Shell seguro (SSH)

El protocolo SSH (*Secured SHell*) se basa en servicios de autentificación y de confidencialidad para asegurar los intercambios entre los clientes y el servidor a través de un transporte seguro de datos. Lo podemos usar para conectarnos como terminal a un servidor SSH para una sesión en modo de línea de comandos (como una especie de Telnet seguro), pero también puede asegurar funcionalidades de tipo transferencia de datos segura (SFTP, SCP) así como el transporte seguro para otros protocolos aplicativos.

1. Usos de OpenSSH

El protocolo Telnet se ha utilizado durante mucho tiempo para permitir a los clientes que se conecten en modo terminal a los sistemas remotos, a través de redes TCP/IP. Pero ese protocolo no es seguro: todos los intercambios entre el cliente y el servidor atraviesan las redes sin cifrar, incluyendo la contraseña del usuario que solicita la conexión.

El protocolo SSH, más reciente, usa una comunicación cifrada entre el cliente y el servidor, y procede a una auténticación entre el cliente y el servidor, que garantiza la confidencialidad de los intercambios.

OpenSSH es una implementación open source del protocolo SSH, muy usada en los entornos Linux y BSD.

a. Configuración del servidor OpenSSH

El archivo de configuración por defecto del servidor es `/etc/ssh/sshd_config`. Contiene una serie de directivas bajo la forma:

```
Directiva Valor,...,Valor
```

Las directivas más habituales son las siguientes:

`Protocol`	Version del protocolo (a menudo la `2`).
`ListenAddress`	Direcciones en las que el servidor SSH espera las solicitudes de conexión (por defecto: todas las direcciones locales).
`Port`	Número del puerto en el que el servidor se pone en escucha (por defecto: `22`).
`PermitRootLogin`	Autoriza o no a un cliente a conectarse con la cuenta de superusuario (por defecto: `yes`).
`AllowUsers`	Lista de los usuarios autorizados. Los otros serán rechazados.
`DenyUsers`	Lista de los usuarios denegados. Los otros serán autorizados.
`PasswordAuthentication`	Autoriza o no la conexión con la contraseña del usuario (por defecto: `yes`).
`PubkeyAuthentication`	Autoriza o no la autenticación usando una clave pública.

b. Cifrado de las comunicaciones

El cifrado de los datos transmitidos entre el cliente y el servidor se basa en un par de claves que pertenecen al servidor SSH. Este último transmite su clave pública al cliente. El cliente usa esta clave para transmitir de manera cifrada una clave de sesión simétrica. Solamente el servidor podrá descifrar el mensaje, con su clave privada. La clave simétrica es entonces utilizada para cifrar los datos intercambiados durante la sesión entre el

cliente y el servidor.

El servidor almacena su par de claves en el directorio `/etc/ssh`.

2. Gestión de las autenticaciones

La comunicación entre un cliente y un servidor SSH empieza por una fase de autenticación de los dos sistemas. Puede hacerse por contraseña o por certificado.

a. Autentificación por contraseña y fingerprint

Cuando un cliente SSH solicita la conexión a un servidor SSH, tiene que proporcionar una cuenta de usuario válido que esté autorizado a conectarse, con una contraseña.

El cliente SSH, durante su primera conexión al servidor, toma la huella digital (*fingerprint*) recibida de este último y, después de la validación del cliente, la almacena en un archivo local del directorio de conexión del usuario (en el lado del cliente), `~/.ssh/known_hosts`. Para las conexiones siguientes, el cliente podrá asegurarse de que el servidor es el mismo de la conexión inicial.

Ejemplo

El archivo `known_hosts` contiene una línea por cada servidor conocido.

cat .ssh/known_hosts

```
|1|ylD8iNr8P/bPZa3YAPOcmmyBM3Y=|xhqCHYi4rAMhHXi8ZX4wi+w7GBg= ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBNKAuE2Suav2PBHXMbnEd92JzUuF3X8T
ZnO5nfogwJScY1xf2PsDYq2nY6P0xMpn8p2se//Em6gLZ4guq4MmlLY=
|1|7r5pFnApUtjikcWP5H7PszEdAgQ=|HpsP3FLI06OmIWpkSk6+lb8pp3k= ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBNKAuE2Suav2PBHXMbnEd92JzUuF3X8T
ZnO5nfogwJScY1xf2PsDYq2nY6P0xMpn8p2se//Em6gLZ4guq4MmlLY=
|1|7eXzpkN3TMKe7J8X3mQl+j1BhqE=|HIwneWpyEuRSMosodLVo9jy6V3o= ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBNKAuE2Suav2PBHXMbnEd92JzUuF3X8T
ZnO5nfogwJScY1xf2PsDYq2nY6P0xMpn8p2se//Em6gLZ4guq4MmlLY=
|1|HhHtERejU9MjrbEHqanXNdBbr98=|3oLtlk21FokBOm0sRT5W91IjFXQ= ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBNKAuE2Suav2PBHXMbnEd92JzUuF3X8T
```

```
ZnO5nfogwJScY1xf2PsDYq2nY6P0xMpn8p2se//Em6gLZ4guq4MmlLY=
localhost ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBNKAuE2Suav2PBHXMbnEd92JzUuF3X8T
ZnO5nfogwJScY1xf2PsDYq2nY6P0xMpn8p2se//Em6gLZ4guq4MmlLY=
debian10,192.168.0.70 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIb
mlzdHAyNTYAAABBBJM2jFqBIp+dtGOwKTJ4fj61GBN9CKlnjbbKNweCDvhPHV4yd8i4InA7Q+Ax7Q0bD16/
TiCMid4MwTzEtmwXUVQ=
```

b. Autentificación por claves

Un método más seguro para autenticar las conexiones SSH se basa en las claves de autentificación, almacenadas localmente en el sistema del cliente. La autentificación por claves dispensa de teclear la contraseña del usuario (excepto si una frase de contraseña se ha asociado a la clave), y garantiza al usuario la identidad del servidor remoto. Cada usuario debe tener un par de claves de autenticación.

c. Creación del par de claves en el cliente

Para que el servidor pueda ser formalmente identificado, tiene que proporcionar la clave pública del cliente. Esta clave le permite cifrar los datos, que solo podrán ser descifrados por el cliente, propietario de la clave privada.

Para generar el par de claves, privada y pública, en el sistema del cliente, utilizamos el comando `ssh-keygen`.

Sintaxis

```
ssh-keygen -t TipoCifrado
```

Donde `TipoCifrado` especifica el algoritmo que se va a emplear: RSA (versión 1 o 2 de SSH) o DSA (versión `2` de `SSH`). RSA y DSA son dos algoritmos de cifrado asimétrico. El algoritmo por defecto es RSA.

El comando, por defecto, solicita que se teclee una frase de contraseña. Esta es facultativa. Si se especifica, habrá que teclearla en cada conexión.

Ejemplo

Generamos un par de claves con el algoritmo por defecto (RSA) para el usuario `pba`.

```
id
uid=1000(pba) gid=1000(pba) groupes=1000(pba),100(users)
ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/pba/.ssh/id_rsa):
Created directory '/home/pba/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pba/.ssh/id_rsa.
Your public key has been saved in /home/pba/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:dgH4ebogubKCiFarn9UTMsQuV51n4lXa5IkxMP7kS+s pba@centos8
The key's randomart image is:
+---[RSA 3072]----+
|      ..o.o o    |
|    . . o.o X .  |
|     o o *.B +   |
|    o . + X.     |
|   . * .S+.+     |
|   .= =.o.. o    |
|+ . .+ + . o     |
|=...+   o .      |
|.o+=      E      |
+----[SHA256]-----+
```

El comando `ssh-keygen` crea dos archivos, por defecto en el directorio `.ssh` del directorio de conexión del usuario. Esos archivos son, por defecto, `id_rsa` para la clave privada y `id_rsa.pub` para la clave pública.

Ejemplo

Contenido de los dos archivos de claves.

ls -l .ssh
total 8
-rw-------. 1 pba pba 2602 18 junio 16:04 id_rsa
-rw-r--r--. 1 pba pba 565 18 junio 16:04 id_rsa.pub
cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCx1vQ5HlmLZC4XydSm6T2jBirjKeTjFT5eHOJfW0rJfP
OAWa0khQjyC9N6mB7E56yQX1XfPee21iPPFJFMLr4oiCNtfLx/6qGwD5cmX6eQisP3heeWAJ
RA4kkStXSOk1tzk7i9bIpXdVZW+dqbZN+rNGNDH3vrHiQtZNiiZVXieCmer0211OuLdD2jZ/
kjPIHbRu1Cv6OBOJz5PLYQLjqsF5znuOi76vdGsfx/JGrfJFj+/KXX+ctsb/18QRgncFkyb
7kbGaEzAq8xcwbPMRh6Ncrv+RMW2G0d2+yh95HmJd1e6Sg/brjzDznuY703t6LbP0eBg2uS5ZDD
RtZuTyW20hujm8y9parmX36rMAEB4xhbLKWJHno9FnRZ7740533u4iMsHSEanjW81fUTUW06CJk
y8jk++5xl8uuhX92IG17mkDP0rEERJ5U1vTKjvMPPDY2QmrfGCSjhmLYnfXuvFutZazXKGNjgs2
ikmhq9euc3E1CMto7qXqR+J24r1HE= pba@centos8
cat .ssh/id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAsdb0OR5Zi2QuF8nUpuk9owYq4ynk4xU+XhziX1tKyXzzgFmtJlUl
8gvTepgexOeskF9V3z3nttYjzxSRTC6+KIgjbXy8f+qhsA+XJl+nkIrD94XnlgCUQOJJEr
V0jpNbc5O4vWyKV3VWVvnam2TfqzRjQx976x4kLWTYomVV4ngpnq9NtdTri3Q9o2f5IzyB
20btQr+jgTic+Ty2EC46rBec57jou+r3RrH8fyRq3yRY/vyl1/nLbG/9fEEYJ3BZMm+5Gx
mhMwKvMXMGzzEYejXK7/kTFthtHdvsofeR5iXdXukoP2648w857mO9N7ei2z9HgYNrkuWQ
w0bWbk8lttIbo5vMvaWq5l9+qzABAeMYWyyliR56PRZ0We++NOd97uIjLB0hGp41vNX1E1
[...]
q6pL0binleMuD3fXL3Lb4QspPvY5bg1HsmRr7+2iZ39dkM+1T/mGuzzJ3f7cywETF8RJuo
pyGxCVGCmc70wYnVOHTqIsMgxJoeqdVQAAAMEAxEccoIxzmizZ4omEf2FZGYnZXP74Sp00
XvicE3Ho965CSaBRW6VJhfUmH90WSx8WHkZdLIGv8UDDOgMGOBB/roDVdD4P7JvV1R3BLB
nuBnyFNUpNuh5c48j5+GXrQ7PZ2fv9pwlqB1BdPU2fI2jX4D//bajqg3ShTfbvd5qja9sB
VBFNKy1qrAla015kTdRheelPeYdOkOze12ZkJfRb+PvI8trVcxVZOGFxHgB+I5RMSsLEzZ
0NOp+QIIO9EXqtAAAAC3BiYUBjZW50b3M4AQIDBAUGBw==
-----END OPENSSH PRIVATE KEY-----

Durante una solicitud de conexión por el usuario, el servidor SSH busca en el directorio de conexión de este usuario (en el sistema del servidor) si el archivo `.ssh/authorized_keys` existe y si contiene la clave pública del cliente. Si es el caso, la autenticación del servidor puede ser realizada por el cliente, a través del par de claves.

Para que este mecanismo funcione, el usuario que haya creado el par de claves en el sistema cliente tiene que añadir, antes de realizar esta autentificación, el contenido de su archivo de clave pública, en el directorio del conexión del lado del servidor, en el archivo `~/.ssh/authorized_keys`. Hay que posicionar los permisos de acceso en el directorio del archivo.

Ejemplo

Copia del archivo de clave pública creado en el sistema `centos8` (cliente) hacia el servidor `SSH debian10`, para el usuario `pba` (usamos el comando de copia de red seguro `scp`, ver más adelante).

En el cliente SSH `centos8`, con la cuenta `pba`, copiamos el archivo de clave pública hacia el directorio de conexión de `pba` en el servidor `debian10`:

```
scp .ssh/id_rsa.pub debian10:
The authenticity of host 'debian10 (192.168.0.70)' can't be established.
ECDSA key fingerprint is SHA256:Cbob+4M9v2AjgHYHBK/bSV3E7ZsJ+TCep/4lBkdHzGM.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'debian10,192.168.0.70' (ECDSA) to the list of known hosts.
pba@debian10's password:
id_rsa.pub
```

En `debian10`, con la cuenta `pba`:

```
mv id_rsa.pub .ssh/authorized_keys
chmod 640 .ssh/authorized_keys
chmod 770 .ssh
```

Desde el sistema `centos8`, el usuario `pba` se conecta a servidor `debian10`:

```
ssh debian10
Linux debian10 4.19.117 #2 SMP Thu Apr 23 10:04:02 CEST 2020 x86_64

The programs included with the Debian GNU/Linux system are free software;
```

```
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
No mail.
Last login: Thu Jun 18 16:12:56 2020 from 192.168.0.60
```

El servidor ha autorizado la conexión, sin solicitar la contraseña del usuario.

d. El agente SSH

Para los administradores o los usuarios que tengan necesidad de acceder a menudo a distintas máquinas a través del protocolo SSH, se puede configurar un agente SSH, ejecutado por el comando `ssh-agent sh`, para que conserve en memoria las claves privadas necesarias para las autenticaciones, así como la frase de contraseña. Esta claves privadas se transmiten una vez al agente usando el comando `ssh-add`. Las claves están disponibles a continuación automáticamente para las autenticaciones, sin intervención por parte del usuario.

El comando `ssh-add` busca los archivos de claves privadas, `id_rsa, id_dsa` y `identity`, en el directorio `.ssh` del directorio de conexión del usuario.

La opción `-l` del comando muestra las claves almacenadas por el agente SSH.

El agente puede utilizarse de manera remota, lo que permite administrar las claves privadas de autentificación solamente en un sitio, y poder usarlas desde diferentes máquinas clientes SSH.

Ejemplo

Activamos un shell gestionado por un agente SSH:

```
ssh-agent bash
```

Cargamos las claves:

```
ssh-add
Identity added: /home/pba/.ssh/id_rsa (pba@centos8)
```

El comando `ssh-add -l` muestra las claves contenidas dentro del agente:

```
ssh-add -l
3072 SHA256:dgH4ebogubKCiFarn9UTMsQuV51n4lXa5IkxMP7kS+s pba@centos8 (RSA)
```

3. Confidencialidad de las comunicaciones

SSH también se utiliza para establecer comunicaciones cifradas entre sistemas clientes y servidor, garantizando de esta manera la confidencialidad de distintas funcionalidades aplicativas.

a. Sesión interactiva con SSH

El comando `ssh` permite abrir una sesión interactiva desde un cliente SSH hacia un servidor SSH, proporcionando una cuenta de usuario válida para el sistema del servidor.

Sintaxis

```
ssh [NombreUsuario@]IdServidor
```

Donde:

`NombreUsuario`	Nombre de la cuenta de usuario remoto. Por defecto, cuenta de usuario local.
`Idservidor`	Nombre o dirección IP del servidor remoto.

Ejemplo

```
ssh 192.168.0.60
Bienvenido al servidor CentOS8.
Acceso reservado a personas autorizadas.
pba@192.168.0.60's password: XXXX
Bienvenido al servidor CentOS 8
Last login: Thu Jun 18 16:54:20 2020 from 192.168.0.70
[pba@centos8 ~]$
```

b. Copia de archivos

El comando `scp` se conecta al servidor SSH para solicitar la copia de archivos entre el sistema local y el sistema remoto. Usa los servicios de autenticación y de confidencialidad de SSH. La copia puede hacerse del cliente hacia el servidor o del servidor hacia el cliente.

Copia de archivo del cliente hacia el servidor con `scp`

```
scp ArchivoLoc [NombreUsuario@IdServidor]:ArchivoRemoto
```

Copia de archivo del servidor hacia el cliente con `scp`

```
scp [NombreUsuario@]IdServidor:ArchivoRemoto ArchivoLoc
```

Donde:

`NombreUsuario`	Nombre de la cuenta de usuario remoto. Por defecto, cuenta de usuario local.
`Idservidor`	Nombre o dirección IP del servidor remoto.
`ArchivoLoc`	Camino de acceso al archivo local.
`ArchivoRemoto`	Camino de acceso al archivo remoto.

c. Uso de aplicaciones en los túneles SSH

Un túnel SSH permite usar una conexión segura SSH para hacer que se comuniquen entre ellos un cliente y un servidor de otro protocolo aplicativo. Se establece desde el puesto cliente una conexión SSH hacia el servidor SSH. El servidor SSH redirige los datos recibidos hacia el servidor aplicativo de destino, y devuelve al cliente SSH los datos transmitidos por este servidor. El cliente aplicativo está conectado al cliente SSH que hace de intermediario.

El túnel SSH permite atravesar una red insegura, como Internet, en una comunicación cifrada, de manera transparente para el cliente y el servidor del protocolo aplicativo.

Sintaxis

```
ssh -L PuertoCli:IdDest:PuertoDest [usuario@]IdServidor
```

Donde:

`-L`	Modo túnel SSH.
`PuertoCli`	Puerto cliente que se tiene que redirigir hacia del túnel SSH.
`IdDest`	Dirección o nombre del servidor aplicativo de destino.
`PuertoDest`	Puerto del servidor aplicativo de destino.
`Usuario`	Cuenta de usuario del servidor SSH (por defecto, cuenta local).
`IdServidor`	Nombre o dirección IP del servidor SSH en el extremo del túnel.

d. Reenvío de sesiones X11 a través de SSH

El protocolo cliente/servidor `X`, usado por las aplicaciones gráficas en red, no es muy seguro. Sin embargo, podemos usar SSH para establecer una conexión segura entre el cliente y el servidor `X`.

El servidor SSH tiene que estar configurado para aceptar este tipo de tráfico, mediante su archivo de configuración `/etc/ssh/sshd_config`, con la directiva siguiente:

```
X11Forwarding yes
```

La conexión entre el cliente y el servidor SSH, que puede ser utilizada por las aplicaciones `X`, está activada con la opción `-X` del comando `ssh`:

```
ssh -X [usario@]IdServidor
```

Las aplicaciones gráficas pueden a continuación ser ejecutadas desde la sesión del cliente SSH.