la carpeta .ssh. Ejemplo:

```
/home/usuario/.ssh
```

Si no existiese, como el usuario debemos ejecutar el siguiente comando que nos creará la carpeta junto con la clave pública y privada que utilizará nuestro usuario por defecto, o cada vez que queramos loguearnos como el usuario:

```
ssh-keygen -b 2048 -t rsa -N ‘’
```

Donde:

- -b: tamaño de la clave.
- -t: tipo de clave.
- -N: passphrase. En este caso creamos una clave sin passphrase que ingresar al momento de desear utilizarla.

Dentro del directorio podemos encontrar los siguientes archivos de configuración:

authorized_keys	Este archivo contiene una lista de claves públicas autorizadas. Cuando un cliente se conecta al servidor, el servidor autentica al cliente chequeando su clave pública firmada almacenada dentro de este archivo.
id_dsa	Contiene la clave privada DSA del usuario.
id_dsa.pub	La clave pública DSA del usuario.
id_rsa	La clave RSA privada usada por ssh para la versión 2 del protocolo SSH.
id_rsa.pub	La clave pública RSA usada por ssh para la versión 2 del protocolo SSH.
identity.	La clave privada RSA usada por ssh para la versión 1 del protocolo SSH.
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.

Archivo “sshd_config”

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

CONFIGURACIÓN DEL ARCHIVO SSHD_CONFIG

Este archivo lo podrá localizar en la siguiente ruta:

```
/etc/ssh/
```

El siguiente paso será abrir el archivo con la ayuda del editor de textos vi:

```
# vi /etc/ssh/sshd_config
```

A partir de este punto comenzaremos a blindar SSH.

CAMBIANDO EL PUERTO POR DEFECTO

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 archivo de configuración shd_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 34765, por lo que tendrá que editar el parámetro Port del archivo de configuración el cual deberá quedar así:

```
# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options change a
# default value.

Port 34567
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

DESACTIVANDO EL PROTOCOLO 1

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 más, 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 lo que tendrá que editar el parámetro Protocol del archivo de configuración el cual deberá quedar así:

```
# Disable legacy (protocol version 1) support in the server for new
# installations. In future the default will change to require explicit
# activation of protocol 1
Protocol 2
```

DESHABILITANDO EL ACCESO A ROOT

Este es quizá el parámetro más 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 más 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 archivo de configuración, el cual deberá quedar de la siguiente manera:

```
# Authentication:
#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
```

DEFINIENDO UN NÚMERO MÁXIMO DE INTENTOS DE CONEXIÓN

Muchos de los ataques llevados a cabo por piratas informáticos se basan en fuerza bruta, estableciendo un número máximo de intentos de conexión lograremos que sus intentos por entrar a nuestro servidor sean disuadidos.

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

```
# Authentication:
#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
MaxAuthTries 2
```

El número 2 indica la cantidad de veces que podemos equivocarnos al ingresar el usuario y/o contraseña, en este caso después de dos intentos, se perderá o cerrará la conexión. Claro, es totalmente posible volver a intentarlo, pero con solo dos intentos por vez.

ACTIVANDO EL MODO ESTRICTO

La opción StrictModes debe activarse para que, por ejemplo, los usuarios que establecen permisos de escritura para todos en sus archivos y directorios, no se lleven una desagradable noticia cuando otro usuario los modifique. De esta manera se protege la información de los usuarios.

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

```
# Authentication:
#LoginGraceTime 2m
PermitRootLogin no
StrictModes yes
MaxAuthTries 2
```

IMPIDIENDO LA CONEXIÓN AL SERVIDOR GRÁFICO

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

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

```
# Accept locale-related environment variables
AcceptEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY
LC_MESSAGES
AcceptEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
AcceptEnv LC_IDENTIFICATION LC_ALL
#AllowTcpForwarding yes
#GatewayPorts no
#X11Forwarding yes
```

LIMITANDO EL TIEMPO PARA AUTENTICARSE CON SSH

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 aún, que alguien esté intentando mediante un script varias veces adivinar un usuario y contraseña.

Si somos el único usuario del sistema, considero que con 20 o 30 segundos es más que suficiente.

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

```
# Authentication:
LoginGraceTime 30
PermitRootLogin no
StrictModes yes
MaxAuthTries 2
```

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:

```
# /etc/init.d/sshd start
o Bien
# systemctl start sshd
```

Igualmente existen opciones ya sea para reiniciar, detener, recargar o conocer el status en el que se encuentra el servicio.

Estas opciones pueden ser consultadas en la siguiente tabla:

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 archivo 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.
status	Da a conocer el estado en el que se encuentra el servicio.

Ejemplos de utilización de OpenSSH

En esta parte del capítulo le enseñaremos a:

- Conectarse a un equipo remotamente a través de SSH.
- Copiar archivos o carpetas desde un equipo remoto.
- Enviar archivos o carpetas a un equipo remoto.

CONECTÁNDOSE A UN EQUIPO REMOTO A TRAVÉS DE SSH

Para establecer una conexión con un servidor SSH remoto desde Debian haremos uso de una terminal. La sintaxis para llevar a cabo esta operación es la siguiente:

```
# ssh usuario@ipDelServidor
```

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:

```
# ssh -p[puerto] usuario@ipDelServidor
```

Si quisiéramos redireccionar la ventana a nuestro equipo, esto es, si quisiéramos que la aplicación corra en el equipo remoto; pero la ventana se dibuje en nuestro escritorio:

```
# ssh -X usuario@ipDelServidor
```

La máquina cliente desde la que ejecutamos este comando debe estar corriendo un gestor de ventanas, por ejemplo gnome. La ventana se dibujara en nuestro equipo; pero no olviden que el proceso correrá en la máquina remota.

FINGERPRINT SSH

Algo que notaremos la primera vez que nos conectemos a un servidor será que nos dice algo de un fingerprint. ¿Qué es eso?

La idea de SSH es que nadie nos engañe al momento de conectarnos a un servidor, entonces la advertencia que nos muestra es que es la primera vez que nos conectamos a ese servidor y que el fingerprint o huella es la siguiente. Pero nuevamente ¿qué es eso?

Antes de continuar debemos aclarar que la huella está asociada a la clave pública del servidor. O sea como cada servidor tiene su clave pública, cada fingerprint será diferente.

Como es la primera vez que nos conectamos, nos está avisando que no tiene ese fingerprint. Si somos muy paranoicos podremos llamar al administrador del server al que nos estamos queriendo conectar, quien ejecutando el siguiente comando en el mismo nos dirá el fingerprint que tiene y verificar si coincide:

```
# cd /etc/ssh
# for file in *sa_key.pub; do ssh-keygen -lf $file; done
```

Esto nos mostrará el fingerprint por cada una de las claves que tiene, si alguna salida coincide con lo que nos mostró del lado del cliente. Una vez que se acepte el fingerprint, el mismo se almacenará en la carpeta:

```
/home/usuario/.ssh/known_hosts
```

A partir de ese momento, cada vez que nos conectemos a esa URL/IP, siempre deberá tener el mismo fingerprint, lo que implicará que siempre tendrá la misma clave pública, lo que implica que tendrá la misma clave privada y como esta no se divulga implicará que es siempre el mismo server.

Si alguien quisiera hacerse pasar por el servidor al que nos queremos conectar para robar tráfico, cuando el cliente detecta que el fingerprint es distinto a la IP que tenía almacenada en el known_hosts nos denegará el acceso.

Otro ejemplo más común que el anterior es cuando se le asigna la IP a otro servidor o cuando reinstalamos ese servidor. En ambos casos nos dará la misma alerta. Si este es el caso, deberemos ingresar a la línea known_hosts que nos indica y eliminar esa línea, o si es posible ejecutar el siguiente comando:

```
ssh-keygen -R hostname
```

Un ejemplo de dicho error es el siguiente:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle
attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
5c:9b:16:56:a6:cd:11:10:3a:cd:1b:a2:91:cd:e5:1c.
Please contact your system administrator.
Add correct host key in /home/user/.ssh/known_hosts to get rid of
this message.
Offending key in /home/user/.ssh/known_hosts:1
RSA host key for ras.mydomain.com has changed and you have reques-
ted strict checking.
Host key verification failed.
```

EJEMPLO 1:

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

```
IP del servidor -> 207.249.25.60
Nombre del usuario remoto -> adminlog
Puerto de autenticacion -> 34765
```

Solución:

01 Para conectarnos al servidor remoto habrá que especificar el puerto de escucha, el usuario remoto y la IP del servidor remoto. Recuerde que no está permitido conectarse como root desde SSH.

02 El siguiente paso será teclear la contraseña del usuario remoto.

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

```
[adminlog@ web ]# su -
Contraseña: xxxxxxxxxxxxxxxxx
[root@ web ]#
```

04 Por último, solo bastara reiniciar el servidor de apache:

```
[root@ web ]# /etc/init.d/apache2 restart
```

05 Para salir del SSH solo basta teclear “exit”:

```
[root@ web ]# exit
Connection to 207.249.25.60 closed.
# _
```

COPIAR U OBTENER ARCHIVOS O CARPETAS DESDE UN EQUIPO REMOTO

Para copiar archivos o carpetas desde un equipo remoto hacia nuestro equipo existen dos maneras:

- Mediante el uso del comando SCP
- Mediante el uso del comando SFTP

COPIANDO ARCHIVOS A TRAVÉS DE SCP (SHELL SECURE COPY)

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

La diferencia en utilizar SCP (Shell) y SFTP (Security File Transfer Protocol) para copiar carpetas o archivos radica en que para SCP tenemos que conocer exactamente donde se encuentra el recurso que queremos copiar, de otra forma nunca lo descargará. En cambio SFTP nos deja navegar entre las carpetas, lo cual hace más sencillo la ubicación del recurso que deseamos copiar. La única desventaja que presenta SCP es que únicamente permite la transferencia de archivos (descarga y subida de archivos).

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

```
#scp usuario@ipDelServidor:/rutaDelRecurso
```

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 1:

```
#scp -P[puerto] usuario@ipDelServidor:/rutaDelRecurso
```

Aunado a esto, para descargar una carpeta tendrá que seguir la siguiente sintaxis:

```
#scp -P[puerto] -r usuario@ipServidor:/rutaDelDirectorio
```

EJEMPLO 2:

La misma empresa nos ha pedido copiar la carpeta de inventarios de la empresa la cual esta hospedada en un servidor remoto, para ello nos ha proporcionado los siguientes datos:

```
IP del servidor -> 111.111.111.111
Nombre del usuario remoto -> adminlog
Puerto de autenticacion -> 34567
Ruta del Recurso Remoto -> /tmp/Conta
```

Solución:

01 Para poder hacer la copia desde servidor remoto habrá que especificar el puerto de escucha, el usuario remoto, la IP del servidor remoto y la ruta (sin errores) del recurso remoto.

02 Lo anterior nos copiará la carpeta “/tmp/Conta” remota en el directorio actual “.” naturalmente siempre que usuario tenga permisos sobre la carpeta y su cuenta esté entre las de los que pueden hacer ssh.

03 La opción “-r” significa recursivo, es decir, copia la carpeta y todo su contenido, incluidas las subcarpetas y el contenido de éstas.

COPIANDO ARCHIVOS A TRAVÉS DE SFTP (SECURITY FILE TRANSFER PROTOCOL)

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 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:

```
# sftp usuario@ipDelServidor
```

01 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:

```
# sftp -o Port=[PuertoDeEscucha] usuarioRemoto@ipDelServidorRemoto
```

02 El siguiente paso será autenticarnos con la contraseña del usuario remoto:

```
Connecting to IPDelServidorRemoto...
usuarioRemoto@IPDelServidorRemoto’s   password:  xxxxxxxxxxxxxxxxxxx
```

03 Una vez dentro del servidor solo bastará ejecutar el comando “get” para descargar algún archivo.

```
[root@ localhost]# sftp -o Port=[PuertoDeEscucha] usuarioRemoto@
ipDelServidorRemoto
Connecting to IPDelServidorRemoto...
usuarioRemoto@IPDelServidorRemoto’s    password:  xxxxxxxxxxxxxxxxxx
sftp> get recursoRemoto
```

La siguiente tabla explica más a detalle los comandos que pueden ser utilizados con SFTP:

cd [rutaRemota]	Cambia de directorio dentro del servidor remoto.
lcd [rutaLocal]	Cambia de directorio en el equipo local.
chgrp [grp] [rutaRemota]	Cambia el grupo de trabajo de un archivo remoto. El [grp] tiene que ser un Group ID.
chmod [opciones] [rutaRemota]	Cambia los permisos de Lectura, Escritura o de Ejecución a un archivo remoto.
chown [own] [rutaRemota]	Cambia el grupo de trabajo de un archivo remoto. El [own] tiene que ser un User ID.
get [rutaRemota] [rutaLocal]	Copia un recurso remoto en un equipo local.
lmkdir [rutaLocal]	Crea una carpeta en el equipo local.
lpwd	Imprime la ruta local en la cual estamos trabajando.
mkdir [rutaRemota]	Crea una carpeta en el equipo remoto.

put [rutaLocal] [rutaRemota]	Sube un archivo o archivo desde una ruta local hasta una ruta remota.
pwd	Imprime la ruta remota en la cual estamos trabajando.
exit	Salimos de SFTP.
Rename[rutaLocal][rutaRemota]	Renombra un archivo remoto.
rmdir [rutaRemota]	Borra una carpeta remota.
rm [rutaRemota]	Borra un archivo remoto.

EJEMPLO 3:

La misma empresa nos ha pedido copiar el archivo inventarioEnero.odt que se encuentra dentro la ruta “/tmp/Conta/”, para ello nos ha proporcionado los siguientes datos:

```
IP del servidor remoto -> 111.111.111.111
Nombre del usuario remoto -> adminlog
Puerto de autenticacion -> 34567
Ruta del Recurso Remoto -> /tmp/Conta
```

Solución:

01 Para poder traer la copia desde servidor remoto hacia nuestro equipo habrá que especificar el puerto de escucha, el usuario remoto y la IP del servidor remoto.

02 Nos pedirá autenticarnos con la contraseña del usuario remoto, en este caso la contraseña del usuario “adminlog”.

03 Una vez autenticados con el servidor nos dará acceso a través de SFTP.

04 Nos moveremos entre directorios con la ayuda del comando “cd” hasta estar ubicados en “/tmp/Conta”.

05 Dentro de la carpeta “Conta” aplicar el comando “dir” para visualizar el contenido de la misma.

```
sftp>dir
inventarioEnero.odt  inventarioFebrero.odt  inventarioMarzo.odt
```

06 Con la ayuda del comando “get” descargaremos el archivo nombrado “inventarioEnero.odt” dentro de la carpeta “home” de nuestro sistema.

SUBIR O ENVIAR ARCHIVOS O CARPETAS A UN EQUIPO REMOTO

Para subir archivos o carpetas desde nuestro equipo hacia un equipo remoto existen dos maneras:

- Mediante el uso del comando SCP
- Mediante el uso del comando SFTP

ENVIANDO ARCHIVOS A TRAVÉS DE SCP (SHELL SECURE COPY)

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

```
[root@localhost]#scp rutaDelRecursoLocal usuario@ipDelServidor:/
ruta
```

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]#scp -P[puerto] rutaDelRecursoLocal usuario@ip-
DelServidor:/ruta
```

Aunado a esto, para subir una carpeta tendrá que seguir la siguiente sintaxis:

```
[root@localhost]#scp -P[puerto] -r directorioLocal usuario@ipDel-
Servidor:/ruta
```

EJEMPLO 4:

Se nos ha pedido subir una actualización referente a la página web de la empresa, para ello nos ha proporcionado los siguientes datos:

```
IP del servidor remoto -> 111.111.111.111
Nombre del usuario remoto -> adminlog
Puerto de autenticacion -> 34567
Ruta del Servidor a donde se tiene que subir la información -> /
tmp/Conta
```

Solución:

01 Para subir este directorio al servidor remoto habrá que especificar la ruta del directorio local, el puerto de escucha, el usuario remoto, la IP del servidor remoto y la ruta (sin errores) a donde se quiere enviar el directorio.

02 Luego de haber hecho esto nos pedirá autenticarnos con la contraseña del usuario remoto:

```
adminlog@111.111.111.111’s password:xxxxxxxxxxxx
```

03 Al finalizar nos mostrara un ventana mostrando el progreso de cada copia hecha al servidor remoto como la que se muestra a continuación.

```
Actualizacion1.html      100%      0.0KB/s        05:00
actualizacion2.html      100%      0.0KB/s        07:00
actualizacion3.html      100%      0.0KB/s        15:00
actualizacion4.html      100%      0.0KB/s        15:00
actualizacion5.html      100%      0.0KB/s        25:00
actualizacion6.html      100%      0.0KB/s        30:00
actualizacion7.html      100%      0.0KB/s        31:00
actualizacion8.html      100%      0.0KB/s        40:00
```

ENVIANDO ARCHIVOS A TRAVÉS DE SFTP (SECURITY FILE TRANSFER PROTOCOL)

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

```
# sftp usuario@ipDelServidor
```

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:

```
# sftp -o Port=[Puerto] usuario@ipDelServidor
```

El siguiente paso será autenticarnos con la contraseña del usuario remoto:

```
Connecting to IPDelServidorRemoto...
usuarioRemoto@IPDelServidorRemoto’s password:  xxxxxxxxxxxxxxx
```

Una vez dentro del servidor solo bastara ejecutar el comando “put” para descargar algún archivo:

```
# sftp -o Port=[PuertoDeEscucha] usuarioRemoto@ipDelServidorRemoto
Connecting to IPDelServidorRemoto...
usuarioRemoto@IPDelServidorRemoto’s password:  xxxxxxxxxxxxxxx
sftp> put recursoLocal
```

La siguiente tabla explica más a detalle los comandos que pueden ser utilizados con SFTP:

cd [rutaRemota]	Cambia de directorio dentro del servidor remoto.
lcd [rutaLocal]	Cambia de directorio en el equipo local.
chgrp [grp] [rutaRemota]	Cambia el grupo de trabajo de un archivo remoto. El [grp] tiene que ser un Group ID.
chmod [opciones] [rutaRemota]	Cambia los permisos de Lectura, Escritura o de Ejecución a un archivo remoto.

chown [own] [rutaRemota]	Cambia el grupo de trabajo de un archivo remoto. El [own] tiene que ser un User ID.
get [rutaRemota] [rutaLocal]	Copia un recurso remoto en un equipo local.
lmkdir [rutaLocal]	Crea una carpeta en el equipo local.
lpwd	Imprime la ruta local en la cual estamos trabajando.
mkdir [rutaRemota]	Crea una carpeta en el equipo remoto.
put [rutaLocal] [rutaRemota]	Sube un archivo o archivo desde una ruta local hasta una ruta remota.
pwd	Imprime la ruta remota en la cual estamos trabajando.
exit	Salimos de SFTP.
rename [rutaLocal] [rutaRemota]	Renombra un archivo remoto.
rmdir [rutaRemota]	Borra una carpeta remota.
rm [rutaRemota]	Borra un archivo remoto.

EJEMPLO 5:

Se nos ha pedido subir una actualización referente a la página web de la empresa, pero esta vez será usando SFTP, para ello nos ha proporcionado los siguientes datos:

```
IP del servidor remoto -> 111.111.111.111
Nombre del usuario remoto -> adminlog
Puerto de autenticacion -> 34567
Ruta del Servidor a donde se tiene que subir la información -> /
tmp/Conta
```

Solución:

01 Para subir este directorio al servidor remoto habrá que especificar el puerto de escucha, el usuario remoto y la IP del servidor remoto.

02 Nos pedirá autenticarnos con la contraseña del usuario remoto, en este caso la contraseña del usuario “adminlog”.

03 Una vez autenticados con el servidor nos dará acceso a través de SFTP.

04 Nos moveremos entre directorios con la ayuda del comando “cd” hasta estar ubicados en “/tmp/Conta”.

05 Dentro de la carpeta “Conta” aplicar el comando “lpwd” para verificar la ruta en la cual estamos ubicados localmente:

```
sftp> lpwd
Local working directory: /home/juanito
```

06 Si no se encuentra ubicado en el directorio de trabajo indicado cámbiese de directorio mediante el comando “lcd”:

```
sftp> lcd /home/juanito/datosActualizados
lcd /home/juanito/datosActualizados
```

07 Cuando esté ubicado en el directorio de trabajo que contiene la información que desea subir al servidor remoto teclee lo siguiente:

```
sftp> put datosactuales
```

08 El comando “put” tiene la funcionalidad de subir archivos desde una maquina local hasta un equipo remoto.

09 Por ultimo escribimos la palabra exit para salir del “SFTP”:

```
sftp> exit
[root@localhost ]#
```