ssh

SSH

Clientes SSH

- http://sourceforge.net/directory/internet/ssh/freshness:recently-updated/
- http://sourceforge.net/projects/pacmanager/?source=directory

Para Windows

- http://smartty.sysprogs.com/

Automatizar SSH

Una de las cosas más frecuentes es conectarte por ssh a varias máquinas y ejecutar los mismos comandos. Para automatizar esta tarea podemos usar varias herramientas:

- ClusterSSH http://sourceforge.net/projects/clusterssh/
- Paramiko http://www.lag.net/paramiko/

Bastionado

http://inteligenciaux.wordpress.com/2010/09/14/aseguramiento-ssh-linux/#more-1319

Trucos

- http://www.openredes.com/2011/09/23/los-25-mejores-comandos-trucos-ssh/
- http://inteligenciaux.wordpress.com/2010/05/26/redhat-enjaular-usuarios-ssh/

Instalación de un servidor SSH

```
yum install openssh-server
```

Configuración

El fichero de configuración está en /etc/ssh/sshd_config

Es recomendable hacer los siguientes cambios:

- Cambiar el puerto 22 por otro→ Port 22022
- Usar sólo la versión 2 del protocolo→ Protocol 2
- Denegar iniciar la sesión como root → PermitRootLogion no

- Definir la interfaz por la que vamos a escuchar (en caso de tener varias) → ListenAddress 192.168.1.10

Uso del SSH

la forma de conectar por ssh es

```
ssh ctausuario@ip_destino
```

Si en vez del puerto 22 se usa otro puerto

```
ssh -p puerto ctausuario@ip_destino
```

Configurar SSH

La configuración del servicio ssh se encuentra en /etc/ssh/sshd_config

- Protocol: Versión a usar del protocolo ssh, lo mejar es usar la versión 2
- LoginGraceTime: tiempo para hacer login
- PermitRootLogin: este valor indica si es posible hacer login como root, lo mejor es poner que no y que el usuario ejecute sudo si lo necesita
- AllowUsers: podemos indicar que usuarios si pueden conetarse por ssh e incluso indicar desde que ip se permite. Basta poner el nombre del usuario o con usuario@ip
- MAxAuthTries: Número máximo de intentos para hacer login
- MaxStartups: Número máximo de usuarios conectados simultaneamente

Desactivar el login del root por ssh

Proxy Socks con SSH

Podemos crear un proxy a partir de una conexión ssh con un comando del tipo:

ssh -D 9999 USUARIO@IP

Y en el navegador seleccionar en la configuración del proxy 'localhost' como servidor SOCKS y 9999 como puerto.

Otra opción es ejecutar

```
$ ssh -f -N -D 9999 usuario@hostremoto
```

Con el flag -f ejecutamos SSH en segundo plano. Con el flag -N le decimos que no vamos a ejecutar ningún comando, por lo que no nos dará acceso a la consola. El flag -D es el que crea una redirección de puertos local a nivel de aplicación.

Están soportadas las versiones SOCKS4 y SOCKS5. La principal diferencia entre las dos es que la versión 5 incorporando autenticación. Sólo el root puede redirigir puertos bien conocidos.

Si una aplicación no soporta el proxy, podemos usar **tsocks**. tsocks, permite que cualquier aplicación utilice este tipo de proxies de forma transparente.

Lo instalamos

```
sudo apt-get install tsocks
```

, lo configuramos el fichero /etc/tsocks.conf:

```
server = 127.0.0.1
server_type = 5
server_port = 9999
```

Para utilizarlo :

```
tsocks telnet micorreo.org 25
```

Para las aplicaciones en java hay que ejecutar lo siguiente

```
java   -DsocksProxyHost=127.0.0.1   -DsocksProxyPort=1080
MiAplicacionJava
```

podemos usar autossh en vez de ssh para aquellos casos en que se nos corte la comunicación ya que con autossh el sólo vuelve a reconectar

Túneles ssh

En este ejemplo, vamos a conectarnos por ssh al equipo 99.99.99.99 por el puerto 9999 y como usuario “bender” y redirigiremos el puerto 5900 (el puerto por defecto para vnc) del equipo de la red remota con ip 192.168.0.99 al puerto 1111 de nuestra máquina:

ssh -p 9999 -P -L 1111:192.168.0.99:5900 bender@99.99.99.99

Como resultado, si tenemos el vnc server trabajando en 192.168.0.99, podríamos acceder a él por el puerto forwadeado:

127.0.0.1:1111

Para conectarnos a otra máquina por ssh:

ssh -p puerto usuario@ip

Si queremos copiar un fichero:

scp -P puerto fichero usuario@ip:directorio

Si queremos ejecutar aplicaciones gráficas añadimos el parámetro -X:

ssh -X -p puerto usuario@ip

En caso de querer ejecutar un navegador por este método, es recomendable usar epiphany, pues es mucho más liviano que otros.

SSH sin pedir contraseña

Para validarte en otra máquina por ssh sin que te pida la contraseña, necesitamos crear una clave que copiaremos a la máquina destino para validarnos.

Las dos máquina necesitan tener instalada una versión de openssh

Para conectarnos haríamos lo siguiente:

- En la máquina origen hacemos

```
ssh-keygen -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/home/username/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/username/.ssh/id_rsa.
Your public key has been saved in /home/username/.ssh/id_rsa.pub.ssh-
keygen -t rsa
```

.

Dejar el nombre fichero que dice y no poner passphrase.

- Copiamos las claves a la máquina destino

```
 ssh-copy-id usuario@ipdestino
id@server's password:
```

* Si no funciona, en /etc/ssh/sshd_config debería estar descomentado y activado: RSAAuthentication yes y PubkeyAuthentication yes.

```
ssh usuario@maquina
rsync –e ssh –vrclHptogt –delete –force –stats –progress
servidor:/var/lib/prueba
```

ssh en nautilus

Cuando estamos trabajando en otra máquina a través de una conexión ssh, ¿no es una lata tener que andar copiando ficheros con scp?

Una alternativa mucho más cómoda es iniciar sesión en dicha máquina con nuestro navegador de archivos y arrastrar y soltar con el ratón. Para ello, en la barra de direcciones de nautilus escribiremos:

```
sftp://usuario@direccion_ip:puerto/directorio
```

Por ejemplo:

```
sftp://lc@192.168.2.1:5566/home/lc
```

Al darle a enter nos solicitará la contraseña y accederemos a la máquina. Si usamos el puerto por defecto para las conexiones ssh, y el usuario tiene permisos de lectura en la carpeta raíz, podemos obviar los datos puerto y directorio.

Copiar ficheros

Desde local a remoto

```
 scp fichero_origen usuario_remoto@ip_ordenador_remoto:fichero_destino
```

Desde servidor remoto a local

```
scp usuario_remoto@ip_ordenador_remoto:/fichero_origen fichero_destino
```

Para copiar carpetas completas, sólo hace falta agregar el parámetro **-r**

Referencias

- http://blogofsysadmins.com/crear-un-proxy-socks-usando-un-tunel-ssh
- http://blogofsysadmins.com/secure-shell-hacks-linux
- http://submarley.espacioblog.com/post/2008/11/04/shh-dios-la-administracion-remota
- http://linuxamartillazos.blogspot.com/search/label/ssh
- http://www.vicente-navarro.com/blog/2009/05/24/creando-tuneles-tcpip-port-forwarding-con-ssh-los-8-escenarios-posibles-usando-openssh/
- http://terminus.ignaciocano.com/k/2011/08/12/utilizar-ssh-para-establecer-un-servidor-proxy-so

cks/
- http://hpantaleev.wordpress.com/2012/02/14/openssh-sftp-chroot-con-chrootdirectory/
- http://bootlog.org/blog/linux/tip-ssh-scp-y-un-as-bajo-la-manga
- http://www.alcancelibre.org/staticpages/index.php/como-ssh-clave-publica
- http://es.softuses.com/28144
- http://jessenoller.com/2009/02/05/ssh-programming-with-paramiko-completely-different/
- http://ubuntulife.wordpress.com/2011/02/16/script-mejorando-la-productividad-automatizando-conexiones-ssh-mediante-expect/
- http://ubuntulife.wordpress.com/2010/08/03/sencillo-script-en-bash-para-ejecutar-un-comando-remoto-mediante-ssh/
- http://www.alcancelibre.org/staticpages/index.php/como-ssh-clave-publica