Capítulo 22

SSH

SSH (Secure SHell) es un protocolo de aplicación que permite establecer conexiones con máquinas remotas. Podríamos decir que es una versión mejorada del veterano[1] `telnet`, sin embargo, es mucho más que eso. SSH proporciona un canal seguro (cifrado) para ejecutar comandos, transferir archivos[2], redireccionar conexiones, crear túneles que pueden transportar datos de conexiones arbitrarias o simplemente como sistema de autenticación.

22.1. Shell segura

Como ya indica su nombre, el uso más común de la aplicación SSH es el de «shell remota», es decir, un programa que permite abrir una sesión en un computador remoto y ejecutar programas. En este documento vamos a usar la aplicación `OpenSSH`, que en el caso de sistemas GNU/Linux suele estar disponible como dos paquetes: `openssh-server` y `openssh-client`. Por supuesto, existen otras muchas implementaciones para casi cualquier SO.

Se trata, por supuesto, de una aplicación cliente-servidor: El servidor SSH, que debe estar instalado en cada computador que queramos que sea accesible, siempre está a la espera de conexiones, mientras que el cliente SSH es el que inicia la conexión.

Para probar SSH vamos a utilizar un contenedor docker. Para construir la imagen docker correspondiente, descarga el repositorio `git` con el siguiente comando:

```
alice@maine:~$ git clone https://github.com/destripando-internet/code
```

Y ejecuta el siguiente comando dentro del directorio `ssh-docker`. Esto crea la imagen y arranca el contenedor que ejecuta el servidor `SSH`:

[1] ...e inseguro
[2] con `scp`

```
alice@maine:~$:~/net-book-code/ssh-docker$ docker-compose up -d
[...]
```

En un sistema GNU/Linux utilizar SSH es muy sencillo. Basta con abrir un terminal e indicar a través del comando `ssh` el nombre de usuario, el computador remoto y el puerto (si es distinto del 22).

```
alice@maine:~$ ssh user@172.20.0.2
user@172.20.0.2's password:
user@viper:~$
```

El comando `ssh` trata de establecer una conexión segura con el servidor SSH que hay en 172.20.0.2 (el contenedor docker). Concretamente queremos acceder con la cuenta del usuario `user`. Este proceso por defecto pide la contraseña del sistema asociada a ese usuario, que no se muestra al introducirla. Es nuestro caso es **secret**.

22.2. Configuración

Para simplificar el comando de conexión se puede escribir un archivo de configuración llamado `\textasciitilde /.ssh/config` en el **computador cliente**. Para el caso anterior, el archivo podría ser algo como:

```
Host viper
    hostname 172.20.0.2
    User user
```

Es importante señalar que este archivo debe tener permisos de lectura/escritura solo para el usuario (`-rw-r--r--`) o será ignorado. Una vez tengas esta configuración puedes conectar simplemente con:

```
alice@maine:~$ ssh viper
user@172.20.0.2's password:
```

Este modo de autenticación con usuario/clave se llama de «clave privada» porque se asume que solamente el usuario conoce dicha clave.

22.3. Acceso con clave pública

Para evitar tener que escribir constantemente la clave cada vez que conectas a una misma máquina, SSH ofrece un sistema de autenticación de «clave pública». Este sistema utiliza un par de claves privada/pública. La clave pública se coloca en el sistema al que quieres acceder y, de hecho, no hay ningún problema en que cualquiera la pueda leer o copiar (por eso es pública). Sin embargo, la clave privada debe quedar custodiada por el usuario y nunca debe ser accesible para nadie más.

Para generar el par de claves ejecuta el siguiente comando aceptando todos los valores por defecto (pulsa ENTER):

```
alice@maine:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ana/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ana/.ssh/id_rsa
Your public key has been saved in /home/ana/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:KXUrUqPxk/EPiXEBfgqK3HlSqvhphSeh6O+qm9tz1DM alice@maine
```

Esto genera dos archivos llamados `id\_rsa` (clave privada) y `id\_rsa.pub` (clave pública) en el directorio `/home/ana/.ssh/`.

Ahora se debe enviar la clave pública al servidor. Esto se puede hacer fácilmente con:

```
alice@maine:~$ ssh-copy-id viper
user@172.20.0.2's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'viper'"
and check to make sure that only the key(s) you cantead were added.
```

Este programa almacena la clave pública del usuario —por defecto, `\textas-ciitilde /.ssh/id\_rsa.pub`— en el archivo `\textasciitilde /.ssh/authorized\_keys` del servidor (`viper` en este ejemplo).

Y tal como indica, ahora deberíamos poder entrar en la máquina `viper` simplemente con:

```
alice@maine:~$ ssh viper
Last login: Mon Mar 14 21:07:15 2022 from 172.20.0.1
user@viper:~$
```

Esto no solo es interesante porque ahorra al usuario tener que escribir su contraseña constantemente. Además permite que las aplicaciones puedan automatizar tareas sin la intervención de un usuario. Los sistemas de control de versiones (como `git`) o sistemas de Integración o Despliegue Continuo (CI/CD) utilizan habitualmente autenticación SSH con clave pública.

También es posible ejecutar un único comando en el servidor y ver la salida directamente en el computador cliente. Simplemente hay que escribir el comando a continuación:

```
alice@maine:~$ ssh viper ls /home/
user
```

22.4. Autenticación con certificado

Otra forma de autenticar un usuario es mediante certificados, una opción algo más avanzada que la autenticación con clave pública, que hemos visto en la sección anterior. Estos certificados son emitidos por una autoridad de certificación o CA (Certificate Authority), que firma (con su clave privada) el par de claves (privada/pública) de los usuarios. El resultado son certificados que permiten autenticar usuarios en hosts y viceversa.

La ventaja de este sistema es que el administrador de un servidor puede expedir certificados para un número arbitrario de usuarios. Un mismo par de claves firmado por un administrador puede otorgar derecho de acceso a múltiples servidores sin tener que modificar su configuración.

Las claves de la CA se crean también con el comando `ssh-keygen`:

```
$ ssh-keygen -t rsa -f ca_key -P "" -C ca_key
Generating public/private rsa key pair.
Your identification has been saved in ca_key
Your public key has been saved in ca_key.pub
The key fingerprint is:
SHA256:nbN+pdyB3paUUnoPe60KJbkC3flKiXCVbSIH51wQXbo ca_key
```

Aunque `ssh-keygen` proporciona multitud de opciones, en este caso solamente utilizamos algunas de ellas:

`-t` permite indicar el algoritmo (en este caso cifrado RSA).

`-f` especifica el nombre del archivo destino.

`-P` define la contraseña (vacía en este caso).

`-C` añade un comentario que aparece al final de la clave pública.

La clave privada (`ca\_key`), por supuesto, debe almacenarse de forma segura, ya que es con la que se firma y emite los certificados. La clave pública (`ca\_key.pub`) es la que se utiliza el servidor SSH en el proceso de autenticación de los clientes.

22.4.1. Un certificado para el usuario

Los certificados SSH de usuario hacen posible autenticar usuarios en un servidor determinado. Para ilustrar cómo funcionan vamos a generar un certificado para el usuario `user` de la máquina `viper` como lo haría su administrador. Primero creamos un par de claves pública/privada para el usuario, con el nombre `user\_key`.

```
$ ssh-keygen -f user_key
```

Después, se firma la clave pública del usuario (`user\_key.pub`) con la clave privada de la CA (`ca\_key`):

```
$ ssh-keygen -s ca_key -I viper -n user user_key.pub
```

Con este comando se ha generado también el certificado `user\_key-cert.pub`.

Las opciones que se han usado en la firma son:

`-s` para indicar la clave con la que se quiere firmar (*sign*).

`-I` para determinar la identidad del certificado, normalmente el nombre del *host* (puede utilizarse en el futuro para revocar un certificado, por ejemplo).

`-n` que permite definir una lista de usuarios y/o *hosts* donde el certificado es válido (en este caso únicamente el usuario `user`).

En el servidor, es necesario almacenar la clave pública de la CA con los permisos adecuados (644). En la máquina `viper`:

```
root@viper:/etc/ssh# chmod 644 ca_key.pub
root@viper:/etc/ssh# ls -la ca_key.pub
-rw-r--r-- 1 user user    562 Mar  3 09:13 ca_key.pub
```

Para indicar que todas las claves de usuario firmadas por la CA se pueden autenticar en el servidor, se debe añadir lo siguiente en el archivo de configuración `/etc/ssh/sshd\_config`:

```
TrustedUserCAKeys /etc/ssh/ca_key.pub
```

Para que los cambios tengan efecto reiniciamos el servidor SSH (`sshd`) con:

```
root@viper:/home/user# systemctl restart sshd
```

Con esto termina la configuración el servidor (`viper` en este ejemplo). Este proceso lo realizar automáticamente la construcción de la imagen docker y se puede ver en el archivo `Dockerfile`.

Por otro lado, el cliente tiene que copiar su clave privada (`user\_key`) y su certificado (`user\_key-cert.pub`) en su directorio `\textasciitilde /.ssh/`:

```
alice@maine:~$ ls -l ~/.ssh
-rw------- 1 ana    ana    2590 mar  3 09:35 user_key
-rw-rw-r-- 1 ana    ana    2020 mar  3 09:40 user_key-cert.pub
-rw-rw-r-- 1 ana    ana     137 mar  3 10:04 config
```

Además, habrá de indicar que quiere usar esa clave a la hora de acceder a la máquina `viper`:

```
alice@maine:~$ ssh -i ~/.ssh/user_key viper
```

Para mayor comodidad, puede añadirse la clave a la configuración de `viper` en `\textasciitilde /.ssh/config`:

```
Host viper
    Hostname localhost
    Port 2200
    User user
    IdentityFile ~/.ssh/user_key
```

Y con esto se podrá acceder simplemente con:

```
alice@maine:~$ ssh viper
```

La diferencia respecto a lo explicado en la sección 22.3 es que el usuario nunca necesitó una clave textual para acceder al servidor. De hecho, el servidor se puede configurar para que la única forma de autenticación permitida sea mediante clave pública [3], de modo que no hay posibilidad de usar `ssh-copy-id`. Esto se considera más seguro.

22.5. Copia de archivos con SCP

SCP (Secure Copy Protocol) es un protocolo de copia segura de archivos entre computadores cualesquiera conectados a Internet. Resulta especialmente cómodo ya que el propio servidor de `OpenSSH` lo incorpora y está activado por defecto.

Con la configuración previa es tan sencillo como:

```
$ scp un-\file viper:
un-\file                                      100%    5    16.6KB/s   00:00
```

También es posible copiar directorios si se utiliza la opción `-r`.

[3] `PasswordAuthentication no`