

Comunicaciones encubiertas en Linux

PASADIZOS SECRETOS

Moviendo datos desde y hacia sistemas Linux por debajo del radar. **POR KURT SEIFRIED**

Todos hemos pasado por eso: Nos conectamos a la red inalámbrica y no funciona. Tratamos de conectarnos por ssh a nuestro servidor y obtenemos un bonito “Falló la conexión”. Tratamos de conectarnos con el servidor de correo a través del puerto 25 usando TLS (*Transport Layer Security*, cifrado) y lo único que alcanzamos a ver es el banner del proxy de correo de nuestro ISP, o peor aún, algún otro fallo de conexión. Pero no todo está perdido, aún podemos navegar la web. Por desgracia, cada vez que escribimos mal una URL, acabamos en la página de búsqueda del ISP, y todo lo que albergue contenidos cuestionables, como los relacionados con el hacking, es bloqueado.

En esta tesitura tenemos dos opciones: buscar un buen libro que leer, o usar software VPN para conectar con un host remoto y saltarnos cualquier tipo de filtrado. De todos modos, hay ISPs realmente malvados (o simplemente incompetentes) que bloquean también el software VPN y SSH con el fin de evitar el libre acceso a Internet a través de sus redes. Es como tratar de evitar que el gato se meta en la caja de cartón: si se le impide a un geek hacer algo, lo más seguro es que ponga más empeño en conseguirlo.

Mientras se pueda pasar algún tipo de dato a un sistema remoto (IPSec, SSH, http, mensajería instantánea, señales de humo...), se podrá usar ese mismo canal para transportar cualquier otra cosa. El truco consiste en encontrar un protocolo de red permitido pero no modificado (o no mucho) al vuelo y con el que se pueda usar algún programa existente que permita crear un túnel de datos sobre él.

Por suerte, los tres protocolos de red básicos, ICMP, DNS y HTTP, casi siempre están permitidos, además de otros muchos, como SSH o la mensajería instantánea. Con algo de suerte, se podrá incluso usar software como SSH mediante redirección de puertos o capacidades de VPN sobre un puerto permitido, como el 80 (http). Si no hay tanta suerte, el ISP puede obligar a pasar a través de sus servidores proxy web y sus propios DNS para acceder a Internet.

A Través de ICMP

ICMP es un estupendo protocolo sobre el que crear túneles de datos debido a que casi siempre está permitido (bloquearlo implicaría romper un montón de cosas) y es capaz de transportar grandes cantidades de datos [1]. Un paquete ICMP se compone de 20 bytes de datos en la cabecera (los

típicos campos de fuente, destino, etc.) y 8 bytes de datos de payload (tipo de mensaje, código, etc), más una cantidad de datos variable. La cantidad de datos enviados con el paquete ICMP normalmente sólo está limitada por el tamaño máximo de paquete de la red en cuestión (generalmente son 1500 bytes para Ethernet), algo que también suele ser cierto en la mayoría de las redes inalámbricas. Como consecuencia, se pueden enviar grandes cantidades de datos sobre paquetes ICMP con muy poca sobrecarga.

Cuando se trata de software para la creación de túneles ICMP, disponemos de un par de opciones, aunque quizá la mejor sea Ptunnel (*Ping Tunnel*), ya que es el que está más actualizado [2]. La instalación de Ptunnel es relativamente sencilla; hay RPMs disponibles de la penúltima publicación cortesía de Dag [3].

Para instalarlo, tecleamos:

```
rpm -Uvh http://dag.wieers.com/
rpm/packages/ptunnel/
ptunnel-0.61-1.rf.src.rpm
cd /usr/src/redhat/
rpmbuild -ba ptunnel.spec
```

Para disfrutar de la versión más reciente de Ping Tunnel se habrá de actualizar el RPM o compilarlo desde las fuentes. Para actualizarlo:

```
wget http://www.cs.uit.no/
~daniels/PingTunnel/
PingTunnel-0.70.tar.gz
tar -xf PingTunnel-0.70.tar.gz
cd PingTunnel
make
make install
```

Dejaremos la actualización de los fuentes RPM como ejercicio para el lector, puesto que, compilarlo desde las fuentes no son más que dos líneas.

La ejecución de Ptunnel no es mucho más complicada. En el lado del servidor (el proxy), basta con ejecutar Ptunnel especifi-

Listado 1: Proxytunnel

```
01 Host proxytunnel.example.org
02 ProtocolKeepAlives 30
03 ProxyCommand /ruta/a/proxytunnel -p
proxy.cliente.com:8080 -u usuario -s contraseña -d
proxytunnel.example.org:443
```

cando opcionalmente un dispositivo de red (-c) y una contraseña (-x). Del lado del cliente, se ha de especificar la dirección del servidor proxy, el puerto local en el que se esperará la llegada de conexiones y la dirección y puerto remotos para la realización de la conexión. En el siguiente ejemplo se da por hecho que el servidor proxy *ptunnel.example.org* está conectado a Internet a través de *eth0*, con un proxy Squid ejecutándose en el puerto 3128 del servidor *squid.example.org* y usando la contraseña *blabla* para asegurar la conexión:

```
Servidor:
./ptunnel -c eth0 -x blabla
Cliente:
./ptunnel -p >
ptunnel.example.org -lp 3128 >
-da squid.example.org -dp 3128 >
-x blabla
```

Ahora sólo le indicamos al navegador web local el proxy web en el puerto 3128 de localhost, y todo el tráfico http será convertido en tráfico ICMP y enviado a *ptunnel.example.org*. Una vez allí, se desempaqueta y envía al servidor proxy web *squid.example.org* y luego a Internet. El servidor squid se puede ejecutar localmente en el mismo servidor que Ptunnel, permitiéndonos saltar completamente cualquier filtrado.

A Través de DNS

Aunque no sea tan fiable como ICMP, DNS es otro protocolo que se puede usar como túnel para datos. Ciertos ISPs envían a una pasarela de pago a los sistemas desconoci-

dos o no registrados. Para ello, responden a cualquier petición DNS devolviendo la dirección IP de dicha pasarela. Otros ISPs usarán simplemente un proxy web transparente para interceptar cualquier petición web y redirigirla a su pasarela de pago (en este caso, probablemente sea posible entunelar los datos sobre DNS).

DNS ofrece varias ventajas sobre ICMP. Aunque el bloqueo de ICMP pueda causar problemas, es posible hacerlo. Por el contrario, al bloquear DNS dejará de funcionar todo. Aunque Ping Tunnel 0.70 ya soporta la transmisión de datos sobre el puerto 53 UDP, en realidad no envía paquetes DNS válidos, por lo que no es posible hacer pasar este tráfico a través de los servidores de DNS. Para ello, hemos de disponer de una conexión de redirección al servidor proxy, en cuyo caso podemos usar OpenVPN o bien OpenSSH sobre el puerto 53.

Como proxy real que encapsule los datos en paquetes DNS válidos, existen un par de opciones disponibles: OzymanDNS [4][5] y NSTX [6]. Por desgracia, el proyecto NSTX no actualiza su código desde 2002, y además se ha de usar necesariamente CVS para descargarlo debido a que los paquetes de fuentes no parecen estar disponibles ya. Aparte de esto y debido a varios problemas de diseño, NSTX es bastante lento. Sin ninguna actualización desde su liberación inicial, OzymanDNS está en cierto modo desactualizado.

A Través de HTTP

La última de las opciones que mostramos consiste en el uso de http o https [7] para la

creación del túnel por el que se enviará el tráfico. Hay bastantes probabilidades de que sea cual sea la red en la que nos encontremos, permitirá las conexiones https salientes. https es mejor que http porque cifra el tráfico, de modo que las probabilidades de que se modifique el código son menores. Al igual que con el software de entunelamiento sobre ICMP, podemos compilar Proxytunnel desde las fuentes o descargar un RMP con las fuentes (o los binarios) [8] [9].

Para usar Proxytunnel, basta con ejecutarlo en el servidor, bien como demonio autónomo o bien desde *inetd*, mientras que del lado cliente, bastará con añadir al cliente OpenSSH como ProxyCommand (Listado 1).

Como se puede apreciar, existen varias opciones disponibles para la creación de túneles para el tráfico de red, dependiendo de la disponibilidad de protocolos. Un poco de configuración previa, puede ahorrarnos un montón de problemas en el caso de encontrarnos en alguna red ajena rota, filtrada o que no funcione por cualquier otro motivo. ■

RECURSOS

- [1] Entunelamiento ICMP, Proyecto Loki: <http://www.phrack.org/issues.html?issue=49&id=6#article>
- [2] Ping Tunnel (Ptunnel): <http://www.cs.uit.no/~daniels/PingTunnel/>
- [3] RPM de Ping Tunnel: <http://dag.wieers.com/rpm/packages/ptunnel/>
- [4] OzymanDNS: <http://www.doxpara.com/?p=51>
- [5] OzymanDNS HOWTO: <http://www.dnstunnel.de/>
- [6] NSTX: <http://savannah.nongnu.org/projects/nstx/>
- [7] Túnel SSH sobre http(s): <http://dag.wieers.com/howto/ssh-http-tunneling/>
- [8] Proxytunnel: <http://proxytunnel.sourceforge.net/>
- [9] RPM de Proxytunnel: <http://dag.wieers.com/rpm/packages/proxytunnel/>
- [10] Artículo sobre redirección del puerto SSH: <http://magazine.redhat.com/2007/11/06/ssh-port-forwarding/>
- [11] VPN OpenSSH Layer 3: http://www.debian-administration.org/article/Setting_up_a_Layer_3_tunneling_VPN_with_using_OpenSSH

Redirección del puerto SSH y Funcionalidad VPN

Prácticamente todos los sistemas disponen de un servidor de SSH. Los clientes de SSH son fáciles de localizar y de usar [10]. Mientras se pueda establecer una conexión TCP directa al puerto 80, se podrá ejecutar un servidor de SSH en dicho puerto (*ListenAddress 10.2.3.4:80* en *sshd_config*, aunque hay que estar seguros de definir también *Port 22* si se quieren aceptar además las conexiones en el puerto 22 como de costumbre) y

habilitar la redirección de puertos (*AllowTcpForwarding* definido como *yes* en el archivo *sshd_config*). `ssh -L 8080:www.example.org:80 usuario@servidor-de-ssh`
Otra opción disponible con SSH consiste en usar sus capacidades para VPN. La ventaja de esto último es que se puede enrutar fácilmente todo el tráfico sobre la conexión (Mensajería Instantánea, BitTorrent, etc.) [11].