11. ACCESO REMOTO SEGURO A UN SISTEMA

11.1.- Introducción

Cuando se administra un sistema, lo habitual es que el sistema no esté en la misma ubicación física que el administrador y éste necesita conectarse a él de forma segura. Para ello se utiliza el protocolo SSH (**S**ecure **Sh**ell) que ejecuta una shell en un sistema remoto encriptando las comunicaciones.

También se pueden utilizar herramientas que permiten hacer copias, transferencias de archivos, etc., sobre SSH, como por ejemplo *scp*, *sftp*, *rsync*, …

11.2.- SSH

El comando *ssh* implementa el protocolo SSH. Proporciona un intérprete de comandos en el sistema remoto y es capaz de redirigir el tráfico de X para ejecutar programas con entorno gráfico en un sistema que no tiene porqué tenerlo. La sintaxis del comando *ssh* es:

```
ssh <usuario_remoto>@<sistema_remoto>
```

si no se proporciona el usuario remoto, se conectará al sistema remoto con el usuario con el que haya hecho *login* en el sistema de origen.

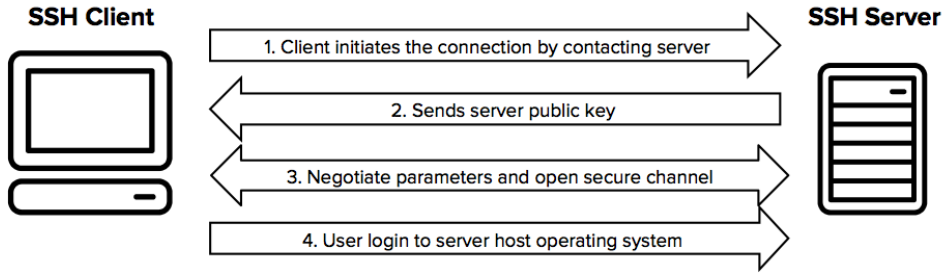
Se puede incluso ejecutar un comando de forma remota sin necesidad de iniciar una shell con:

```
ssh <usuario_remoto>@<sistema_remoto> <comando>
```

se pueden pasar varios comandos separados por ; si los entrecomillamos, bien con comillas simples o dobles. Con el comando *w -fi* puedo ver qué usuarios están conectados al sistema y si están conectados por SSH, veré desde qué sistema de origen se han conectado.

11.2.1.- Claves SSH

SSH securiza la comunicación usando una encriptación de clave pública. Cuando un cliente SSH conecta con un servidor SSH, el servidor le envía una copia de su clave pública para que cualquier cosa que envíe el cliente al servidor, lo encripte con dicha clave. En el servidor, se utilizará la clave privada para desencriptar lo que se ha recibido del cliente.



Funcionamiento de SSH

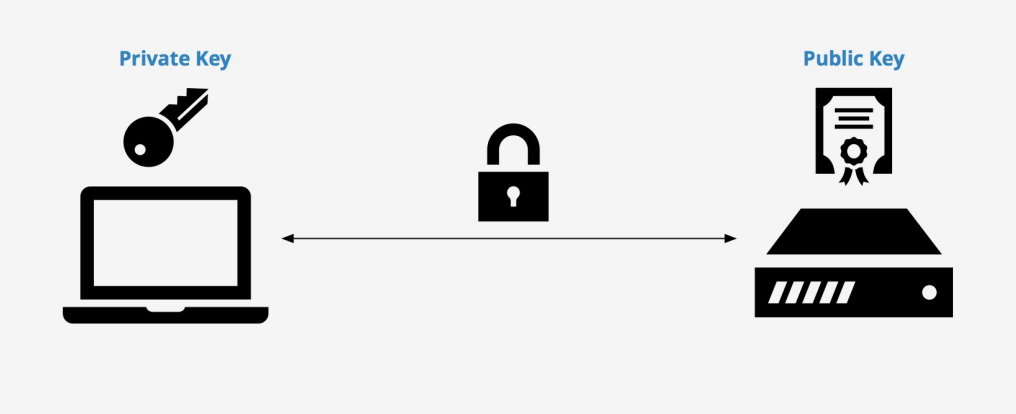
La primera vez que nos conectamos a un servidor SSH, en el home del usuario, dentro del directorio *.ssh*, se almacena la clave pública del servidor SSH en el archivo *known_hosts*. Cada vez que se vuelve a realizar una conexión a ese servidor SSH, se comprueba que la clave pública almacenada es igual a la que envía el servidor; si no coinciden, el cliente asume que la conexión se ha visto comprometida y cierra la conexión inmediatamente.

Si la conexión es fiable, el servidor SSH ha debido cambiar su clave pública, hay que eliminar la línea en ese archivo *~/.ssh/know_hosts* que nos impide la conexión.

En el servidor SSH, las claves están almacenadas en */etc/ssh/* con los nombres *ssh_host_**.

11.2.2.- Autenticación basada en claves

Con esto se podrá realizar una conexión SSH a un sistema remoto con un usuario determinado sin que nos pida contraseña. Esto es posible por el intercambio previo de claves entre los sistemas.



Autenticación basada en claves

Pasos para configurar la autenticación basada en claves:

1.- Generar el par de claves pública-privada, utilizando el comando:

```
ssh-keygen
```

Por defecto se guardan ambas claves en el home del usuario, dentro del directorio *.ssh*. La clave pública *id_rsa.pub* con permisos 644 y la clave privada *id_rsa* con permisos 600, para que nadie tenga acceso a ella.

2. Copiar la clave pública en el sistema remoto con el comando:

```
ssh-copy-id <usuario_remoto>@<sistema_remoto>
```

Pedirá la contraseña del usuario remoto en el sistema remoto y a partir de este momento, podremos conectarnos con ese usuario al sistema remoto sin que nos pida la contraseña.

NOTA: Si en el sistema local no tiene instalado el paquete *openssh-clients*, no se puede usar el comando *ssh-copy-id;* se puede simplemente copiar el contenido del archivo de clave pública dentro del archivo *authorized_keys* del directorio .*ssh* del home del usuario remoto:

cat ~/.ssh/id_rsa-pub | ssh <usuario_remoto>@<sistema_remoto> “cat >> ~/.ssh/authorized_keys”

11.2.3.- Securizar el servidor SSH

Para securizar el servidor SSH, hay que hacer una serie de cambios en su archivo de configuración */etc/ssh/sshd_config*. Una vez realizado cualquier cambio en dicho archivo de configuración, hay que reiniciar el servicio *sshd*:

```
systemctl restart sshd
```

Los cambios recomendables que hay que realizar en la configuración del servidor SSH:

- *Cambiar el puerto de acceso a SSH*: por defecto, SSH escucha por el puerto 22, para modificar este puerto, modificaremos la línea *Port 22* y pondremos el número de puerto a usar. Este nuevo puerto habrá que abrirlo en el *firewall* y cerrar el 22 además de etiquetarlo con el contexto de SELinux de *ssh_port_t*.
- *Evitar que root tenga acceso por SSH*: se modifica la linea *PermitRootLogin yes*, cambiando *yes* por *no*. Se puede hacer que root tenga acceso usando sólo autenticación basada en claves, con el valor *without-password* de *PermitRootLogin*.
- *Limitar el número de veces que se puede introducir de forma incorrecta la password*: se modifica la línea *MaxAuthTries 6* a otro valor más bajo, p.e. 3.
- *Obligar a que los usuarios utilicen autenticación basada en claves*: con la línea *PasswordAuthentication* con valor *no*.
- *Limitar los usuarios que pueden usar SSH*: en la línea *AllowUsers* se ponen los username de los usuarios separados por espacios y/o en la línea *AllowGroups* los nombres de grupo separados por espacios. También se puede hacer a la inversa, restringir qué usuarios no se pueden conectar por SSH con la línea *DenyUsers* o *DenyGroups.*

  Se puede añadir el hostname o la ip desde donde se conecta el usuario añadiendo tras su username, el carácter @ y dicho host o ip.
- *Establecer un timeout de sesión inactiva*: en la línea *ClientAliveInterval* se pone un número de segundos razonable, p.e. 300, que equivale a 5 minutos sin actividad.
- *Desconectar si el cliente no se ha hecho login en un intervalo razonable*: en la línea *LoginGraceTime* se pone un número de segundos no muy alto, p.e. 60 o 120.
- *Limitar el número de logins simultáneos*: en la línea *MaxStartups* y un número de *login*s.

Otra forma de restringir el acceso por SSH a un sistema, es limitar el acceso al puerto del SSH con el *firewall* o utilizar los archivos */etc/hosts.deny* y */etc/hosts.allow.* De esta forma podemos permitir o restringir el acceso a ciertas subredes o sistemas en concreto. Lo habitual es denegar a cualquier subred el acceso en el archivo */etc/hosts.deny* con esta línea:

```
sshd: all
```

e ir añadiendo las subredes o sistemas en concreto en el archivo */etc/hosts.allow*:

```
sshd: <subred> <sistema1> <sistema2>
```

11.2.4.- Ejecutar aplicaciones gráficas remotas

X11 es el servidor gráfico en Linux que una de las cosas que permite, es hacer *forwarding* a través de SSH, haciendo posible ejecutar aplicaciones gráficas de un sistema remoto, exportando el display a nuestro sistema gráfico local. La aplicación se ejecuta en el servidor remoto, pero la interfaz gráfica se visualiza en el sistema local.

Primero se necesita en el sistema local tener configurado el servidor de X, y con la opción *-X* en el comando *ssh*, abriremos una sesión con las X exportadas. Al loguearnos de esta forma se genera un archivo *.Xauthority* en el home del usuario local que es una clave pública usada para conectarnos por SSH al servidor.

Para habilitar las X en el servidor SSH, seguiremos los pasos:

1. Instalar el paquete *xorg-x11-xauth:*

   ```
   yum -y install xorg-x11-xauth
   ```

2. Poner el parámetro *X11forwarding* a *yes* en el archivo de configuración:

   ```
   vim /etc/ssh/sshd_config
   X11forwarding yes
   ```

3. Reiniciar el servicio *sshd*:

   ```
   systemctl restart sshd
   ```

ahora al conectarnos a este servidor SSH con:

```
ssh -X <usuario_remoto>@<sistema_remoto>
```

podremos ejecutar aplicaciones gráficas, p.e. *xclock* o *xlogo*.

En el caso de que sea importante incrementar la velocidad de conexión en la ejecución remota de aplicaciones gráficas, en la configuración del servidor SSH, se puede poner el parámetro *ForwardX11Trusted* a *yes* y entonces, utilizar la opción *-Y* en lugar de *-X* en el comando *ssh.* Hay que tener en cuenta de que así la conexión es menos segura ya que se salta algunos protocolos de seguridad y esto no es muy recomendable.

Otra forma de incrementar la velocidad sin perder seguridad, es utilizar la opción *-C* junto con *-X* en el comando *ssh* para que el servidor envíe los datos comprimidos.

11.3.- SCP

El comando *scp* tranfiere archivos entre sistemas utilizando el servidor SSH para autenticar y encriptar las conexiones. Los archivos en el sistema remoto se especifican con el formato:

```
<usuario_remoto>@<sistema_remoto>:/<ruta>/<archivo>
```

si el usuario remoto es el mismo que el local, podemos no poner la parte de *<usuario_remoto>@*.

La sintaxis del comando *scp* es:

```
scp <origen> <destino>
```

siendo el origen y el destino, uno en la máquina local y otro en un sistema remoto.

Utilizando la opción *-r* copiamos un directorio de forma recursiva, con todo su contenido.

11.4.- SFTP

Una sesión con el comando *sftp* es muy parecida a las sesiones FTP tradicionales, salvo que, al igual que con *scp*, la autenticación y el encriptado de las comunicaciones se hace a través del servidor SSH. Para establecer una conexión SFTP utilizaremos:

```
sftp <usuario_remoto>@<sistema-remoto>
```

Los comandos que podemos utilizar dentro del prompt de SFTP son los habituales *ls*, *cd*, *mkdir*, *rmdir* y *pwd*. Para realizar una copia al sistema remoto utilizaremos el comando *put* y para obtener un archivo o directorio del sistema remoto al local, utilizamos *get*. Con *exit* salimos de la sesión SFTP.

11.5.- RSYNC

La herramienta *rsync* es otra forma de copiar archivos entre sistemas, a diferencia de *scp* sólo copia las diferencias, lo que ya existe y es igual no lo copia, sólo copia lo nuevo, con lo que ganamos tiempo. Sintaxis de *rsync*:

```
rsync <opciones> <origen> <destino>
```

donde el origen y el destino pueden uno local y otro remoto, y se usa el mismo formato que en *scp*, o pueden ser ambos locales, cosa que no permite *scp*.

Tiene una opción muy interesante, *-n*, que permite una simulación de lo que haría pero sin llegar a realizarlo.

Las opciones más usadas son:

-*i*: sincroniza recursivamente un directorio.

-*l*: sincroniza enlaces simbólicos.

-*p*: preserva los permisos de los archivos y directorios.

-*t*: preserva los *timestamps*.

-*g* y -*o*: preservan el grupo propietario y usuario propietario respectivamente.

-*D*: sincroniza archivos de dispositivos.

-*a*: activa todas las anteriores.

-*H*: preserva enlaces físicos.

-*v*: modo *verbose*.

-*A*: preserva ACLs.

-*X*: preserva contextos SELinux.

Hay que tener cuidado cuando especificamos directorios porque si los terminamos con /, nos estamos refiriendo al contenido del directorio y si no, nos referimos al directorio.

11.2.6.- Caso práctico

En **central** se va a restringir el acceso a *root* y los usuarios deberán utilizar autenticación basada en claves para cualquier conexión SSH.

En **server1** no se van a permitir las conexiones SSH salvo al usuario *admin* y al usuario *root* y deberán hacerlo con autenticación basada en claves. Las conexiones inactivas tendrán un *timeout* de 3 minutos y no se podrá introducir la contraseña más de tres veces.

El usuario *admin* va a transferir el archivo */var/log/messages* de **central** a **server1**, dejándolo en su home con el nombre *messages_central* utilizando los tres comandos *scp*, *sftp* y *rsync*.

RESOLUCIÓN

- En **central**, vamos a configurar la autenticación basada en claves para *admin* desde **server1**:

```
[admin@server1 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/admin/.ssh/id_rsa):
Created directory '/home/admin/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/admin/.ssh/id_rsa.
Your public key has been saved in /home/admin/.ssh/id_rsa.pub.
The key fingerprint is:
6c:a7:70:5b:ad:37:5d:ba:c0:a3:b7:a8:37:b5:3b:78 admin@server1.miempresa.com
The key's randomart image is:
+-----[ RSA 2048]------+
```

```
|        E..              |
|       . o  .            |
|        o o*.            |
|       .. o===..         |
|       S.+ ++. .         |
|      . . o  ..          |
|       .   .  ..         |
|        . .    .         |
|         .               |
+-------------------------+
```

[admin@server1 ~]$ ssh-copy-id central

The authenticity of host 'central (10.11.1.254)' can't be established.

ECDSA key fingerprint is 06:15:41:c8:12:b6:ea:34:b5:b6:40:25:5d:12:a6:a1.

Are you sure you want to continue connecting (yes/no)? **yes**

/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed

/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys

admin@central's password: ← password de admin en central, (no es visible)

Number of key(s) added: 1

- Modificamos la configuración de SSH para que *root* no pueda acceder por SSH y obligar a que los usuarios usen la autenticación basada en claves:

[root@central ~]# vim /etc/ssh/sshd_config

```
…..
#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
……
PasswordAuthentication no

……...
```

[root@central ~]# systemctl restart sshd

- Verificamos que *root* no puede hacer *ssh* a **central** y que *admin* puede hacer ssh y no tiene que introducir la contraseña:

[root@server1 ~]# ssh root@central

Permission denied, please try again.

[admin@server1 ~]# ssh central

Last login: Sat Sep 16 08:35:45 2017 from 10.11.1.

[admin@central ~]#

- En **server1**, vamos a modificar la configuración de SSH para que sólo *root* y *admin* puedan conectar, creando las claves y copiándolas en **server1** desde **central** para ambos usuarios:

[root@central ~]# ssh-keygen

```
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
13:69:75:78:48:61:d9:f2:5e:09:52:85:30:c3:1a:7f root@central.miempresa.com
The key's randomart image is:
+----[ RSA 2048]-------+
|        oOOoo.         |
|       .+B+=           |
|       ++ = . .        |
|       .... E o        |
|       S  o .          |
|        .  .           |
|                       |
|                       |
|                       |
+-----------------------+
```

[root@central ~]# ssh-copy-id server1

```
The authenticity of host 'server1 (10.11.1.101)' can't be established.
ECDSA key fingerprint is 81:44:15:2f:6e:77:e6:98:16:4a:c3:6a:a3:22:c0:fb.
Are you sure you want to continue connecting (yes/no)? yes
/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@server1's password:          ← Introducir la password de root en server1

Number of key(s) added: 1
Now try logging into the machine, with:   "ssh 'server1'"
and check to make sure that only the key(s) you wanted were added.
```

[root@central ~]# su – admin

[admin@central ~]$ ssh-keygen

[admin@central ~]$ ssh-copy-id server1

En **server1**, restringimos SSH para todos los usuarios excepto *root* y *admin:*

[root@server1 ~]# echo “AllowUsers root admin” >> /etc/ssh/sshd_config

[root@server1 ~]# systemctl restart sshd

- Verificamos que con el usuario *hlamarr* de **server1** no se puede hacer *ssh* desde **central** y que el usuario admin :

[root@central ~]# ssh hlamarr@server1

Permission denied (publickey,gssapi-keyex,gssapi-with-mic).

- Modificamos la configuración de SSH en server1 para que los usuarios tengan que utilizar autenticación basada en claves, con un timeout de sesión de 3 minutos y sólo 3 intentos de introducir la clave.

[root@server1 ~]# echo “ClientAliveInterval 180” >> /etc/ssh/sshd_config

[root@server1 ~]# echo “MaxAuthTries 3” >> /etc/ssh/sshd_config

[root@server1 ~]# systemctl restart sshd

- Copiamos el archivo */var/log/messages* de **central** a **server1** con *scp*:

[root@central ~]# scp /var/log/messages admin@server1:/home/admin/messages_central

messages 100% 251KB 251.0KB/s 00:00

- Copiamos el archivo */var/log/messages* de **central** a **server1** con *sftp*:

[root@central ~]# sftp admin@server1

Connected to server1.

sftp> put /var/log/messages ./mensajes_central

Uploading /var/log/messages to /home/admin/./mensajes_central

/var/log/messages 100% 251KB 251.0KB/s 00:00

sftp> exit

- Copiamos el archivo */var/log/messages* de **central** a **server1** con *rsync*:

[root@central ~]# yum install -y rsync

[root@server1 ~]# yum install -y rsync

[root@central ~]# rsync -avhP /var/log/messages

admin@server1:/home/admin/messages_central

sending incremental file list
messages
257.27K 100% 26.76MB/s 0:00:00 (xfer#1, to-check=0/1)

sent 257.38K bytes received 31 bytes 102.96K bytes/sec
total size is 257.27K speedup is 1.00