Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Port forwarding con SSH (35929 lectures)

Per **Guillem Cantallops Ramis**, ***Beowulf*** (http://bulma.net/beowulf/)

Creado el 21/01/2002 23:57 modificado el 21/01/2002 23:57

Vamos a ver como conseguir conexiones TCP seguras aunque tengamos que pasar por una red insegura. Se trata de una situación muy frecuente, y se puede resolver de muchas maneras, algunas más transparentes y más "limpias" (a nivel IP, por ejemplo), pero ninguna tan sencilla como esta que usa el SSH de toda la vida...

Todos sabemos como funciona SSH: en pocas palabras es como telnet pero seguro. O no? En realidad ofrece muchas más posibilidades...

Supongamos que queremos conectarnos a un servidor de correo, o que simplemente queremos navegar, pero la parte de la red que tenemos más cerca es insegura. Por ejemplo, nos estamos conectando mediante IEEE 802.11b (*Wireless LAN*) y no terminamos de fiarnos del WEP (*Wired Equivalent Privacy*). O nos conectamos con un ISP cualquiera, o desde la Universidad, pero nuestra paranoia nos hace pensar que la seguridad tradicional (casi nula) no es suficiente.

Supongamos también que tenemos acceso mediante SSH a una máquina de confianza, una máquina en una parte "segura" de la red, o por lo menos más segura que la parte que nos preocupa. Puede ser la máquina a la que nos queremos conectar para leer el correo, o el proxy que usamos para navegar, o la máquina que hace de gateway entre la red inalámbrica y la red "alámbrica"....

En ese caso, podemos tender un "puente", o mejor dicho, un "túnel" mediante SSH entre nuestra máquina y la máquina "segura", salvando los peligros de la parte insegura de la red. En cierta manera, conseguiremos "acercarnos" a la máquina segura. Ya, vale, eso ya lo hacia el SSH de toda la vida... pero sólo nos daba un shell, que sirve para muchas cosas pero no para todo. Veamos como aprovechar la conexión SSH, veamos como meter ahí dentro otras conexiones TCP para usar el servidor POP3 o el proxy remoto con seguridad, como si fuera local.

Para esto se utilizan las opciones -L, -R y -D de SSH. Las opciones -L y -R son simétricas (no, no son *left* y *right*, sino *local* y *remote* `;-)`, y la -D tiene un uso bastante marginal, así que veremos la opción -L.

La sintaxis completa para usarla es la siguiente:

```
ssh -L puerto_local:maquina_destino:puerto_destino maquina_segura
```

Vamos por partes. Lo que está en negrita es lo nuevo, lo demás es lo que haciamos siempre. Con `ssh maquina_segura` conseguiamos un shell en la máquina segura. Ahora vamos a ver como funciona la parte divertida.

El `puerto_local` especifica un puerto TCP de la máquina local (odio decir estas cosas, pero es así de simple), y evidentemente la `máquina_destino` y el `puerto_destino` especifican justo lo que indica su nombre. La idea es que mientras tenemos la conexión SSH de toda la vida, si nos conectamos a `localhost:puerto_local`, esa conexión TCP será redirigida a través del túnel SSH (con la seguridad correspondiente) a la máquina remota, la `maquina_segura`, y desde ahí se establecerá una conexión a `maquina_destino:puerto_destino`. Es muy importante tener en cuenta que `maquina_destino` se resolverá en la `maquina_segura` así que si ponemos ahí `localhost` convertiremos automáticamente la `maquina_destino` en la `maquina_segura`.

Bién, ahora que lo he liado todo lo suficiente como para que hayan huido todos excepto los cuatro locos suficientemente interesados en esto, os pongo varios ejemplos `;-)`

- `ssh -L 8080:localhost:3128 miproxy.segu.ro`: Suponiendo que tengamos acceso SSH a miproxy.segu.ro, y que esta máquina esté ejecutando un proxy (por ejemplo el squid) en el puerto 3128, con este comando conseguimos (además del shell de siempre) que todas las conexiones al puerto 8080 local se vayan por el túnel SSH al puerto 3128 de miproxy.segu.ro. Así que configuramos el navegador para que use como proxy localhost:8080, y en realidad estaremos usando miproxy.segu.ro:3128, solo que la comunicación irá protegida por SSH.
- `ssh -L 110:mipop3:110 miservidor.segu.ro`: Supongamos que tenemos acceso SSH a miservidor.segu.ro, y una cuenta POP3 en la máquina mipop3, que está en la misma red local que miservidor.segu.ro. O bién no tenemos acceso a mipop3 desde fuera, o bién nos da miedo que nuestro password viaje en pelotas por medio mundo hasta llegar a él. Usando este comando SSH conseguimos que las conexiones al puerto local 110 sean redirigidas a mipop3:110, via miservidor.segu.ro, y la conexión entre la máquina local y miservidor.segu.ro va protegida por SSH.

La versión con `-R` es muy parecida, pero el primer puerto que se especifica es remoto y el host:puerto que vienen después se resuelven en local. Por supuesto la redirección se hace en sentido contrario.

Es decir, si hacemos `ssh -R 6110:localhost:110 miservidor.segu.ro` conseguimos que las conexiones a miservidor.segu.ro:6110 sean redirigidas de forma transparente a través del túnel SSH al puerto 110 local. Así que si en la máquina local tenemos un servidor POP3 funcionando, las máquinas de la parte remota pueden usarlo accediendo al puerto 6110 de miservidor.segu.ro.

Para los curiosos, diré que la opción `-D` simplemente toma un puerto (local) como argumento, y pone ahí un servidor SOCKS4. Es más flexible todavia, pero requiere aplicaciones que lo soporten (creo que Mozilla lo hace), u otras soluciones para que cualquier programa pueda usarlo.

Espero que alguien haya entendido algo. Si no, que ridículo más espantoso... `X'-DDD`

E-mail del autor: beowulf _ARROBA_ bulma.net
Podrás encontrar este artículo e información adicional en: http://bulma.net/body.phtml?nIdNoticia=1147