

OpenSSH

1. Presentación

OpenSSH es un protocolo de shell con prestaciones de seguridad, un mecanismo que permite la autenticación segura, la ejecución remota y la conexión a distancia. Permite también el transporte seguro del protocolo X Window. En realidad, es capaz encapsular protocolos no seguros redireccionando los puertos.

Los paquetes que se deben utilizar para un servidor son **openssh**, **openssl** y **openssh-clients**. Para X se añaden los paquetes **openssh-askpass*** (puede haber varios en función del entorno de escritorio). La lista de los paquetes que hay que instalar depende de cada distribución.

El uso más común sigue siendo el acceso remoto seguro a una máquina mediante el cliente ssh.

2. Configuración

La configuración es `/etc/ssh/sshd_config`. Si es preciso, se pueden modificar algunas opciones:

- ˘ **Port**: el número de puerto por defecto es 22;
- ˘ **Protocol**: fijado en 2,1, autoriza SSH1 y SSH2. Se preferirá SSH2 y así se dejará el valor 2;
- ˘ **ListenAddress**: por defecto ssh escucha en todos los IP del servidor. Se puede autorizar únicamente la escucha en una interfaz dada;
- ˘ **PermitRootLogin**: **ssh** autoriza las conexiones de root. Se puede poner el valor a **"no"**. En este caso, habrá que conectarse como simple usuario y pasar por **su** o **sudo**;
- ˘ **Banner**: ruta de un archivo cuyo contenido se mostrará a los usuarios durante la conexión.

ssh es un servicio System V que puede iniciarse con service o directamente por /etc/init.d/sshd, o una unidad systemd.

```
# service sshd start
```

3. Utilización

El comando **ssh** permite establecer una conexión.

```
$ ssh -l login host  
$ ssh login@host
```

La opción **-X** permite activar la redirección (forwarding) del protocolo X Window.

```
$ ssh -X login@host
```

4. Claves y conexión automática

Es posible establecer una conexión automática hacia otra máquina sin introducir una contraseña. Para ello, es necesario generar un par de claves, privada y pública, desde la cuenta de usuario del cliente (la máquina que se va a conectar). No se debe introducir ninguna frase de contraseña, en el caso de que la clave privada haya sido protegida con una contraseña, para no tener que teclearla, esta tendrá que haber sido guardada en la colección de claves de sesión o a través de ssh-agent.

Por parte del servidor ssh, se debe colocar la clave pública del cliente en un archivo que contiene las claves autorizadas para conectarse en la cuenta de destino.

a. Tipos de cifrado

SSH soporta cuatro tipos de claves de cifrado:

- ˆ **DSA**, *Digital Signature Algorithm*, ya no se considera como segura y no debe ser utilizada.
- ˆ **RSA**, *Rivest, Shamir and Adleman*, se trata de los nombres de sus creadores, es un algoritmo asimétrico, el cual todavía no ha sido roto, y cuyas posibilidades de ataque solamente están determinadas a la potencia de cálculo de las máquinas. Al final de 2019, los métodos y las máquinas más potentes solo podían romper, teóricamente, una clave de 795 bits, lo que puede hacernos pensar que una clave de 1024 bits podría (por el momento esta suposición es teórica) romperse en algunos años. Una clave de 2048 o 4096 bits es, de momento, virtualmente irrompible.
- ˆ **ECDSA**, *Elliptic Curve Digital Signature Algorithm*, es, particularmente, el formato elegido por Sony para firmar los productos de la PS3. Esta llave ha sido rota, y no por el propio algoritmo, sino por un uso pésimo con la implementación de un valor estático que ha permitido resolver la ecuación. El fallo viene sistemáticamente de la mala implementación del algoritmo, pero ese cifrado es seguro cuando está bien implementado.
- ˆ **Ed25519**, clave de tipo EdDSA, *Edwards Curve Digital Signature Algorithm*, es a la vez el algoritmo más rápido, más seguro y el que mejor rendimiento presenta, además su implementación es pública. El único problema es la ausencia de compatibilidad entre productos: este algoritmo no está soportado de manera masiva.

Si no tiene ningún problema de compatibilidad, use el algoritmo de cifrado ed25519. En caso contrario, RSA será el mejor compromiso.

Cada clave posee una talla, expresada en bits. Cuanto más grande es la talla, más segura se considera la clave. Para una clave RSA, elija un tamaño de 4096 bits, es un buen compromiso entre la seguridad y el rendimiento. Compruebe, sin embargo, si los sistemas con los que se está comunicando admiten el tamaño que ha elegido.

b. Lado cliente



Genere una clave en formato RSA con el comando **ssh-keygen**:

```
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/bean/.ssh/id_rsa):
Created directory '/home/bean/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in/home/bean/.ssh/id_rsa.
Your public key has been saved in/home/bean/.ssh/id_rsa.pub.
The key fingerprint is:
f6:39:23:4e:fa:53:d0:4e:65:7f:3f:fd:a3:f4:8e:2a bean@server
```

→ El directorio del usuario contiene ahora un directorio .ssh:

```
$ cd .ssh
$ ls
id_rsa id_rsa.pub
```

→ El archivo id_rsa.pub contiene la clave pública:

```
$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAkB/VskR9v708J2EDG1LM1Q6HmKJc
P2UenurnSr7rWTSZK5w9Hzn4DCz5iMzLAPc4659I0uKJbmF3vBXozlgLrCdCZCQE
hhPLwJVLXbGNc8IMf742E/WqkkJ/uQYb31iPAU7Efosei+DVZ21No725XjiSCZ2q
zKKx7ZuNQEtXW0eVkwvIA0u7Hvrwn+FQksW3NXwTxwHhudSw7S6kIC3tyF5rkzfk
vu7zQbOGDGGPiF3aOvd0oSBNGiJtZ+M0PaoXXI3brMd66WkGfSwf4ofYKNDCA/3T
Q4xU6WxkxqTBcsjEm1glymFAyxDo+zzf63jxLGO8Pp50DKf7DUqBx7+rjw==
bean@server
```

c. Lado servidor

→ Vaya al directorio .ssh de la cuenta a la cual desea acceder en el servidor (cree una si no existe):

```
$ cd /home/seb/.ssh
```

- Edite el archivo `authorized_keys2` (créelo si no existe) y copie dentro en una nueva línea el contenido del archivo `id_rsa.pub` del cliente. Guarde.

```
$ echo "ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAkB/VskR9v708J2EDG1LM1Q6HmKJcP2Uenurn
Sr7rWTSZK5w9Hzn4DCz5iMzLAPc4659I0uKJbmF3vBXozlgLrCdCZCQEHhPLwJVL
XbGNc8IMf742E/WqkkJ/uQYb31iPAU7Efosei+DVZ21No725XjiSCZ2qzKKx7ZuN
QEtXW0eVkwvIA0u7Hvrwn+FQksW3NXwTxwHhudSw7S6kIC3tyF5rkzfkvu7zQbOG
DGGPiF3aOvd0oSBNgiJtZ+M0PaoXXI3brMd66WkGfSwf4ofYKNDCA/3TQ4xU6Wxk
xqTBcsjEm1glymFAyxDo+zzf63jxLG08Pp50DKf7DUqBx7+rjw== bean@slyser
ver" >> authorized_keys2
```

- Intente una conexión; no se requiere la contraseña:

```
$ ssh seb@slyserver
```

d. Copia automática

Si bien el método anterior es didáctico, no es nada práctico. Utilice siempre el comando **ssh-copy-id**:


```
$ ssh-copy-id ~/.ssh/id_rsa.pub seb@slyserver
```

Se solicita la contraseña de la cuenta seb (la primera vez) luego la clave se copia en el archivo de claves autorizadas del host remoto.

5. Passphrase y agente SSH

Además de la seguridad de la clave SSH y el cifrado del flujo de datos, debe considerar el uso de una frase de paso, que se solicita durante la conexión SSH. Si su clave pública no es reconocida por el host, se solicitará la frase y contraseña.

Si olvidó a introducir una contraseña al crear la clave, o desea cambiarla, utilice la opción

 de ssh-keygen:

```
$ ssh-keygen -p
Enter file in which the key is (/home/seb/.ssh/id_rsa):
Key has comment '/home/seb/.ssh/id_rsa'
Enter new passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved with the new passphrase.
```

La introducción de su frase con cada conexión SSH puede convertirse rápidamente en un fastidio. Muchos SO o entornos de escritorio ofrecen almacenarla en sus herramientas de seguridad mientras dure la sesión o de forma permanente en coordinación con el comando **ssh-agent**.

ssh-agent es un servicio que almacenará sus claves privadas y las frases correspondientes durante la sesión, o un tiempo delimitado a escoger. Generalmente lo arranca la sesión gráfica, pero también se puede iniciar de forma manual:

```
$ ssh-agent
SSH_AUTH_SOCK=/tmp/ssh-q0vj5zBtrC1Y/agent.2118; export SSH_AUTH_SOCK;
SSH_AGENT_PID=2119; export SSH_AGENT_PID;
echo Agent pid 2119;
```

Seleccione las últimas líneas. Haga un copiar y pegar y ejecútelas (sobre todo la parte SSH_AUTH_SOCK).

```
$ SSH_AUTH_SOCK=/tmp/ssh-q0vj5zBtrC1Y/agent.2118; export SSH_AUTH_SOCK;
```

Ahora puede iniciar el comando ssh-add para añadir su passphrase:

```
$ ssh-add
Enter passphrase for /home/seb/.ssh/id_rsa:
Identity added: /home/seb/.ssh/id_rsa (/home/seb/.ssh/id_rsa)
```

La passphrase no volverá a ser solicitada durante la duración de la sesión.

Si desea eliminar una identidad antes del fin de la sesión, por ejemplo si se bloquea antes de desconectar, emplee `-d`:

```
$ ssh-add -d
Identity removed: /Users/seb/.ssh/id_rsa ( seb@linux-h7e0.site)
```

6. Autenticación del servidor

Cuando deseamos conectarnos en SSH, la primera etapa consiste en conectarse a la máquina remota, antes incluso de intentar conectarse a la cuenta de usuario. Esta conexión presenta un intercambio de claves. Por defecto, la máquina autoriza a los clientes para que se conecten y el cliente guarda la firma de la máquina remota en el archivo **.ssh/known_hosts**:

```
# cat .ssh/known_hosts
192.168.0.49 ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBIMpDhfpUte9eWTUuCE5
blAVZetEyFZBQXYLF6S+dvakUJ1qI4pJrjuGSAmzC8gbJrTZfQFQLZM4sCzPIjtjVU=
192.168.0.10 ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBBTP4P2jgsuahc+OGF2y
DAGlvD+tyUODY1MAUG2h1keXFngK+fwssdoAfVn1NpYG38XAEAbpTU1ivGZXCxcWVGM=
192.168.0.35 ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBN1n2TX4Vf5twyhNoyGa
Yg9y/kz7ueRZbL/+iSPT8u0fUlb4xbqGtCxOq3OYOgnWuU2fRnoMxJMNjEdozAWesz4=
```

Si tuviera acceso al servidor 192.168.0.10, podría comprobar el contenido del directorio `/etc/ssh`:

```
$ ls -l /etc/ssh
total 568
-rw-r--r-- 1 root root 535195 mar  9 15:17 moduli
-rw-r--r-- 1 root root 1603 may 29 2020 ssh_config
drwxr-xr-x 2 root root 4096 may 29 2020 ssh_config.d
-rw-r--r-- 1 root root 3289 mar  9 15:17 sshd_config
```

```

drwxr-xr-x 2 root root 4096 mar  9 15:17 sshd_config.d
-rw----- 1 root root 505 may  8 10:21 ssh_host_ecdsa_key
-rw-r--r-- 1 root root 173 may  8 10:21 ssh_host_ecdsa_key.pub
-rw----- 1 root root 399 may  8 10:21 ssh_host_ed25519_key
-rw-r--r-- 1 root root 93 may  8 10:21 ssh_host_ed25519_key.pub
-rw----- 1 root root 2602 may  8 10:21 ssh_host_rsa_key
-rw-r--r-- 1 root root 565 may  8 10:21 ssh_host_rsa_key.pub
-rw-r--r-- 1 root root 342 may  8 10:21 ssh_import_id
$ cat ssh_host_ecdsa_key.pub
ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBBTP4P2jgsuahc+OGF2yD
AGlvD+tyUODY1MAUG2h1keXFngK+fwssdoAfVn1NpYG38XAEAbpTU1ivGZXCxcWVGM=

```

Como puede comprobar, así como cualquier cliente puede generar distintos tipos de claves, el servidor también puede hacerlo. El nombre de los archivos es bastante significativo para comprender cuál es el tipo de clave generada, con su clave pública.

Cuando nos desconectamos del servidor, cliente y servidor han negociado una clave ecdsa, las claves públicas se corresponden. Si la clave del servidor cambiara, el cliente no tendría ya la clave correcta y el cliente ssh mostraría un error de seguridad con un riesgo potencial: es posible que el servidor no sea el que pretende ser.

```

$ ssh seb@192.168.0.10
The authenticity of host '192.168.0.10 (192.168.0.10)' can't be established.
ECDSA key fingerprint is SHA256:1WC055PSsQCqRY5z0VmxqtLFcTYGcB4u8CB/Jfzh2IQ.

```