Introducción y configuración de SSH

Qué es SSH?

Hay muchas maneras de acceder de manera remota a la **Shell** (Línea de Comando) en los sistemas Linux. La manera más Simple y antigua de acceder es Telnet, pero por lo tanto es la más vulnerable.

Las sesiones de telnet pueden ser vistas mediante el uso de programas como **Ethereal** (Actualmente conocido como **Wireshark**) o **tcpdump**.

Al tratar de subsanar estos problemas surgió SSH (Security Shell).

Entre sus principales ventajas posee:

- Encriptación de la sesión.
- Mejores servicios de autenticación.
- Transferencia de archivo segura (**SCP**, **SFTP** y servicios de escritorio remoto).
- Redirección de sesión y de puertos (conexión mediante **TCP**).
- Y muchas posibilidades más que incluyen el incremento de la seguridad en distintos protocolos.

Funcionamiento

All conectarse un cliente de SSH con el servidor, se realizan los siguientes pasos:

1. El cliente abre una conexión **TCP** al puerto **22** del host servidor.
2. El cliente y el servidor acuerdan la versión del protocolo a utilizar, de acuerdo a su configuración y capacidades.
3. El servidor posee un par de claves pública/privada de **RSA** (llamadas ``claves de host''). El servidor envía al cliente su clave pública.
4. El cliente compara la clave pública de host recibida con la que tiene almacenada, para verificar su autenticidad. Si no la conociera previamente, pide confirmación al usuario para aceptarla como válida.
5. El cliente genera una clave de sesión aleatoria y selecciona un algoritmo de cifrado simétrico.
6. El cliente envía un mensaje conteniendo la clave de sesión y el algoritmo seleccionado, cifrado con la clave pública de host del servidor usando el algoritmo **RSA**.
7. En adelante, para el resto de la comunicación se utilizará el algoritmo de cifrado simétrico seleccionado y clave compartida de sesión.
8. Luego se realiza la autenticación del usuario.
9. Finalmente se inicia la sesión, por lo general, interactiva.

Seguridad

SSH utiliza diferentes formas de encriptación que abarcan desde **512 bits** hasta **32 Kbits** e incluyen cifrados como **AES**, **3DES**, **Blowfish**, **CAST 128** o **Arcfour**. Obviamente entre mas grandes sean las cantidades de bits para el cifrado más tiempo se tardará en generar y usar las claves y más tiempo demorara el pasaje de información por medio de la coneccion.

Comenzando

La mayoría de las versiones posteriores al 2002 ya poseen OpenSSH
Si no llegara a ser tu caso vamos a instalarlo: sudo apt-get install openssh-server

Comandos que debemos tener en cuenta

Para editar la configuración del servidor SSH debemos hacer en consola:

sudo gedit /etc/ssh/sshd_config

Para arrancar el servidor:

sudo /etc/init.d/ssh start

Mensaje: *Starting OpenBSD Secure Shell server sshd

Para parar el servidor:

sudo /etc/init.d/ssh stop

Mensaje: * Stopping OpenBSD Secure Shell server sshd

Para reiniciar el servidor:

sudo /etc/init.d/ssh restart

Mensaje: * Restarting OpenBSD Secure Shell server sshd

Si queremos saber si el servidor esta iniciado y funciona de manera correcta:

ssh usuario@ip_servidor

- Por ejemplo, si mi usuario es **internetlan** y la ip: **192.178.1.205**, pondremos en el terminal: **ssh internetlan@192.178.1.205.**
- Si está iniciado mostrará: **sshd us running**.

Configuración de OpenSSH

OpenSSH dispone de dos conjuntos diferentes de ficheros de configuración: uno completamente dedicado al **cliente** (ssh, scp y sftp) y otro orientado completamente al **servidor**.

Lado del Servidor

Archivos de configuración del servidor

La ubicación de los ficheros referentes al servidor se encuentran en la ruta: **/etc/ssh/**

Dentro del directorio podemos encontrar los siguientes ficheros de configuración:

moduli: Contiene **grupos Diffie-Hellman** usados para el intercambio de la clave Diffie-Hellman que es imprescindible para la construcción de una capa de transporte seguro. Cuando se intercambian las claves al inicio de una sesión SSH, se crea un valor secreto y compartido que no puede ser determinado por ninguna de las partes individualmente. Este valor se usa para proporcionar la autenticación del host.

ssh_config: El archivo de configuración del sistema cliente SSH por defecto. Este archivo se sobrescribe si hay alguno ya presente en el directorio principal del usuario.

sshd_config: El archivo de configuración para el demonio sshd.

ssh_host_dsa_key: La clave privada DSA usada por el demonio sshd.

ssh_host_dsa_key.pub: La clave pública DSA usada por el demonio sshd.

ssh_host_rsa_key: La clave privada RSA usada por el demonio sshd para la versión 2 del protocolo SSH.

ssh_host_rsa_key.pub: La clave pública RSA usada por el demonio sshd para la versión 2 del protocolo SSH.

Configuración del servidor

La función que desempeñan los ficheros de configuración de **openSSH** son de vital importancia para la seguridad de nuestro servidor , ya que si no se llegaran a configurar apropiadamente estos ficheros la vulnerabilidad de nuestro servidor sería demasiado sensible a ataques informáticos.

vamos a configurar el servidor, hacemos en consola:

sudo gedit /etc/ssh/sshd_config

Port: SSH tiene asignado por defecto el puerto 22, esto es algo que conocen todos nuestros posibles atacantes , por lo que es una buena idea cambiarlo.

Para modificar esta opción y las siguientes que iremos mencionando editaremos el fichero de configuración **sshd_config**, que por defecto se encuentra en el directorio **/etc/ssh/**. Se recomienda usar un puerto cualquiera **por encima del 1024**, así que usted puede elegir el que quiera. En el ejemplo usaremos el 2222.

- Por cierto, si has cambiado el puerto para comprobar que la instalación ha sido correcta tendrás que introducir lo siguiente: ***ssh -p 2222 interntelan@192.178.1.205***

Protocol: Hay dos versiones de ssh en cuanto a su protocolo de comunicación, estas son:

- Versión 1.
- Versión 2.

La **versión 1** de openSSH hace uso de varios algoritmos de cifrado de datos más, sin embargo algunos de estos algoritmos han dejado de ser mantenidos por sus creadores y por lo tanto presenta serios huecos de seguridad que potencialmente permite a un intruso insertar datos en el canal de comunicación. Para evitar el uso del protocolo 1 y sus posibles ataques a este, basta con indicar que solo admita comunicaciones de ssh basadas en el protocolo 2.

Por default en debian/ubuntu ya viene configurado en el versión 2 del protocolo.

PermitRootLogin: Sin modificar nada podemos loguearnos como **Root** por **SSH**, este es un error de seguridad básico ya que un posible atacante sólo puede necesitar la contraseña y con esto tiene el 50% del trabajo realizado. Si configuramos como no ahora tendrá que buscar tanto el nombre de usuario como la contraseña. Con esto **siempre tendremos que ingresar como un usuario normal** y ya estando adentro entonces mediante un su – cambiarnos a la cuenta de root.

Parámetro X11Forwarding: Si nuestro servidor no tienen entorno gráfico instalado, o no queremos que los usuarios se conecten a él, definiremos esta opción en el fichero de configuración (**poner en no**).

ListenAddress: Si nos conectamos siempre desde una computadora con IP estática es recomendable restringir la autenticación solo a esas ip's. **Por defecto no está configurado.**

AllowUsers: Si queremos restringir el servicio a solo algunos usuarios los escribimos de la forma **usuario@host**. Este parámetro sólo permitirá la conexión a los usuario que estén

registrados en el archivo de configuración del **openSSH**, el parámetro no existe nosotros lo podemos agregar el parámetro al final del archivo.

AllowGroups: Permite especificar el grupo o los grupos de usuarios a los que se les permite hacer acceder mediante **SSH** al sistema. Lo mismo que lo anterior.

MaxAuthTries: Con esta opción podemos especificar el número máximo de intentos antes de cerrar la conexión, siempre se sugiere un número bajo por ejemplo 3.

LoginGraceTime: El número indica la cantidad de segundos en que la pantalla de login estará disponible para que el usuario capture su nombre de usuario y contraseña, si no lo hace, el login se cerrará, evitando así dejar por tiempo indeterminado pantallas de login sin que nadie las use, o peor aún, que alguien esté intentando mediante un script varias veces el adivinar un usuario y contraseña.

Banner: Si seleccionamos el **protocolo 2** podemos activar esta opción. En caso de que el sistema detecte intentos de intrusión puede mostrar un archivo con un mensaje de advertencia.

MaxStartups: Aquí declaramos el número máximo de conexiones abiertas, es decir que solicitan una conexión.

Diferencia entre restart y reload

restart reinicia el servicio, al ejecutar un restart este mata todos los procesos relacionado con el servicio y los vuelve a generar de nuevo. En cambio, **reload** recarga el servicio, al ejecutar un reload este solamente carga las actualizaciones hechas al fichero de configuración del servicio sin necesidad de matar los procesos relacionados con el mismo (**cambio en caliente**)

Para evitar que todos los usarios de la máquina estén accesibles por SSH tendremos que editar como root el fichero /etc/ssh/sshd_conf (o /etc/ssh/sshd_config), y añadir la líneaAllowUsers y a continuación los usuarios que deseamos que se puedan conectar remotamente via SSH:

```
$ sudo gedit /etc/ssh/sshd_conf
```

o

```
$ sudo gedit /etc/ssh/sshd_config
```

Y añadimos la línea:

```
AllowUsers usuario1 usuario2 usuario3
```

Para que los cambios surtan efectos habrá que reiniciar el servidor:

```
$ sudo /etc/init.d/ssh restart
```

```
* Restarting OpenBSD Secure Shell server...
```

SSHFS o montar directorio remoto con SSH

Algunas veces necesitamos trabajar durante bastante tiempo con un sistema remoto, copiando y editando ficheros. Existe la posibilidad de usar Nautilus que se puede conectar remotamente al servidor usando el protocolo '*sftp://*, pero suele ser lento y muchas veces es más práctico usar la línea de comandos. La solución sería montar un recurso remoto al estilo de *NFS* y encima sin tener que se *root*. Pues si, se puede :)

Fuse es un módulo del kenel que permite montar distintos sistemas de ficheros con un usuario normal sin privilegios. SSHFS es un programa creado por el autor de *fuse* que permite montar un directorio remoto usando SSH. Accederemos localmente como si estuviera en nuestra propia máquina. Hay que instalar el programa **sshfs**. El usuario que puede montar el sistema de ficheros tiene que pertenecer al grupo **fuse**.

```
$ sudo usermod -G fuse -a usuario_local
```

Tenemos que salir y volver a entrar en el sistema para que se haga efectivo el cambio de grupo. Antes de empezar a usar *fuse* tienes que cargar el módulo en memoria:

```
$ sudo modprobe fuse
```

Ahora vamos a hacer una prueba:

```
$ mkdir ~/directorio_remoto
$ sshfs usuario_remoto@servidor_remoto:/tmp ~/directorio_remoto
```

Este comando monta el directorio */tmp* del servidor remoto en el directorio local *~/directorio_remoto*. Ahora podemos trabajar en el directorio montado como si de un directorio local se tratase. Para desmontar el directorio:

```
$ fusermount -u ~/directorio_remoto
```

Si vamos a trabajar a diario con este directorio remoto, quizás es buena idea añadirlo al fichero */etc/fstab*. De esta forma se montará automáticamente al iniciar nuestro ordenador o manualmente (si elegimos la opción *noauto*) sin necesidad de especificar la localización remota cada vez. Este es un ejemplo de configuración:

```
$ sshfs#usuario_remoto@remote_server:/tmp /home/usuario_local/directorio_remoto fuse defaults,auto 0 0
```

Si vamos a usar *fuse* y *sshfs* regularmente, tendrías que editar el fichero */etc/modules* y añadir el módulo *fuse*. De otra forma tendrás que cargar el módulo manualmente cada vez que lo quieras usar:

```
$ sudo sh -c "echo fuse >> /etc/modules"
```

Enjaular un usuario con OpenSSH en Ubuntu

Imaginemos que queremos crear un usuario a nuestro amigo Sanobis, pero no queremos que pueda ver todos los archivos del sistema, es decir, vamos a enjaularle en su directorio /home/ únicamente.

Nos bajamos este archivo:

http://cdn.redeszone.net/down/soft/redes/make_chroot_jail.sh

Y lo ponemos en el directorio raíz por comodidad.

Le asignamos permisos 700

sudo chmod 700 make_chroot_jail.sh

A continuación escribimos:

bash make_chroot_jail.sh sanobis

Tecleamos yes, y a continuación metemos la clave que queramos que tenga (estamos creando un usuario nuevo).

Ahora, su directorio enjaulado es **/home/jail/home/sanobis**

Salimos y entramos por ssh y veremos que efectivamente no podrá salir de ese directorio. No hace falta que reiniciemos el servidor SSH.

http://www.youtube.com/channel/UC6xWvt7EgotXNnnvB08SwhA?feature=watch

Archivos de configuración del cliente

La ubicación de los ficheros referentes al cliente se encuentran almacenados en el directorio de trabajo de cada usuario:

Ejemplo: “/home/miusuario/.ssh/”

Dentro del directorio podemos encontrar los siguientes ficheros de configuración:

identity.pub La clave pública RSA usada por ssh para la versión 1 del protocolo SSH.

known_hosts Este archivo contiene las claves de host DSA de los servidores SSH a los cuales el usuario ha accedido. Este archivo es muy importante para asegurar que el cliente SSH está conectado al servidor SSH correcto.

Conexión a un servidor remoto

```
$ ssh usuario_remoto@host_remoto
```

Donde *host_remoto* es la IP del servidor SSH o el nombre de este. Luego nos pide la contraseña del usuario. La primera vez que nos conectemos a un servidor tarda un poco más y nos pide confirmación tecleando "yes" con todas sus letras, las subsiguientes ya no. Sabemos que estamos conectados porque el prompt cambia y aparece en lugar del nombre de nuestro host el nombre del host remoto.

Los comandos, programas y scripts que lancemos tras conectarnos se ejecutarán en la máquina a las que nos hayamos conectado, utilizando los recursos del host remoto (CPU, memoria, disco, etc.). Esta arquitectura puede utilizarse, por ejemplo, para tener un servidor más potente y varios clientes que ejecutan aplicaciones en dicha máquina.

Para ejecutar aplicaciones gráficas en la máquina a la que nos conectamos tenemos dos opciones. La primera consiste en definir la variable $DISPLAY apuntando a la máquina desde la que nos conectamos.

```
$ export DISPLAY=host_local:0.0
```

Este mecanismo no se recomienda por motivos de seguridad (el protocolo X11 no se encuentra cifrado) y, además, pueden encontrarse problemas porque cortafuegos intermedios bloqueen ese tráfico (puertos 600x TCP).

Una solución mejor es utilizar un túnel SSH para encapsular el protocolo X11, lo que transmite la información de manera segura y, además, no suele dar problemas con los cortafuegos intermedios.

Para poder ejecutar aplicaciones gráficas en el host remoto de forma segura, necesitamos dos cosas. La primera, que en la configuración del servidor SSH del host remoto (/etc/ssh/sshd_config) se encuentre activada la siguiente opción:

```
X11Forwarding yes
```

Para aprovechar esta característica, hemos de conectarnos usando el parámetro -X, lo que exportará la configuración de la variable $DISPLAY con lo que podremos ejecutar aplicaciones gráficas de forma remota:

```
$ ssh -X usuario_remoto@host_remoto
```

Ahora si ejecutas el programa *xclock* verás que la ventana sale en tu escritorio:

```
$ xclock
```

Evitar que nos pida el password cada vez

Siempre que conectemos a un ordenador remoto con SSH nos va a pedir el password de acceso para asegurarse de que tenemos permisos. Hay una forma de evitar que nos pida el password siempre haciendo que el ordenador remoto confíe en nosotros. Para ello hemos de generar un par de claves DSA que se usarán en la relación de confianza:

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

```
Generating public/private dsa key pair.
Enter file in which to save the key (/home/usuario/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/usuario/.ssh/id_dsa.
Your public key has been saved in /home/usuario/.ssh/id_dsa.pub.
The key fingerprint is:
81:a1:d3:96:fa:a2:73:4d:3a:76:c1:fd:3f:6a:6f:1e usuario@localhost
```

Una vez generada la copiamos al usuario del ordenador remoto con el que queremos mantener la relación de confianza usando el comando *ssh-copy-id*. Este es un ejemplo del uso con la salida del programa:

```
$ ssh-copy-id usuario_remoto@192.168.0.1
```

```
Now try logging into the machine, with "ssh 'usuario_remoto@192.168.0.1'", and check in:
```

```
.ssh/authorized_keys
```

to make sure we haven't added extra keys that you weren't expecting.

Ahora solo hay que arrancar el agente SSH para que recuerde nuestra contraseña. Nos pregunta la contraseña y ahora ya podemos acceder al ordenador remoto sin tener que escribir el password cada vez.

```
$ ssh-add
```

El agente sólo dura mientras esté activa nuestra consola. O sea que si la cerramos y la volvemos a abrir ya no está activo el agente y nos volverá a pedir el password cada vez. Para evitar esto lo que podemos hacer es añadir el agente a la sesión para que se ejecute cada vez que se inicie GNOME. Al iniciar saldrá un diálogo que nos preguntará por la contraseña.

Copia de seguridad

Si vas a migrar la configuración de tu usuario a otra máquina tienes que conservar el directorio *$HOME/.ssh*. Ahí es donde se guarda los ficheros de la clave pública y privada generadas en el punto anterior:

```
$ tar czf ssh.tgz $HOME/.ssh
```

Ejecutar comandos en un host remoto

Ahora que ya sabemos como entrar en un ordenador remoto sin el password, por qué no ejecutar comandos remotamente? Se abre un nuevo mundo de posibilidades muy útiles como tareas que se pueden ejecutar de forma automática.

```
$ ssh usuario_remoto@host_remoto "find /tmp -name *.txt"
```

Este comando buscará los ficheros de texto del directorio temporal que hay en el ordenador remoto. Si bien es un ejemplo poc útil, en la vida de un administrador de sistema hay muchos buenos ejemplos.

Y vamos a dar una nueva vuelta de rosca: si estamos administrando sistemas Unix/Linux es muy posible que necesitemos ejecutar el mismo programa en varios servidores a la vez. Este sencillo script e suna gran ayuda en esos casos:

```
#!/usr/bin/perl
```

```
@hosts=(
   "usuario1\@maquina1.guay.es",
   "usuario2\@maquina2.guay.es"
);

die "Uso: runonall \'command\'\n" unless $ARGV[0];

foreach(@hosts){
  print "$_ -> $ARGV[0]:\n";
  print `ssh $_ $ARGV[0]`;
}
```

Enviar u obtener archivos y carpetas con scp

En una consola o terminal tecleamos:

```
$ scp -r usuario@maquina:/home/carpeta .
```

luego ponemos la password del usuario y nos copiará la carpeta "/home/carpeta" remota en el directorio actual "." naturalmente siempre que usuario tenga permisos sobre la carpeta y su cuenta esté entre las de los que pueden hacer ssh. La opción "-r" significa recursivo, es decir, copia la carpeta y todo su contenido, incluidas las subcarpetas y el contenido de éstas. Si no lo ponemos la orden para copiar todos los archivos de una carpeta sería:

```
$ scp usuario@maquina:/home/carpeta/* .
```

Si lo que queremos es enviar una carpeta con su contenido, utilizaremos la orden:

```
$ scp /home/carpeta/* usuario@maquina:/carpeta/
```

Hacer todo esto en modo gráfico

Si lo que que queremos es conectarnos a un ordenador por ssh en modo gráfico también lo podremos hacer. Es más simple pero también tienes menos posibilidades, o para hacer ciertas cosas tardas más incluso que haciéndolo por consola pero para las operaciones fáciles como administrar, editar un fichero también es posible hacerlo de esta manera.

Entraremos en **Lugares -> Conectarse al servidor...**

Elegiremos en el combo de arriba en lugar de FTP público, SSH y pondremos la IP o el nombre (en caso de anteriormente ya hayamos editado el fichero hosts) y así despues de teclear la contraseña y aceptar estaremos ya dentro del mismo.

Definiciones:

- Claves RSA: El **sistema criptográfico con clave pública RSA** es un algoritmo asimétrico cifrador de **bloques**, que utiliza una clave pública, la cual se distribuye (en forma autenticada preferentemente), y otra privada, la cual es guardada en secreto por su propietario.
  Los mensajes enviados usando el **algoritmo** RSA se representan mediante números y el funcionamiento se basa en el producto de dos **números primos** grandes (mayores que 10^{100}) elegidos al azar para conformar la clave de descifrado.
  https://seguinfo.wordpress.com/2007/09/14/%C2%BFque-es-rsa/