Software de Comunicaciones

Práctica 7 - Secure Shell. SSH

Juan Díez-Yanguas Barber

Software de Comunicaciones

Ingeniería Informática - 5º Curso

Índice

1. Introducción

En esta práctica estudiaremos el protocolo SSH (Secure Shell) el cual sirve para acceder a otras máquinas a través de la red y manejarlas por completo a través del intérprete de órdenes de manera segura.

También permite el establecimiento de túneles seguros y proporciona transferencia de archivos segura usando SFTP o el comando SCP.

En esta práctica montaremos un servidor SSH en una máquina CentOS con el servidor OpenSSH.

2. Instalación del servidor OpenSSH

Tal como hemos hecho en ocasiones anteriores podemos instalar el servicio mediante un paquete rpm que podemos encontrar en el directorio que se proporciona con la máquina virtual que contiene varios paquetes rpm para instalar. Nos movemos al directorio /usr/local/src/openssh-server y usamos el comando rpm para llevar a cabo la instalación.

```
[root@server openssh-server]# rpm -ivh openssh-server-4.3p2-82.el5.i386.rpm
Preparando...                ########################################### [100%]
   1:openssh-server          ########################################### [100%]
[root@server openssh-server]# pwd
/usr/local/src/openssh-server
```

Ahora pasaremos a arrancar el servidor y probar que funciona correctamente. Lo normal en el proceso de instalación hubiera sido que generar las claves para la conexión, es posible que no lo haya hecho puesto que para realizar esta instalación primero se ha desinstalado para poder mostrar la captura.

```
[root@server openssh-server]# service sshd start
Iniciando sshd:                                            [  OK  ]
```

Podemos comprobar que funciona correctamente estableciendo una conexión desde el cliente y posteriormente veremos que ha sido configurado automáticamente en la instalación para ser un servicio que arranque automáticamente al inicio de la máquina en los niveles de ejecución del dos al cinco.

```
[root@server ~]# chkconfig --list |grep ssh
sshd            0:desactivado   1:desactivado   2:activo        3:activo        4:activo        5:activo        6:desactivado
```

3. Configuración de OpenSSH

En este apartado estudiaremos las directivas principales de configuración que podemos encontrar en el fichero de configuración de OpenSSH, este fichero lo podemos encontrar en el siguiente directorio /etc/ssh/sshd_config.

3.1. Configuración de direcciones y puertos de escucha

Al igual que ocurre en otros servicios podemos cambiar las direcciones y puertos de escucha del demonio de ssh. Lo haremos con la directiva Port y ListenAddress. Veamos un ejemplo con estos dos parámetros. Ponemos al servidor a escuchar en el puerto 2222 y solo en la dirección IP 172.16.183.100.

```
Port 2222
ListenAddress 172.16.183.100
```

Una vez que hemos reiniciado el servidor para aplicar los cambios pasamos a comprobarlo. No se ha perdido la conexión que estábamos usando para configurar porque el demonio que hay levantado atendiendo esas peticiones desde esta máquina no se ha cerrado, solo se ha cerrado el demonio principal de ssh, por tanto, mientras no cierre la conexión voy a poder seguir usándola.

```
[root@server sysconfig]# service sshd restart
Parando sshd:                                              [  OK  ]
Iniciando sshd:                                            [  OK  ]
```

Observamos que ya no atiende conexiones en el puerto 22.

```
Juan-DYB-MAC:~ JuanDYB$ ssh root@172.16.183.100
ssh: connect to host 172.16.183.100 port 22: Connection refused
```

Sin embargo si en la conexión especificamos el puerto si que conectará correctamente.

```
Juan-DYB-MAC:~ JuanDYB$ ssh root@172.16.183.100 -p 2222
root@172.16.183.100's password:
Last login: Thu May 16 19:13:16 2013 from 172.16.183.1
[root@server ~]#
```

Ahora comprobamos que la máquina con la que estamos trabajando tiene más interfaces de red configuradas pero no vamos a poder acceder por SSH a las mismas puesto que solo está escuchando en la dirección anterior.

```
[root@server ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:56:24:85:F2
          inet addr:172.16.183.100  Bcast:172.16.183.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10356 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5653 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9172460 (8.7 MiB)  TX bytes:440009 (429.6 KiB)
          Interrupt:59 Base address:0x2000

eth0:0    Link encap:Ethernet  HWaddr 00:50:56:24:85:F2
          inet addr:172.16.183.128  Bcast:172.16.183.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:59 Base address:0x2000
```

```
Juan-DYB-MAC:~ JuanDYB$ ping 172.16.183.128
PING 172.16.183.128 (172.16.183.128): 56 data bytes
64 bytes from 172.16.183.128: icmp_seq=0 ttl=64 time=0.449 ms
64 bytes from 172.16.183.128: icmp_seq=1 ttl=64 time=0.219 ms
64 bytes from 172.16.183.128: icmp_seq=2 ttl=64 time=0.400 ms
^C
--- 172.16.183.128 ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.219/0.356/0.449/0.099 ms
```

```
Juan-DYB-MAC:~ JuanDYB$ ssh root@172.16.183.128
ssh: connect to host 172.16.183.128 port 22: Connection refused
Juan-DYB-MAC:~ JuanDYB$ ssh root@172.16.183.128 -p 2222
ssh: connect to host 172.16.183.128 port 2222: Connection refused
```

3.2. Especificar la versión del protocolo SSH

Otro parámetro que podemos considerar como básico sería la directiva Protocol que podemos ver en el fichero de configuración que indicará con la versión del protocolo SSH con el que se va trabajar.

```
Protocol 2
```

3.3. Parámetros de acceso para control de usuarios

Es posible establecer varias directivas para indicar los usuarios a los que se permite el acceso vía SSH o no. En primer lugar hay una directiva especial para el usuario root. Se puede permitir o no el acceso SSH a root. Lo haremos con la directiva PermitRootLogin.

```
PermitRootLogin yes
```

Si cambiamos este parámetro al valor no podemos ver que ya no se permite el acceso a root.

```
PermitRootLogin no
```

```
Juan-DYB-MAC:~ JuanDYB$ ssh root@172.16.183.100
root@172.16.183.100's password:
Permission denied, please try again.
```

Sin embargo si que se permitirá el acceso a otros usuarios de la máquina, lo que no impide como se ve en la captura que pueda después escalar privilegios de root una vez que me he conectado con un usuario no privilegiado.

```
Juan-DYB-MAC:~ JuanDYB$ ssh juan@172.16.183.100
juan@172.16.183.100's password:
Last login: Mon May  6 07:46:40 2013 from 172.16.183.1
[juan@server ~]$ su
Contraseña:
```

Con el parámetro AllowUsers puedo especificar los usuarios que se pueden conectar por SSH, en este caso doy permiso a juan y a user1. Hay que tener en cuenta que aunque la directiva anterior este activada para permitir el acceso a root sino se incluye en esta lista no puede acceder.

```
AllowUsers juan user1
```

Comprobamos que con user2 no puedo acceder y sin embargo con el usuario juan si que se puede acceder.

También se puede usar la directiva de la siguiente manera y así se restringe más todavía el acceso

```
AllowUsers juan@172.16.183.1 user1@172.16.183.128
```

De esta manera se indica que el usuario juan solo se puede conectar desde esa IP.

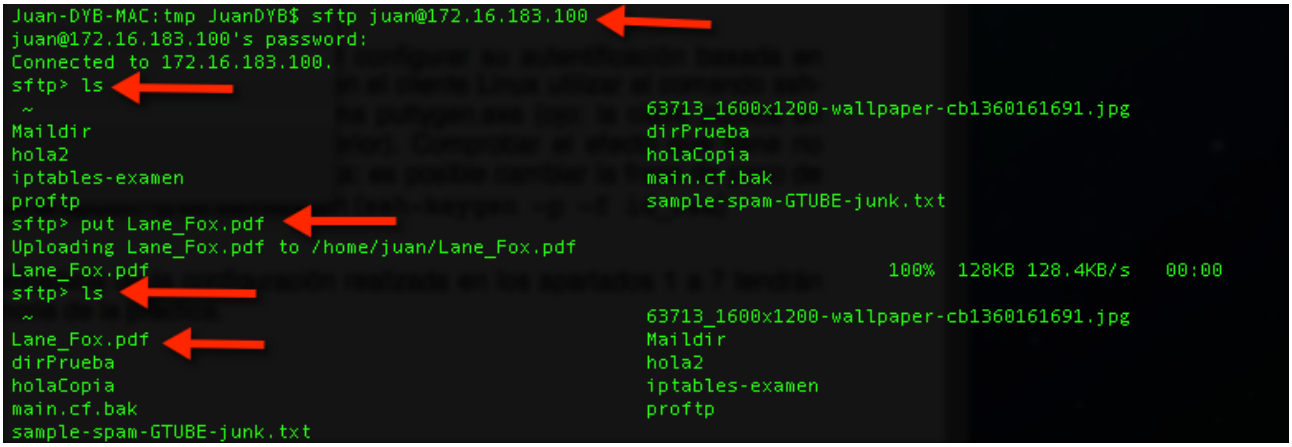
En todos estos cambios que hemos realizado no hemos indicado que haya que reiniar el servidor, pero ha de quedar claro que siempre que hagamos un cambio en el fichero de configuración hay que reiniciar el servidor para que surtan efecto los cambios.

4. Copia de archivos con SSH

Como ya se adelantó SSH proporciona además de la ejecución segura de comandos en remoto proporciona también copia de archivos. Hay dos formas, una de ellas es la simulación de FTP sobre un canal seguro SSH y otra de ellas es con el comando SCP.

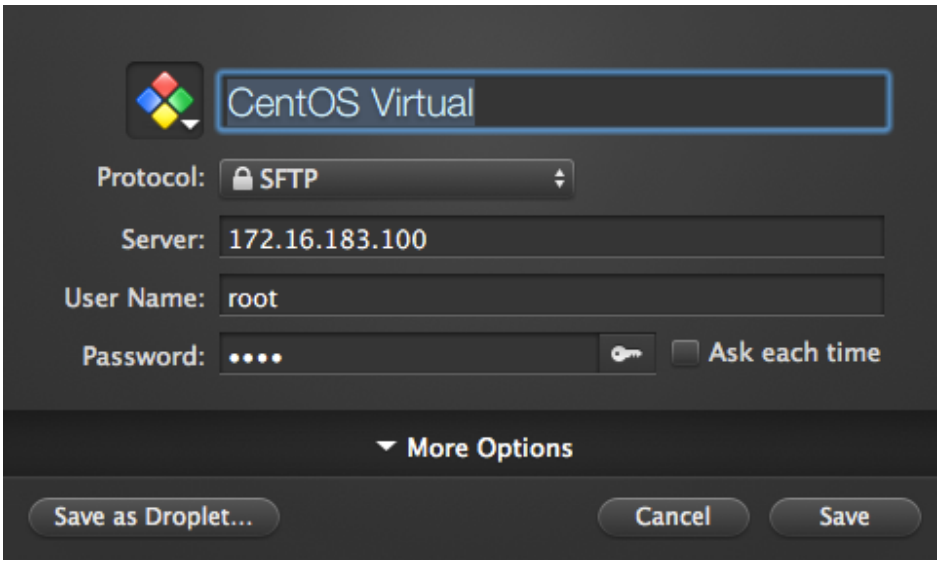
4.1. Uso de SFTP para copia de archivos

Podemos usar el comando sftp para conectarnos desde la máquina anfitriona y simular una sesión de FTP pero esta vez de manera segura.



Hemos usado los mismos comandos que en FTP, lo que hemos hecho ha sido listar el contenido del directorio /home/juan (esta vez no estamos chrooteados) y hemos subido un archivo pdf que teníamos en nuestra máquina anfitriona.

También podemos conectarnos por SFTP usando un cliente gráfico.



Con esta configuración podemos conectarnos vía SFTP con Transmit, un cliente FTP de Mac y podremos realizar las mismas acciones que hacíamos con una conexión FTP pero esta vez sobre un canal seguro.

4.2. Uso de SCP para copia de archivos

Como hemos comentado también podemos usar SCP para la copia de archivos.

```
Juan-DYB-MAC:tmp JuanDYB$ scp Lane_Fox.pdf juan@172.16.183.100:~/
juan@172.16.183.100's password:
Lane_Fox.pdf                                  100%  128KB 128.4KB/s   00:00
```

Como vemos hemo ejecutado el comando desde el cliente sin haber iniciado sesión SSH, primero inicia la sesión y por ello nos pide la contraseña y después envía el archivo.

El comando tiene varios parámetros posibles.

- -p: Preserva el tiempo de modificación, tiempos de acceso y los modos del archivo original.
- -P: Especifica el puerto para realizar la conexión.
- -r: Copia en modo descendente de los directorios especificados.

En lo que respecta a WinSCP no vamos a hacer uso del mismo ya que estamos trabajando con una máquina anfitriona en Mac y por tanto no podemos ejecutar dicho programa.

5. Túneles SSH desde el cliente

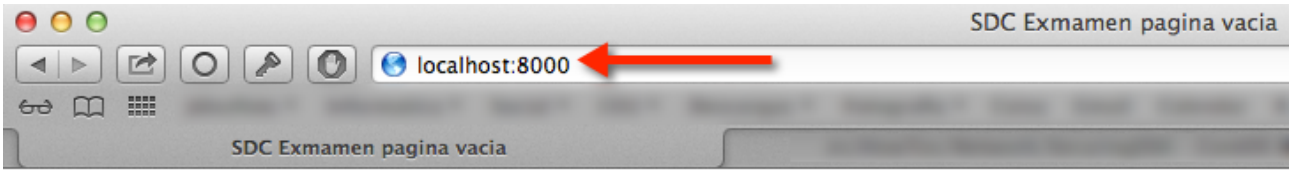
SSH también permite el establecimiento de túneles lo que permite securizar servicios que en un principio no están securizados como por ejemplo HTTP, POP3, etc ... También se suele usar esto para evitar cortafuegos y realizar conexiones al exterior por puertos no permitidos, ya que, si SSH está abierto podemos abrir un túnel para otros servicios que a lo mejor si están restringidos.

```
ssh -L puerto_local:host_remoto:puerto_remoto servidor
```

```
ssh -L root@172.16.183.100 8000:172.16.183.100:80
```

Con este comando lo que hacemos es abrir un túnel que empieza en el puerto 8000 de nuestra máquina anfitriona y acaba en el puerto 80 de la máquina remota. Veamos lo que pasa si ahora abrimos un navegador. Previo arranque del servidor Apache en la máquina remota por supuesto.

```
Juan-DYB-MAC:tmp JuanDYB$ ssh root@172.16.183.100 -L 8000:172.16.183.100:80
root@172.16.183.100's password:
Last login: Thu May 16 19:15:25 2013 from 172.16.183.1
```



Pagina vacia

Anotaciones

Este es el punto de entrada del servidor virtual vacio examen SDC

`/proyectos/vacio`

También se puede hacer un túnel que vaya desde el cliente a través del servidor SSH hacia un servicio que se encuentre en una máquina diferente a la remota. La dirección IP que hemos indicando el la dirección IP de google.

```
Juan-DYB-MAC:tmp JuanDYB$ ssh root@172.16.183.100 -L 8000:173.194.41.31:80
root@172.16.183.100's password:
Last login: Thu May 16 21:06:09 2013 from 172.16.183.1
[root@server ~]#
```

Por último se puede dejar el túnel abierto para que se puedan conectar a él desde otras direcciones, esto lo haremos con la opción -g.

```
Juan-DYB-MAC:tmp JuanDYB$ ssh root@172.16.183.100 -g -L 8000:172.16.183.100:80
root@172.16.183.100's password:
Last login: Thu May 16 21:24:59 2013 from 172.16.183.1
```

Lo comprobamos con una segunda máquina virtual, lo que haremos será conectarnos al puerto 8000 de la máquina anfitriona y tiene que aparecer la web que hay en el servidor apache de la máquina virtual presente en la dirección 172.16.183.100.

```
[root@server ~]# lynx 172.16.183.1:8000_
```

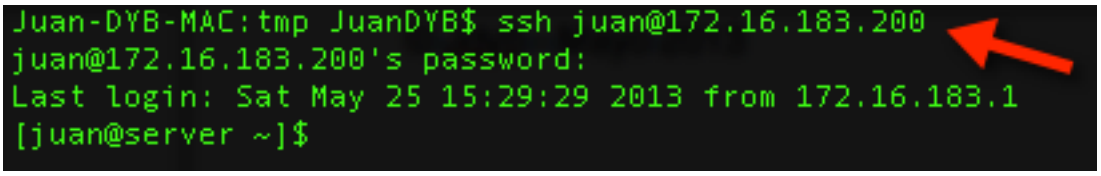
6. Autenticación basada en clave pública/privada

Hasta el momento nos hemos estado autenticando en el servidor mediante claves pero hay otra forma posible. Se puede hacer mediante claves asimétricas. Esto aumenta la seguridad de nuestro servidor.

Una vez activado no hace falta recordar contraseñas salvo que se haya fijado una contraseña para proteger la clave privada, y una vez habilitado esto se puede deshabilitar la autenticación con contraseña y ya no habrá más intentos de acceso al servidor mediante contraseña, solo puede entrar quien tenga una clave autorizada por el servidor.

Vamos a ejecutar los comandos desde una segunda máquina virtual que estará en la dirección 172.16.183.200 y en ella crearemos un usuario juan para que acceda al servidor ssh de la máquina virtual principal.

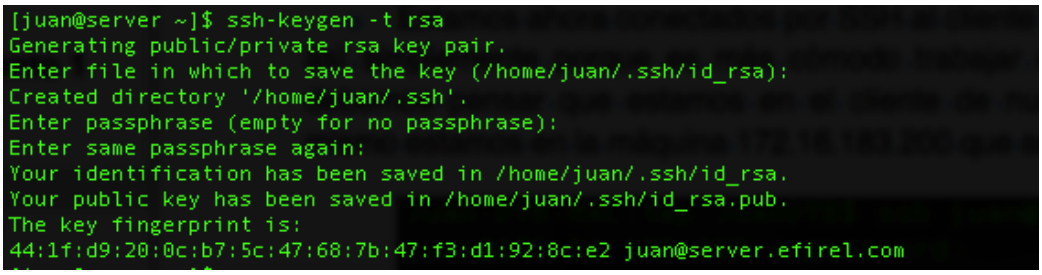
Estamos ahora conectados por SSH al cliente de nuestro servidor SSH lo hacemos así simplemente porque es más cómodo trabajar con SSH que con el terminar, pero debemos pensar que estamos en el cliente de nuestro servidor SSH. Es decir ahora mismo estamos en la máquina 172.16.183.200 que se va a comportar como cliente SSH.



Lo primero que haremos será crear un par de claves públicas y privadas en el cliente que serán usadas para conectarse con el servidor. Eso lógicamente se tendrá que hacer por cada máquina cliente que quiera conectarse al servidor.

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

No hemos puesto clave en la clave privada, lo hemos hecho dejándolo vacío tal y como indicaba el asistente. Esto lo que habrá hecho será guardar las claves en un directorio oculto en el home del usuario ~/.ssh.



En el directorio indicado podemos observar la clave privada y la pública.

```
[juan@server ~]$ ls -la .ssh/
total 16
drwx------ 2 juan juan 4096 may 25 15:34 .
drwx------ 3 juan juan 4096 may 25 15:34 ..
-rw------- 1 juan juan 1675 may 25 15:34 id_rsa
-rw-r--r-- 1 juan juan  404 may 25 15:34 id_rsa.pub
```

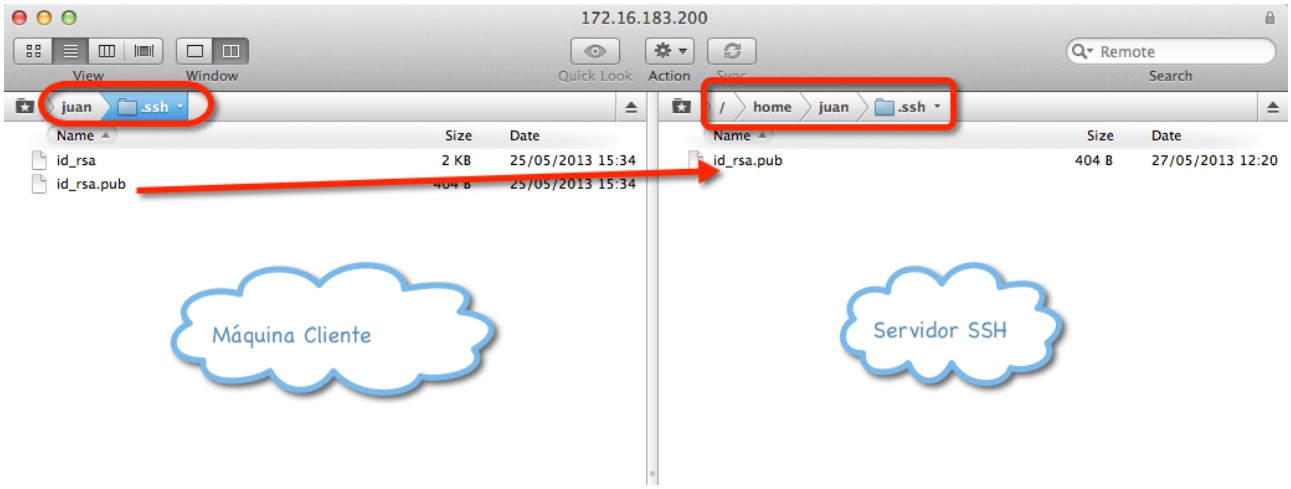
Hay que tener claro que aunque en este ejemplo no se haya protegido la clave privada con contraseña ha sido por el mero hecho de ser un ejemplo y no estar en producción pero se debería de poner porque si se logra entrar en la máquina del cliente también se tendría acceso a la máquina remota y por otra parte el usuario root también podrá ver la clave, si el usuario root no es confiable o está comprometido deberíamos proteger la clave privada con contraseña.

Ahora cambiaremos los permisos de la clave privada para asegurar que solo la va a usar el usuario al que pertenece.

```
[juan@server ~]$ chmod 700 .ssh/
[juan@server ~]$ chmod 600 .ssh/id_rsa
```

Ahora llevaremos la clave pública al servidor. Copio la clave pública del cliente al usuario que me quiero conectar de la máquina servidor por SSH (que se llamen igual no tiene nada que ver). Llevo a cabo la copia mediante SFTP con el cliente gráfico.

Llevo la clave de /home/juan/.ssh/id_rsa.pub en el cliente al servidor en el directorio del usuario con el que me quiero autenticar en el directorio /home/juan/.ssh/id_rsa.pub.



Ahora hemos de crear un fichero llamado authorized_keys a partir del fichero que hemos copiado. En ese fichero se irán añadiendo las claves permitidas para el usuario

juan en el servidor. Una vez hecho esto si queremos podemos borrar el fichero que hemos copiado.

```
[juan@server ~]$ cat .ssh/id_rsa.pub > .ssh/authorized_keys
```

```
[juan@server ~]$ chmod 700 .ssh/
[juan@server ~]$ chmod 600 .ssh/authorized_keys
[juan@server ~]$ ls -la .ssh/
total 16
drwx------ 2 juan juan 4096 may 16 22:33 .
drwx------ 7 juan juan 4096 may 16 21:59 ..
-rw------- 1 juan juan  404 may 16 22:33 authorized_keys
-rw-r--r-- 1 juan juan  404 may 16 22:00 id_rsa.pub
```

Lo que hemos hecho del cambio de permisos realmente solo es necesario si la opción StrictModes del fichero de configuración de SSH está con el valor yes asignado.

Con estos pasos realizados lo primero que se intentará es la autenticación con claves pública y privada y si este método falla pedirá la contraseña convencional.

Hemos usado estos directorios tan específicos porque son en los que busca las claves por defecto SSH de hecho están en el fichero de configuración comentados para indicar la orden. Luego sino se indica nada son los directorios por defecto.

```
#StrictModes yes
MaxAuthTries 2

#RSAAuthentication yes
#PubkeyAuthentication yes
#AuthorizedKeysFile     .ssh/authorized_keys
```

Vamos a intentar entrar desde nuestro cliente al servidor usando el usuario juan.

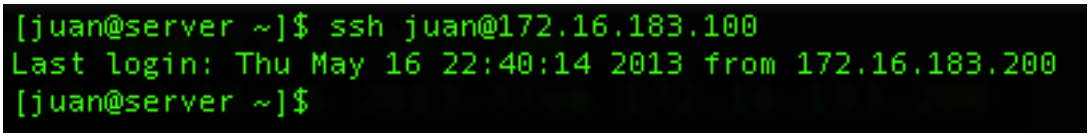
```
[juan@server ~]$ ssh juan@172.16.183.100
Last login: Thu May 16 22:16:36 2013 from 172.16.183.200
[juan@server ~]$
```

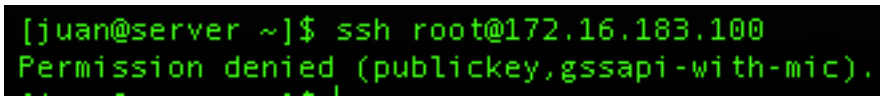
Se puede ver que no se ha pedido contraseña alguna. Podemos deshabilitar la autenticación con contraseña y veremos que sigue funcionando.

```
PasswordAuthentication no
```

```
[root@server juan]# service sshd restart
Parando sshd:                                              [  OK  ]
Iniciando sshd:                                            [  OK  ]
```

Ahora comprobamos que sigue funcionando la entrada con el usuario juan pero sin embargo root no podrá entrar ya que no hay claves para el usuario root por tanto sin autenticación con contraseña solo puede entrar el usuario juan.





Si se desea se puede cambiar la frase de paso para la clave privada en el cliente con el siguiente comando.

```
ssh-keygen –p –f id_rsa
```

Con los pasos realizados como se ha podido comprobar se puede tener un servidor SSH y autenticarse ante el con pares de claves en vez por el método con contraseña tradicional.