

# openSSH

Alvaro Marín Illera [split77@terra.es](mailto:split77@terra.es)

*Instalación, configuración y manejo de openSSH*

## Introducción

OpenSSH (desarrollada por openBSD) es la versión libre (licencia BSD) del protocolo SSH. Permite entre otras cosas, la conexión remota a una máquina, para poder ejecutar comandos sobre ella, al igual que telnet o rlogin, pero con la ventaja respecto a los anteriores programas de que la información va cifrada. Esto es, en el caso de que alguien de la misma red en la que nos encontramos, esté usando un sniffer para "capturar" las contraseñas para conectarnos a un ftp o a una sesión telnet, con ssh, estará perdiendo el tiempo ya que van cifradas.

Pero además de esto, con ssh podremos establecer sesiones seguras con servidores X, smtp, pop3... mediante [túneles SSH](#).

Para la autenticación, ssh puede utilizar algoritmo de cifrado como RSA o DSA. Para el envío de datos a través de la red, usa 3DES, IDEA, Blowfish...

Soporta ambas versiones del protocolo ssh, la 1 y la 2. Dependiendo de cuál usemos los métodos de autenticación, son distintos:

- ssh v1:
    - si la máquina que se va a conectar está en /etc/hosts.equiv o /etc/shosts.equiv del servidor y el username es igual en ambas máquinas, se permite el login. Luego si .rhosts o .shosts existe en el servidor y contiene una línea con el nombre de usuario del cliente, se permite el login.
- Este método "en solitario", no es permitido por medidas de seguridad.
- este segundo método es el anterior junto con autenticación por host mediante claves RSA. Consiste en que si el login es permitido por /etc/hosts.equiv, /etc/shosts.equiv, \$HOME/.rhosts o \$HOME/.shosts y el servidor puede verificar la "clave de host" del cliente (ver /etc/ssh/ssh\_known\_hosts y \$HOME/.ssh/known\_hosts) , será admitida la conexión.
  - el tercer método es autenticación por RSA, es decir, el cliente crea su clave pública y su clave privada; el servidor conocerá la clave pública de cada usuario. El archivo \$HOME/.ssh/authorized\_keys contiene las claves que son permitidas para el login. Al conectarse, el cliente le dice al servidor con qué pareja /pub/priv quiere autenticarse. El servidor le envía un "desafío" al cliente, y si el cliente lo descifra y envía correctamente la respuesta, se permite el login.

El cliente, generará sus claves mediante el programa ssh-keygen y se guardan en \$HOME/.ssh/identity (para la clave privada) y \$HOME/.ssh/identity.pub. El cliente debe copiar el archivo identity.pub en su home del servidor, más concretamente en HOME/.ssh/authorized\_keys . Después de esto, se puede conectar sin necesidad de password.

- Si fallan, autenticación por contraseña.
- ssh v2:
- Cuando el cliente se conecta usando el protocolo v2 de ssh, se usan métodos similares de autenticación que en v1.

Primeramente se intenta con la autenticación de host, luego por clave pública/privada y finalmente por password.

El método que hemos visto en la v1 de autenticación por RSA, cambia un poco en esta segunda versión, ya que podemos usar tanto RSA como DSA. El cliente tendrá su clave privada `$HOME/.ssh/id_dsa` o `$HOME/.ssh/id_rsa`. El servidor también tendrá que comprobar las claves en `$HOME/.ssh/authorized_keys`.

Dentro del paquete 'ssh' vienen bastantes programas adicionales, a parte del servidor y el cliente:

- sshd : servidor de ssh
- sftp-server : servidor ftp mediante ssh. Para usarlo, añadir "Subsystem sftp /usr/lib/sftp-server" en sshd\_config
- ssh : el cliente, con él nos podemos conectar al servidor sshd.
- scp : copia archivos entre máquinas. Sustituto para rcp.
- ssh-keygen : para crear claves públicas y privadas RSA o DSA (host keys y user authentication keys).
- ssh-keyscan : utilidad para obtención de claves públicas de hosts
- ssh-copy-id : copia el identify.pub en una máquina remota
- ssh-agent : agente de autenticación. (Usado para manejar RSA keys en la autenticación.)
- ssh-add : para añadir nuevas claves con el agente.
- sftp : cliente ftp mediante ssh

## Configuración del servidor sshd

Vamos al directorio `/etc/ssh`. El archivo de configuración del servidor se llama `sshd_config`. Si existe un archivo llamado `sshd_not_to_be_run` significa que habremos elegido en la instalación la opción de no arrancar el demonio para aceptar conexiones. Borrando dicho archivo y reiniciando el ssh, tenemos ya el servidor a la escucha en el puerto 22.

Pasamos a comentar las opciones del fichero de configuración:

```
# Definimos el puerto por el que escuchamos las peticiones de conexión
del los clientes
```

```
Port 22
```

```
# Especificamos por qué direcciones locales escucharemos.
```

```
# Use these options to restrict which interfaces/protocols sshd will bind
to
```

```
#ListenAddress ::
```

```
#ListenAddress 0.0.0.0
```

```
# Especificamos las versiones del protocolo que aceptaremos
```

```
Protocol 2,1
```

```
# Archivos que contienen las llaves privadas de host a usar (solo
permisos rw- para root)
```

```
# HostKeys for protocol version 1
```

```
HostKey /etc/ssh/ssh_host_key
```

```
# HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key

# Cada cuantos segundos el servidor regenera las claves, para así evitar
el que se puedan
# descifrar sesiones caputaradas y luego usarse. Nunca es guardada en
disco.
KeyRegenerationInterval 3600

# Especifica el nº de bits para la clave del servidor del protocolo v1
ServerKeyBits 768

# Tipo de logeo de actividades -> AUTH=>/var/log/auth.log También
disponible DAEMON, USER...
# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
# El servidor desconecta al cliente si no se ha logeado correctamente en
600 segs.
LoginGraceTime 600
# Permitimos o no el login del usuario root => poner a no!
PermitRootLogin no
# SSH comprobará que los archivos tienen los permisos correctos, para
evitar tener archivos con todos
# los permisos a todos los usuarios
StrictModes yes

# Permitimos la autenticación por RSA (solo para v1)
RSAAuthentication yes

# Permitimos la autenticación por clave pública (solo para v2)
PubkeyAuthentication yes

# Contiene el archivo que contiene las claves para la autenticación
#AuthorizedKeysFile %h/.ssh/authorized_keys

# NO usaremos el método de autenticación por rhost (.rhost)
# rhosts authentication should not be used
RhostsAuthentication no

# Don't read the user's ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
# For this to work you will also need host keys in /etc/ssh_known_hosts
RhostsRSAAuthentication no

# No usamos el protocolo de autenticación por host
# similar for protocol version 2
HostbasedAuthentication no

# Uncomment if you don't trust ~/.ssh/known_hosts for
```

```
RhostsRSAAuthentication
#IgnoreUserKnownHosts yes

# To enable empty passwords, change to yes (NOT RECOMMENDED)
PermitEmptyPasswords no

# Uncomment to disable s/key passwords
#ChallengeResponseAuthentication no

# HAbilitamos la autenticación "normal" si todo falla
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes

# Use PAM authentication via keyboard-interactive so PAM modules can
# properly interface with the user
PAMAuthenticationViaKbdInt yes

# Opciones para servidor Kerberos
# To change Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#AFSTokenPassing no
#KerberosTicketCleanup no
# Kerberos TGT Passing does only work with the AFS kserver
#KerberosTgtPassing yes

# En un principio, no permitimos exportar las X
X11Forwarding no
# Especificamos el primer nº de pantalla disponible para exportar
X11DisplayOffset 10

# Si queremos que al hacer login, al usuario le salga /etc/motd
PrintMotd no

# Por defecto, enseñaremos los datos del último login
#PrintLastLog no

# Para que el servidor pueda "enterarse" de que el cliente ha caído por
ej
KeepAlive yes

#
#UseLogin no

#MaxStartups 10:30:60

# Si queremos enviar un banner cuando un cliente se conecte
#Banner /etc/issue.net

#ReverseMappingCheck yes

# Servidor ftp por ssh
```

```
Subsystem sftp /usr/lib/sftp-server
```

## Autenticación por claves RSA/DSA

Vamos a configurar ssh para poder acceder al servidor de forma segura y sin password, es decir, mediante claves RSA/DSA.

Debemos asegurarnos que en archivo de configuración del servidor, tenemos la opción: `PubkeyAuthentication yes`

En el cliente, generamos las claves DSA:

```
ssh-keygen -t dsa
```

Podemos poner una password, de tal forma que ésta sería la que habría que introducir cada vez que queramos conectarnos a la máquina remota. En nuestro caso, la dejaremos en blanco.

Vamos al directorio `.ssh` de nuestro home y vemos que nos ha generado dos archivos:

`id_dsa` (llave privada)

`id_dsa.pub` (llave publica)

Es la clave pública la que deberemos mandar al servidor, para ello usamos `scp`:

```
$scp archivo_local user@host:archivo_remoto
scp id_dsa.pub alvaro@intranet:~/.ssh/id_dsa.pub
The authenticity of host 'intranet (192.168.200.70)' can't be
established.
RSA key fingerprint is d8:d7:0a:cf:d9:8f:fe:fa:66:f0:76:46:46:fa:b8:1a.
Are you sure you want to continue connecting (yes/no)?
```

Vemos que no puede establecer la autenticación por host, y por tanto nos pedirá la contraseña. El fingerprint, es la "firma digital" del servidor. La aceptamos, y se nos copiará en el archivo `.ssh/known_hosts` de la máquina local. De esta forma, cada vez que nos conectemos al servidor, se comprobará que ambas "firmas" coinciden.

```
alvaro@intranet's password:
id_dsa.pub          100%
|*****|          605          00:00
```

Ya tenemos nuestra clave pública copiada en el servidor, en el directorio `.ssh` del home del usuario. Ahora conectándonos al servidor, hacemos lo siguiente:

```
cd /home/usuario/.ssh/
cat id_dsa.pub_CLIENTE >> authorized_keys2
rm id_dsa.pub_CLIENTE
```

A partir de este momento, ya podemos hacer desde el cliente, de la forma:

```
ssh usuario@servidor
```

para conectarnos sin necesidad de password. Debemos asegurarnos que solo nosotros tenemos acceso a leer la clave privada, para ello damos al directorio, los permisos siguientes:

```
chmod 700 .ssh
```

Por otra parte, ssh-agent, es un demonio que se encarga de "cachear" las claves privadas. SSH se conectará con ssh-agent para pedirle la clave privada, así no tendremos que introducir la password cada vez que nos conectemos al servidor.

Para añadir una clave privada a ssh-agent utilizamos el comando ssh-add, de la siguiente forma:

```
ssh-add .ssh/id_dsa
```

y a partir de ahora, ya no tendremos que poner la pass cada vez.

Si queremos habilitar esta forma de autenticación, de tal modo que solo las máquinas con su clave en el servidor puedan entrar, podemos poner a "no", la línea de PasswordAuthentication.

## SSH Tunneling

Mediante ssh podemos hacer túneles para poder enviar la información a través de la red, es decir, podemos por ejemplo conectarnos al puerto 110 de un servidor para ver nuestro correo, de tal forma que los datos van cifrados. Esto se consigue mediante el "port-forwarding". Lógicamente, hace falta tener ssh en ambas máquinas.

```
ssh -P -f -L 1234:remoteserver:110 user@remoteserver sleep 25
```

Las opciones que hemos indicado a ssh son:

- -L especifica el "redireccionamiento" del puerto 110 de la máquina remota el 1234 de la local.
- -f indica que se ejecute en background
- -P nos permite usar puertos mayores que 1024, de este modo, podemos usar el tunel sin ser root sleep 25 indica el tiempo en segundos durante el que el tunel estará activo.

Cuando pase dicho tiempo, se esperará a que las conexiones existentes acaben.

Para poder descargar el correo por pop3, bastará conectarnos al puerto 1234 local. Los datos se enviarán y recibirán de forma cifrada a través del túnel.

Para más información sobre el tema, podeis consultar el artículo que hizo Pablo Garaizar:

<http://130.206.100.150/docs/articulo.ssh.html>

Este documento ha sido escrito por un miembro de e-GHOST y su contenido es libre de ser reproducido en otros medios bajo las condiciones de la [Licencia FDL \(GNU Free Documentation License\)](#).