ssh

### Instalación de un servidor SSH

```
yum install openssh-server
```

#### Configuración

El fichero de configuración está en /etc/ssh/sshd_config

Es recomendable hacer los siguientes cambios:

- Cambiar el puerto 22 por otro→ Port 22022
- Usar sólo la versión 2 del protocolo→ Protocol 2
- Denegar iniciar la sesión como root → PermitRootLogion no
- Definir la interfaz por la que vamos a escuchar (en caso de tener varias) → ListenAddress 192.168.1.10

## Uso del SSH

la forma de conectar por ssh es

```
ssh ctausuario@ip_destino
```

Si en vez del puerto 22 se usa otro puerto

```
ssh -p puerto ctausuario@ip_destino
```

## Configurar SSH

La configuración del servicio ssh se encuentra en /etc/ssh/sshd_config

- Protocol: Versión a usar del protocolo ssh, lo mejar es usar la versión 2
- LoginGraceTime: tiempo para hacer login
- PermitRootLogin: este valor indica si es posible hacer login como root, lo mejor es poner que no y que el usuario ejecute sudo si lo necesita
- AllowUsers: podemos indicar que usuarios si pueden conetarse por ssh e incluso indicar desde que ip se permite. Basta poner el nombre del usuario o con usuario@ip
- MAxAuthTries: Número máximo de intentos para hacer login
- MaxStartups: Número máximo de usuarios connctados simultaneamente
- ==== Desactivar el login del root por ssh ===

# Proxy Socks con SSH

Podemos crear un proxy a partir de una conexión ssh con un comando del tipo:

ssh -D 9999 USUARIO@IP

Y en el navegador seleccionar en la configuración del proxy 'localhost' como servidor SOCKS y 9999 como puerto.

Otra opción es ejecutar

```
$ ssh -f -N -D 9999 usuario@hostremoto
```

Con el flag -f ejecutamos SSH en segundo plano. Con el flag -N le decimos que no vamos a ejecutar ningún comando, por lo que no nos dará acceso a la consola. El flag -D es el que crea una redirección de puertos local a nivel de aplicación.

Están soportadas las versiones SOCKS4 y SOCKS5. La principal diferencia entre las dos es que la versión 5 incorporando autenticación. Sólo el root puede redirigir puertos bien conocidos.

Si una aplicación no soporta el proxy, podemos usar **tsocks**. tsocks, permite que cualquier aplicación utilice este tipo de proxies de forma transparente.

Lo instalamos

```
sudo apt-get install tsocks
```

, lo configuramos el fichero /etc/tsocks.conf:

```
server = 127.0.0.1
server_type = 5
server_port = 9999
```

Para utilizarlo :

```
tsocks telnet micorreo.org 25
```

Para las aplicaciones en java hay que ejecutar lo siguiente

```
java   -DsocksProxyHost=127.0.0.1   -DsocksProxyPort=1080
MiAplicacionJava
```

podemos usar autossh en vez de ssh para aquellos casos en que se nos corte la comunicación ya que con autossh el sólo vuelve a reconectar

# Túneles ssh

En este ejemplo, vamos a conectarnos por ssh al equipo 99.99.99.99 por el puerto 9999 y como usuario “bender” y redirigiremos el puerto 5900 (el puerto por defecto para vnc) del equipo de la red remota con ip 192.168.0.99 al puerto 1111 de nuestra máquina:

ssh -p 9999 -P -L 1111:192.168.0.99:5900 bender@99.99.99.99

Como resultado, si tenemos el vnc server trabajando en 192.168.0.99, podríamos acceder a él por el puerto forwadeado:

127.0.0.1:1111

Para conectarnos a otra máquina por ssh:

ssh -p puerto usuario@ip

Si queremos copiar un fichero:

scp -P puerto fichero usuario@ip:directorio

Si queremos ejecutar aplicaciones gráficas añadimos el parámetro -X:

ssh -X -p puerto usuario@ip

En caso de querer ejecutar un navegador por este método, es recomendable usar epiphany, pues es mucho más liviano que otros.

# SSH Transparente

¿Cómo hacer que el ssh te valide sin pedirte las contraseñas (con la clave pública y la privada)?

- Asegurarse de que se tiene en ambas máquinas (maq1 y maq2) una versión de openssh actualizada y de que es la misma versión. (No tiene porque ser obligatorio en versiones recientes).
- Si queremos que el usuario pepe se conecte a la maq2 desde la maq1 con un 'ssh maq2' hacer lo siguiente:
  - Desde maq1 pepe hacer:

```
ssh-keygen -t rsa
```

.

Dejar el nombre fichero que dice y no poner passphrase.

Se habrán generado dos ficheros fichero1.pub y fichero2.rsa con la clave privada y la pública en maq1, más concretamente en el subdirectorio /home/pepe/.ssh/ . Necesitamos copiar el contenido del

primer fichero1.pub al otro ordenador, específicamente al archivo authorized_keys dentro de ~/.ssh/

Por tanto o añadimos al fichero /home/pepe/.ssh/authorized_keys en maq2 la línea que contiene la clave pública de maq1 (fichero .pub), o bien lo creamos si no existe.

OJO , si la cortamos y pegamos, tener cuidado que este en una sola linea!!

- Asegurarse que el directorio .ssh de la maquina destino tenga como permisos drwx——. Es decir, todos los permisos solo para ese usuario, sino esto no funciona!!!
- Probar y hacer desde maq1 ssh maq2.
- Esta configuración es para generación de claves tipo rsa de nivel 2. Las hay rsa de nivel 1 y dsa. La configuración para ambas es diferente. Para enterarse bien hacer un man ssh y man ssh-keygen.
- Si no funciona, en /etc/ssh/sshd_config debería estar descomentado y activado: RSAAuthentication yes y PubkeyAuthentication yes.

```
ssh usuario@maquina
rsync –e ssh –vrclHptogt –delete –force –stats –progress
servidor:/var/lib/prueba
```

# ssh en nautilus

Cuando estamos trabajando en otra máquina a través de una conexión ssh, ¿no es una lata tener que andar copiando ficheros con scp?

Una alternativa mucho más cómoda es iniciar sesión en dicha máquina con nuestro navegador de archivos y arrastrar y soltar con el ratón. Para ello, en la barra de direcciones de nautilus escribiremos:

```
sftp://usuario@direccion_ip:puerto/directorio
```

Por ejemplo:

```
sftp://lc@192.168.2.1:5566/home/lc
```

Al darle a enter nos solicitará la contraseña y accederemos a la máquina. Si usamos el puerto por defecto para las conexiones ssh, y el usuario tiene permisos de lectura en la carpeta raíz, podemos obviar los datos puerto y directorio.

## Copiar ficheros

```
 scp ruta/archivo cuenta_en_ordenador_presente@ip_ordenador_presente:
ruta/fichero
```

## Referencias

- http://blogofsysadmins.com/crear-un-proxy-socks-usando-un-tunel-ssh
- http://blogofsysadmins.com/secure-shell-hacks-linux
- http://submarley.espacioblog.com/post/2008/11/04/shh-dios-la-administracion-remota
- http://linuxamartillazos.blogspot.com/search/label/ssh
- http://www.vicente-navarro.com/blog/2009/05/24/creando-tuneles-tcpip-port-forwarding-con-ssh-los-8-escenarios-posibles-usando-openssh/
- http://terminus.ignaciocano.com/k/2011/08/12/utilizar-ssh-para-establecer-un-servidor-proxy-socks/
- http://hpantaleev.wordpress.com/2012/02/14/openssh-sftp-chroot-con-chrootdirectory/
- http://bootlog.org/blog/linux/tip-ssh-scp-y-un-as-bajo-la-manga
- http://www.alcancelibre.org/staticpages/index.php/como-ssh-clave-publica

From:
https://intrusos.info/ - **LCWIKI**

Permanent link:
**https://intrusos.info/doku.php?id=linux:ssh&rev=1459972245**

Last update: **2023/01/18 13:55**