Introducción a OpenSSH

¿ Qué es OpenSSH ?

OpenSSH (**Open Secure Shell**) es un conjunto de aplicaciones que permiten realizar comunicaciones cifradas a través de una red, usando el protocolo **SSH**. Fue creado como una alternativa libre y abierta al programa **Secure Shell**, que es software propietario.

En principio podríamos pensar en **SSH** como un sustituto del **telnet**, el **rexec** y el **rsh**, pero la realidad es que **SSH** permite mucho más que la simple identificación de usuarios y ejecución de programas en máquinas remotas. A través de una conexión **SSH** se puede encapsular cualquier protocolo y gracias a la habilidad para crear túneles, tanto remotos como locales, podemos llegar a traspasar **routers** y **cortafuegos**.

Telnet (**TELecommunication NETwork**) es el nombre de un protocolo de red que sirve para acceder mediante una red a otra máquina para manejarla remotamente como si estuviéramos sentados delante de ella. También es el nombre del programa informático que implementa el cliente. Para que la conexión funcione, como en todos los servicios de Internet, la máquina a la que se acceda debe tener un programa especial que reciba y gestione las conexiones. El puerto que se utiliza generalmente es el 23, y manda tanto el nombre del usuario como la contraseña como texto plano, es decir con cualquier **sniffer** uno puede obtenerla.

En cambio **SSH** manda tanto el usuario y la contraseña en forma encriptada que no la podemos ver de forma de texto plano.

Aplicaciones Incluidas

Como dijimos dentro del paquete OpenSSH están incluidos los siguientes programas :

- **ssh** reemplaza a **rlogin** y **telnet** para permitir shell el acceso remoto a otra máquina.
  - ssh pepe@ejemplo.com

- **scp** reemplaza a **rcp** nos sirve para copiar archivos y/o directorios.
  - scp archivo_1.c **pepe@ejemplo.com**:/programas

- **sftp** reemplaza al **ftp** para copiar archivos entre dos computadoras, el **ftp** también manda el usuario y contraseña en texto plano.
  - sftp pepe@ejemplo.com

- **sshd** es el demonio de SSH

- **ssh-keygen** una herramienta para inspeccionar y generar claves RSA y DSA que son usadas para la autentificación del cliente o usuario.
  - ssh-keygen -t rsa -b 2048

- **ssh-agent** y **ssh-add**, herramientas para autenticarse de manera mas fácil, manteniendo las claves listas para no tener que volver a introducir la frase de acceso cada vez que utilice la clave.

- **ssh-keyscan** escanea una lista de clientes y recolecta sus claves públicas.

- **ssh-copy-id** instala la clave pública dentro de una maquina remota en **authorized_keys**.

Instalación de OpenSSH

En Debian :

```
# apt-get install openssl openssh-server openssh-client
```

Configuración de OpenSSH

Dentro de **/etc/ssh** tenemos un archivo de configuración del servicio de **ssh** llamado **sshd_config** este servicio por defecto escucha en el puerto **22**, cosa que podemos cambiar entre otras cosas.

```
# vi /etc/ssh/sshd_config

        # Package generated configuration file
        # See the sshd_config(5) manpage for details

        # What ports, IPs and protocols we listen for
        Port 22
        # Use these options to restrict which interfaces/protocols sshd will bind to
        #ListenAddress ::
        #ListenAddress 0.0.0.0
        Protocol 2
        # HostKeys for protocol version 2
        HostKey /etc/ssh/ssh_host_rsa_key
        HostKey /etc/ssh/ssh_host_dsa_key
        #Privilege Separation is turned on for security
        UsePrivilegeSeparation yes

        # Lifetime and size of ephemeral version 1 server key
        KeyRegenerationInterval 3600
        ServerKeyBits 768

        # Logging
        SyslogFacility AUTH
        LogLevel INFO

        # Authentication:
        LoginGraceTime 120
        PermitRootLogin yes
        StrictModes yes

        RSAAuthentication yes
        PubkeyAuthentication yes
        #AuthorizedKeysFile     %h/.ssh/authorized_keys

        # Don't read the user's ~/.rhosts and ~/.shosts files
        IgnoreRhosts yes
        # For this to work you will also need host keys in /etc/ssh_known_hosts
        RhostsRSAAuthentication no
```

```
# similar for protocol version 2
HostbasedAuthentication no
# Uncomment if you don't trust ~/.ssh/known_hosts for RhostsRSAAuthentication
#IgnoreUserKnownHosts yes

# To enable empty passwords, change to yes (NOT RECOMMENDED)
PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication no

# Change to no to disable tunnelled clear text passwords
#PasswordAuthentication yes

# Kerberos options
#KerberosAuthentication no
#KerberosGetAFSToken no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes

# GSSAPI options
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes

X11Forwarding yes
X11DisplayOffset 10
PrintMotd no
PrintLastLog yes
TCPKeepAlive yes
#UseLogin no

#MaxStartups 10:30:60
#Banner /etc/issue.net

# Allow client to pass locale environment variables
AcceptEnv LANG LC_*

Subsystem sftp /usr/lib/openssh/sftp-server

# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# be allowed through the ChallengeResponseAuthentication and
# PasswordAuthentication.  Depending on your PAM configuration,
# PAM authentication via ChallengeResponseAuthentication may bypass
# the setting of "PermitRootLogin without-password".
# If you just want the PAM account and session checks to run without
# PAM authentication, then enable this but set PasswordAuthentication
# and ChallengeResponseAuthentication to 'no'.
UsePAM yes
```

Algunas modificaciones que podemos realizar :

- **Port 22** : puerto por defecto que funciona el ssh (22).

- **ListenAddress 0.0.0.0** : por defecto si tengo varias placas de red, estaría escuchando en todas. Puedo especificar que escuche por una IP determinada por ejemplo :
  - ListenAddress 192.168.1.10

- **Protocol 2** : le indico que solo utilice el protocolo 2 que no funcione en protocolo 1 porque tiene ciertos bugs. Es decir que solo pueden comunicarse aquellos que solo usen el protocolo 2.

- **PermitRootLogin yes** : permito que el administrador es decir **root** pueda conectarse. Esto no es bueno, es conveniente poner que **no** y que se comunique por un usuario y luego pase a ser **root** por seguridad.

- **PermitEmptyPasswords no** : esto le indicamos que el usuario que se conecte tiene que tener una password.

- **X11Forwarding yes** : esto permite que si yo tengo parte gráfica puede ejecutar un programa gráfico en el servidor y me lo mostraria en mi máquina local es decir en mi desktop.

- **Subsystem sftp /usr/lib/openssh/sftp-server** : permite levantar el servicio **sftp**.

Ejemplos :

```
# ssh pepe@ejemplo.com
# scp mi_archivo.c pepe@ejemplo.com:/Programas
# sftp pepe@ejemplo.com
```