GNU/Linux, software libre para la comunidad universitaria

Redes privadas virtuales en GNU/Linux

COLABORADORES

	TÍTULO : GNU/Linux, software libre para la comunidad universitaria	

ACCIÓN	*NOMBRE*	*FECHA*	*FIRMA*
ESCRITO POR	José María Peribáñez Redondo	19 de octubre de 2009	

HISTORIAL DE REVISIONES

NÚMERO	FECHA	MODIFICACIONES	NOMBRE
2.0	18-04-2008		José María Peribáñez Redondo
2.1	18-04-2008		José María Peribáñez Redondo

Índice general

Capítulo 1

Distribuciones GNU/Linux para servidores de red

A lo largo del curso, la distribución elegida ha sido OpenSuSE. Esta distribución, al igual que otras populares como Ubuntu, Mandriva o Fedora son interesantes para uso personal, pero no son la mejor elección para un servidor de red, un cortafuegos o una VPN en una empresa. En general para empresas son más adecuadas las distribuciones de ciclo largo, que tengan actualizaciones durante años y no es preciso actualizar a otra versión al cabo de unos meses. Las distribuciones de ciclo largo no tratan de ofrecer al usuario el tener siempre el software más reciente, sino estabilidad.

Para quien desee una distribución de ciclo largo, Novell tiene su propia versión de SuSE adecuada para estos menesteres, pero es software privativo por el que hay que pagar una subscripción anual por cada máquina. Algo parecido (aunque técnicamente no sea software privativo sino libre, gracias a lo cual existen clónicos gratuitos como CentOS) ocurre con Red Hat y su versión Enterprise. La alternativa a estas distribuciones de pago es usar Ubuntu LTS o un derivado de Red Hat Enterprise gratuito, siendo la alternativa más conocida CentOS.

Para instalar un cortafuegos, servidor DHCP etc también está la opción de usar una distribución de propósito específico como IPCop o Smoothwall. Ambas tienen una intuitiva interfaz web. No sólo incluyen el cortafuegos, también gestión del ancho de banda, servidor DHCP, DNS dinámico, servidor de tiempo (NTP), servidor SSH y VPN (con IPsec) admitiendo la VPN tanto unir dos redes como configuración road-warrior (conexión de un usuario remoto a la red local de la empresa).

Smoothwall Express está financiado por una empresa, que comercializa software de seguridad relacionado, incluyendo también cortafuegos y filtros de contenidos; periódicamente smoothwall contribuye código de sus productos privativos a esta versión enteramente libre (por ejemplo el código para mostrar el uso de ancho de banda de forma gráfica). En el caso de IPCop no hay una empresa detrás del producto, pero en su web hay una relación de empresas que venden soporte técnico. Tanto IPCop como Smoothwall soportan características avanzadas como gestión del ancho de banda de las conexiones peer-to-peer (Emule, Bittorrent...) e integran un IDS (detector de intrusiones).

IPCop y Smoothwall son soluciones para un cortafuegos; no están diseñadas para ejecutar todo tipo de servidores, por ejemplo el correo o los servicios de ficheros e impresión. No es buena idea poner un gran número de servicios sobre un cortafuegos y aprovechar una máquina como servidor para todo. Cuantos más servicios críticos corran sobre una máquina, mayor riesgo es que exista alguna vulnerabilidad en uno de ellos que al comprometer la máquina afecte a la seguridad de todos los demás.

Probablemente la distribución especializada en redes y con administración web más interesante sea Zeroshell, con características muy avanzadas por ejemplo a la hora de construir un Acess Point: http://www.zeroshell.net/

Hay otras distribuciones orientadas a servidores de empresa (sobre todo a PYMES). Destacaremos tres:

1. Ebox (http://ebox-platform.com/). Software libre de una empresa española, con toda una arquitectura para desarrollar los nuevos componentes. En realidad no es una distribución, sino una plataforma que se instala sobre una distribución: por ejemplo está disponible en los repositorios para Ubuntu. De todos modos se puede descargar una distribución basada en Ubuntu 8.04 LTS con todo ya instalado.

2. SMEServer (www.smeserver.org/). GPL, basado en CentOS 4. Servidor de correo, con filtro de SPAM y virtus, conversión de correos en formato TNEF, webmail. Servidor de ficheros, con sistema de administración de usuarios, cuotas. Servidor web. Compartición de la conexión a Internet. I-Bays: Un sistema para compartir información entre un grupo de usuarios. Servidor de impresión.

3. ClarkConnect: (http://www.clarkconnect.com/). Servidor de correo, impresión, ficheros, backup, VPN... Lamentablemente no parece ser software libre al 100 % , porque hay distintas ediciones. La edición gratuita además de tener limitaciones no es de ciclo largo.

Es muy recomendable acudir a las páginas de estos proyectos en la Wikipedia, porque además veremos referencias a proyectos similares y a herramientas de administración web que se pueden integrar en las distribuciones habituales

También cabe reseñar algunas soluciones basadas en Linux que corren en routers de precio asequible, como el SoC (System On Chip) ADM5120, un hardware que cuesta menos de 50EUR, tienen un consumo de energía mínimo y el tamaño de un switch barato para el hogar. Sobre estos sistemas corren distribuciones Linux específicas como OpenWRT (www.openwrt.org/. Se puede adquirir un router basado en una placa ADM5120 con 5 interfaces de red, 16MB de RAM y la opción de usar discos externos por USB (lamentablemente limitados a la velocidad de USB 1.0) en http://www.omnima.co.uk/store/catalog/Embedded-controller-p-16140.html. Si lo que necesitamos es un servidor y no necesitamos tener varias interfaces de red, hay otros dispositivos como http://www.dealextreme.com/details.dx/sku.20383 o http://bifferos.bizhat.com/. Soluciones más caras son las placas de Soekris.

Para evitar la dependencia de OpenSuSE y facilitar al lector elegir la distribución que más le convenga para sus servidores, en los capítulos sobre redes no se usará la herramienta de administración Yast, sino que se indicará como hacer la configuración manualmente o bien se comentarán herramientas genéricas disponibles en cualquier distribución.

Capítulo 2

Paquetes de red. Modelo en Capas

Para entender los conceptos avanzados de redes, es muy útil tener presente el modelo en capas de las redes. Cada capa da servicio a la capa superior implementando una funcionalidad. A la hora de enviar un paquete, la capa superior pasa a la que está debajo los datos que quiere enviar, quien hará lo propio con la capa inmediatamente inferior, tras añadir sus propias cabeceras. Al recibir un paquete los pasos son los inversos, cada capa tras interpretar y eliminar sus cabeceras pasará el resto de datos a la capa inmediatamente superior.

1. Layer 1: capa física. Nos basta saber que existe: se implementa enteramente en hardware. No nos detendremos más en ella. Es a partir de la capa 2 dónde podemos ver los paquetes con un sniffer.
2. Layer 2: capa de enlace de datos/acceso de red: Depende del tipo de red local o comunicación punto a punto que utilicemos: por ejemplo si es una Ethernet, una FDDI (red metropolitana de fibra óptica, muy usada para unir los centros de las facultades o los edificios en un parque tecnológico), una conexión PPP sobre una línea telefónica... En la actualidad un administrador de red raramente maneja otro enlace de datos distinto de Ethernet, pues aunque se trate de un router que conecta con otros routers vía FDDI, normalmente se usa un transceptor de fibra al que la conexión es vía Ethernet y la comunicación es realmente a nivel Layer3. Lo mismo ocurre con el enlace punto a punto utilizado para el ADSL, podrá usar PPPoA o PPPoE (un poco más eficiente usar PPPoA), pero lo normal es que eso sea en el router ADSL y la comunicación hasta el router ADSL es vía ethernet y de nuevo a nivel IP (layer3), aunque cuando el router ADSL está en modo monopuesto sí se suele utilizar un cliente PPPoE. A nivel usuario, esta capa puede presentar la dificultad de configurar un acceso Wifi o una conexión Bluetooth. La capa 2 es la más alta que ven los bridges, switchs y hubs.
3. Layer 3: capa de red. La capa IP. Es la capa más alta que ven los routers. Sobre una red Ethernet puede haber más de una red IP, si los rangos de direcciones no se solapan o incluso pueden mezclarse distintos tipos de redes, por ejemplo una red IP con una AppleTalk o IPX (Netware).
4. Layer 4: capa de trasporte: TCP,UDP... En esta capa y en la 5, sólo suelen intervenir el nodo origen y destino, aunque se puede ver afectado por ejemplo por un router que haga NAT. Es la capa con la que suelen trabajar los aplicaciones, a través de sockets. En Unix los sockets se manejan como ficheros.
5. Layer 5: capa de aplicación: HTTP, MAIL, JABBER... En ocasiones esta capa se menciona como l7 (layer7) porque en el modelo de referencia OSI hay otras dos capas entre la capa de transporte y la de aplicación: sesión y presentación.

La mayoría de las aplicaciones utilizan protocolos IP y para ellos las capas 1 y 2 son transparentes, como si no existieran. Sin embargo hay protocolos que se implementan directamente sobre la capa2.

A veces sobre la capa de red, la de transporte o incluso sobre la capa de aplicación se encapsula otro paquete desde la capa de red o incluso desde la física. El motivo es hacer túneles y se usa mucho en las redes privadas virtuales: se hace que dos nodos que no están en la misma red local se comporten como si lo estuvieran, gracias a que hemos cargado los paquetes sobre otros paquetes que van por Internet y en destino los reconstruimos.

2.1. Capa enlace de datos (Layer2)

Aunque la mayoría de las aplicaciones son independientes de esta capa al utilizar protocolos IP, hay excepciones, por ejemplo muchos juegos operan sobre layer2.

Tanto en una red Ethernet como en una Wifi, cada tarjeta tiene una dirección que es única, la dirección MAC (es como un número de serie que no se repite, para ello cada fabricante tiene asignado un rango de direcciones, aunque también existen rangos para uso privado). Son 48 bits, o lo que es lo mismo, 12 caracteres hexadecimales. Suelen representarse así: 00:09:6C:EF:E4:69.

Las direcciones MAC al ser únicas en principio se pueden usar para filtrar paquetes en un cortafuegos o para restringir el acceso en un Access Point Wifi. Error. Lo más probable es que nuestra tarjeta nos permita cambiar la dirección ARP con un simple ifconfig eth0 hw ether 00:09:6c:EF:e4:69. Este cambio es temporal, dura hasta que lo volvamos a cambiar o reiniciemos la máquina. Además en muchas tarjetas se puede hacer el cambio permanente, pues la dirección MAC se escribe en una memoria flash dentro de la tarjeta. En este caso la herramienta a utilizar se llama ethertool y la opción que cambia la MAC es phyad.

El hardware de las tarjetas de red y de los hubs no sabe nada de direcciones IP, lo único que entiende es el formato de los paquetes ethernet, con sus direcciones MAC.

Es un error pensar que las tramas Ethernet se generan en el hardware o en el sistema operativo sin posibilidad de manipulación. En GNU/Linux mediante los sockets RAW (ver man 7 packet) es posible escribir tramas Ethernet. Así mismo mediante TUN/TAP tanto en GNU/Linux como en la mayoría de los sistemas operativos es posible leer/escribir a través de un dispositivo tanto tramas ethernet como paquetes IP. Si en nuestra red existe un túnel (por ejemplo para hacer una VPN) es posible generar paquetes Ethernet en un equipo remoto, enviarlos vía Internet y que un nodo en la red local los emita como si hubieran sido generados localmente.

En una red Ethernet para unir los distintos equipos se usa un hub o un switch. Un hub opera a nivel físico y es muy primitivo: no deja de ser el equivalente en conectores de red a un "ladrón" en clavijas para enchufes. También se le compara con un repetidor. El problema de los hubs es que este modelo es poco escalable cuando existen muchos equipos en una red (especialmente si se enlazan varias redes locales físicamente para no usar enrutadores, simplificando la configuración de la red y que funcionen servicios que usan broadcast a nivel Ethernet como DHCPD (en realidad también se puede tener un único servidor para varias subredes no conectadas a nivel de enlace de datos si se usa un bootp/DHCPD agent relay), o esté disponible funcionalidad como wake-on-lan o servicios de autoconfiguración PnP). Un hub es poco escalable porque lo que se hace es transmitir para cada nodo todos los paquetes.

La alternativa a los hub son los switchs. La diferencia entre un hub y un swtich es que el segundo analiza las tramas ethernet que pasan por él; si ve que un paquete ha entrado por un puerto (cada conector para un cable de red es un puerto) registra la dirección MAC de origen junto con el puerto en una tabla llamada CAM, de modo que si llega un paquete para esa dirección no lo envía por todos los puertos sino sólo por ese. Hoy en día si vamos a una tienda a comprar un hub, lo normal es que ya no lo tengan y sólo vendan switchs.

En una red local Ethernet pueden coexistir paquetes de distintos protocolos, por ejemplo tráfico IP y tráfico AppleTalk. También es posible que sobre una misma red Ethernet haya más de una subred IP: si los rangos no se solapan tampoco habrá problemas.

En redes de capa2 existe la posibilidad de crear varias VLAN (virtual LAN: 802.Q), que consiste en añadir a los paquetes Ethernet una cabecera adicional que indica la LAN virtual que se aplica: un switch que soporte VLAN (los baratos no lo hacen, pero sí lo soporta por ejemplo los switchs de los AP que usan OpenWRT; en cualquier caso tiene que ser un switch administrable de alguna forma) sólo transmitirá por un puerto los paquetes que vayan a esa VLAN (esta asociación la hace y deshace el administrador). Se puede configurar un puerto para recibir paquetes destinados a más de una VLAN (trunk port). La VLAN marcada como "nativa" para un puerto implica que los paquetes que van para esa VLAN se les quita el "tag" que indica la VLAN, por lo que serían paquetes Ethernet normales.

Mediante una VLAN los paquetes broadcast sólo son a nivel de esa VLAN y sería posible que hubiera dos redes usando el mismo cableado con rangos de Ips solapadas, si usan distintas VLANs. Podemos ver las VLANs como una forma de tener más de una red de capa2 sobre una única infraestructura de cableado (es decir, multiplexar varias redes de nivel 2 sobre una). Las VLANs funcionan no sólo sobre una red local "física", sino sobre toda una infraestructura de redes unidas por swtichs en un campus, por ejemplo.

Típicamente se hacen coincidir las VLANs con las subredes IP de capa 3. De este modo se fuerza a usar enrutamiento para comunicarse entre nodos de distinta subred. Un ejemplo de uso de VLANs es la infraestructura de red de una empresa, dónde es necesario aislar las redes de distintos proyectos, pero sin que haga falta relocalizar los ordenadores de los programadores y de los servidores

Hay un identificador reservado para paquetes que no pertenecen a ninguna VLAN, el cero. El uno también está reservado, para gestión

En GNU/Linux el comando para crear redes VLAN es vconfig.

2.1.1. WDS

A la hora de tratar de hacer un bridge entre una tarjeta Ethernet y una tarjeta Wireless nos podemos encontrar con que no funciona. El motivo es que para que funcione un bridge los paquetes Wifi tienen que incluir hasta 4 direcciones: la dirección MAC origen, la dirección MAC de la tarjeta Wifi, la dirección MAC del Access Point, la dirección MAC destino. Si la tarjeta (más que su hardware el driver no tiene en cuenta estas 4 direcciones sino 3 (todas menos la origen, al suponer que la origen es la de la tarjeta) no funcionará. Lo mismo ocurrirá si el AP no reconoce las 4 direcciones. Con un bridge que sólo afecta a tarjetas Ethernet no hay este problema: sólo hay dirección origen y destino, las tarjetas del bridge se ponen en modo promiscuo para recibir todos los paquetes y no sólo los de su MAC, por lo que la dirección de las tarjetas que forman el bridge no van en el paquete al no hacer falta.

Para que una conexión Wifi pueda participar en un bridge, la tarjeta por lo tanto tiene que soportarlo y el AP también: los AP que soportan WDS son aptos. WDS (Wireless Distribution System) permite implementar un bridge entre dos AP ampliando la cobertura al alcance de los dos.

2.1.2. IEEE802.1X

Es un sistema para autenticar un puerto (físico) en un dispositivo de red, por ejemplo lo ofrecen algunos switch avanzados y los Access Point Wifi. La idea es que el cliente (por ejemplo un PC) que se conoce es el supplicant y el dispositivo de red es el authenticator que tiene que autorizarle. Típicamente el authenticator contacta con un servidor Radius[1]. Para autenticar se utiliza AEP.

Windows incluye software supplicant y en el caso de Linux hay dos programas: Xsupplicant, que sirve para todo tipo de hardware y wpa-supplicant, que es específico para Wifi.

Autenticarse mediante IEEE802.1X no evita todos los ataques posteriores derivados de insertar un hub entre la máquina y el switch, pues la autenticación sólo es al principio: sería conveniente usar además IPsec (en el caso del Wifi la conexión tras autenticarse va cifrada luego ya sería segura).

2.1.3. WEP/WPA/WPA2/IEEE802.11i/

En las redes Wifi sobre la capa 2 se implementó cifrado mediante WEP. Sin embargo este método se ha demostrado muy inseguro.[2]. La solución de compromiso utilizando el mismo hardware (cifrado de flujo con RC4) es WPA, que cambia dinámicamente la clave (TKIP) e introduce varios mecanismos de seguridad. WPA2 (IEEE802.11i) usa AES para cifrar en lugar de RC4 (no es compatible pues con el hardware viejo) e introduce algunas otras mejoras. Tanto WPA como WPA2 admiten dos configuraciones: personal y enterprise. En personal (WPA/PSK) se usa una clave compartida, que será la misma en todos los nodos, por lo que es una solución válida para protegerse de intrusos externos, pero no frente a ataques de personas con un nodo en la red. La versión empresarial en cambio utiliza para autenticarse (y para obtener la clave de sesión) IEEE802.1X. Tanto si se usa la configuración personal como la empresarial, el programa a utilizar es wpa_supplicant (para depurar posibles errores cabe conectarse a ese demonio con wpa_cli así como observar con dmesg los mensajes del kernel,[3] por ejemplo podemos deducir que es necesario

[1] Radius es un protocolo definido para hacer AAA (Authentitacion, Authorization and Accounting) usado sobre todo para los ISP. Se utiliza usuario y contraseña, protegida con MD5 y un secreto compartido entre el servidor Radius y el NAS (Network Access Server, en este caso el authenticator, por ejemplo el Access Point, pero en el escenario más clásico es el servidor PPP del ISP), que es el cliente Radius. De todos modos hoy en día los servidores Radius soportan sistemas de autenticación distintos de usuario y contraseña para autenticar, en concreto AEP con métodos como AEP-TLS que implica certificados cliente y que la comunicación entre el supplicant y el servidor Radius se haga sin que el authenticator tenga que soportar cada uno de estos métodos AEP. La secuencia clásica de Radius es que el usuario se autentique con su ISP usando PPP y el servidor PPP contacte con el servidor Radius pasándole las credenciales que ha recibido del usuario vía PPP. El servidor puede asignar datos como una IP o rango de IPs, así como parámetros L2TP o QoS (en realidad cualquier tipo de información, pues como los servidores DHCP permite extensiones de vendedor) . Las funciones de Accounting básicamente consisten en el que el NAS pasa información a registrar como duración de la sesión o tráfico transferido.

[2] Existen varias aplicaciones para automatizar el conseguir una clave WEP, como el ya antiguo airsnort, si bien hoy en día hay programas más avanzados como aircrack-ng.

[3] En algunos casos puede resistirse una configuración WPA. Una página que muestra gran cantidad de opciones posibles es http://manual.sidux.com/es/internet-connecting-wpa-es.htm. Sobre WPA Enterprise y el uso de EAP, algunas referencias interesantes son

ejecutar iwpriv eht1 host_roaming 1). En Ubuntu utilizar WPA es tan sencillo como incluir en /etc/network/interfaces unas líneas como estas:

```
iface eth0 inet dhcp
wpa-ssid MIWLAN
wpa-key-mgmt WPA-PSK
wpa-passphrase mipassphrase_1asfPasfd
```

WPA a día de hoy previene los ataques más problemáticos que existen sobre RC4, aunque ya existen ataques que lo hacen no seguro y probablemente se descubran nuevas vulnerabilidades en el futuro, que haga que si alguien almacena hoy tráfico WPA pueda descifrarlo dentro de a lo sumo unos años. WPA/PSK no es sólido cuando las claves de los usuarios son débiles o están en un diccionario; no facilita la prevención de este tipo de ataques. Las vulnerabilidades existentes actualmente de WPA no permiten robar ancho de banda, pero sí provocar problemas inyectando paquetes[4].

2.1.4. Wakeonlan

Una característica interesante que permiten la mayoría de las tarjetas de red actuales es encender el equipo remotamente: wake on lan. Es útil para tareas de mantenimiento y también para poder conectarse al equipo desde casa en caso de necesidad, por ejemplo mientras se está de soporte. Esta funcionalidad se implementa mediante un paquete a nivel de enlace de datos.

Para que funcione esta funcionalidad, normalmente hay que activarla en la BIOS. En algunas tarjetas de red, también hay que unir un pequeño cable que sale de la tarjeta a un conector en la placa base que pone "wake on lan". Dependiendo del caso, también es posible que tengamos que activar la funcionalidad en la tarjeta, para eso utilizamos la herramienta ethtool. Con ethtool <nombre_interfaz> (por ejemplo ethtool eth0) veremos los distintos modos wake-on-lan que soporta la tarjeta, si es que soporta alguno. Con **`ethtool eth0 -set wol g`** activamos el modo wake-on-lan basado en "magic packet", el modo más habitual. Se trata de un paquete que en alguna parte debe contener un patrón consistente en primero 6 bytes con todos sus bits a 1, seguido por la dirección MAC de la tarjeta repetida 16 veces. Opcionalmente continua con una password, que también se fija con ethtool pero que sólo admiten algunas tarjetas.

El paquete normalmente se construye usando UDP al puerto 9 (discard) y enviando el paquete por broadcast, aunque puede ir encapsulado en cualquier tipo de paquete, ni siquiera tiene que ser IP.

¿Es posible despertar a un equipo mediante un "magic packet", enviándolo desde fuera de la red local? normalmente no, salvo que tengamos una infraestructura de VPN. El problema es cómo hacer llegar el paquete hasta la máquina; si está conectada a un switch sólo recibirá los paquetes que van destinadas a su MAC o paquetes broadcast/multicast. Es bastante frecuente que los enrutadores no dejen pasar los paquetes broadcast. Pero si intentamos la otra opción, enviar el paquete a su IP, al llegar al enrutador posiblemente ya no tenga la entrada ARP, pues caducan tras un tiempo; cuando haga la petición ARP no obtendrá respuesta, pues la máquina está apagada. Una forma de evitarlo es insertar estáticamente la entrada en la tabla ARP.

Los dos programas más recomendables son wakeonlan y etherwake. Etherwake genera directamente un paquete Ethernet, por lo que requiere privilegios de root y sólo sirve para ejecutarlo en la propia red local.

2.2. Capa de red: capa IP (Layer3)

Sobre la capa de enlace de datos va la capa IP; en realidad pueden ir otro tipo de capas de protocolos distintos a IP, como IPX, el protocolo de Netware, pero nos centraremos sólo en los paquetes IP que son los que se usan en la casi totalidad de las redes locales de hoy en día además de por supuesto en Internet.

Un paquete IP se llama datagrama. La información más importante que incluye un datagrama es la IP de origen y la IP destino.

Las direcciones IP a diferencia de las MAC están pensadas para ser enrutables, es decir, hay unos nodos enrutadores que examinan la dirección destino y deciden por qué camino siguen los paquetes. Gracias a que las direcciones IP se asignan por grupos a las distintas organizaciones y proveedores de Internet es factible establecer rutas. Esto con las direcciones MAC sería imposible: como vienen a equivaler a números de serie del hardware son inútiles para enrutar: puedo tener dos equipos juntos con MACs

[4]El antes citado aircrack-np no se usa sólo para WEP, pues también se usa con redes en las que se conoce la clave WPA/PSK o se trata de atacar si está en un diccionario. No sólo utiliza el modo monitor de las tarjetas sino que "inyecta" paquetes para facilitar la tarea de romper la seguridad. Aunque en general aircrack es un programa que parece dedicado a la gente que desea colarse en las redes ajenas, su utilidad airtun-ng sí es aprovechable en redes con WPA/PSK para usar un IDS. La web es interesante por la documentación sobre seguridad en redes Wifi.

que no tienen nada que ver porque son de fabricantes distintos o uno tiene años más que el otro: quizás el nodo con la siguiente MAC a la nuestra se vendió en una ciudad de Japón.

Hay rangos de direcciones IP que están reservados para uso privado dentro de una organización. Por ejemplo todas las direcciones que empiezan por 192.168.[5]. En las empresas, o en los hogares cuando se tienen varios ordenadores en casa y se quiere compartir la ADSL, lo habitual es usar direcciones de este tipo (privadas) en lugar de direcciones IP públicas. Las direcciones privadas no son enrutables fuera del ámbito en que se usan, por lo que no son utilizables para conectarse fuera de la red local. Por ejemplo no podemos conectarnos vía mensajería instantánea a un usuario de Internet utilizando la IP 192.168.120.7, porque los paquetes de respuesta no nos llegarían, dado que esa IP no es pública y no es pública porque puede haber mucha gente que la está usando en su red local. La solución está en el uso del NAT (Network Address Translator). El enrutador de salida a Internet de nuestra red modifica el paquete IP para que use como dirección IP origen la IP pública que tiene asignada la empresa y guarda en una tabla información para hacer la modificación inversa en los paquetes de respuesta entrantes.

Siempre que hay enrutado y esto incluye casos especiales como tener una VPN en un nodo, hay que activar el forwarding. Para ello hay que ejecutar **`sysctl net.ipv4.ip_forward=1`**. El forwarding consiste simplemente en que si un paquete viene por una interfaz de red (por ejemplo eth0) y se ve por la tabla de rutas que está destinado a la red de otra interfaz (por ejemplo la eth1) se traspasa a esa otra interfaz. También es posible activar el forwarding para unas interfaces de red sí y para otras no, por ejemplo para eth0 sería **`sysctl net.ipv4.conf.eth0.forwarding=1`**.

Hay una parte de la dirección que indica el nodo dentro de la red pero otra indica la red. Los enrutadores tienen rutas para saber por dónde tienen que encaminar los paquetes según la parte de red destino, con una ruta por defecto para los paquetes que no encajan en ninguna de las otras rutas. Los enrutadores sólo suelen tener en cuenta la dirección destino, no la dirección origen, pero en GNU/Linux hay herramientas avanzadas que permiten considerar también la dirección origen.

2.2.1. Máscaras de red. Notación CIDR

Es importante conocer no sólo la IP de un equipo, sino su red. Para determinar la red a la que pertenece un equipo a partir de su IP, se usa la máscara de red, que al aplicar a nivel de bits un AND sobre la IP nos da la red. Una notación más concisa que utilizar el par IP y máscara de red es la notación CIDR: tras la dirección IP, separado por / se indica el número de bits a uno de la máscara de red. Por ejemplo, 192.168.120.19/24 indica que la máscara de bits es de 255.255.255.0. Esta notación además de más concisa tiene la ventaja que es más rápido de ver cuantos nodos tiene una red, pues es 2^(32-x)-2; en nuestro caso sería 2^(32-24) = 2^8-2= 254.

La máscara de red más grande (es decir, la red con menos nodos) es 30, con lo que tendría 2^2-2=2. El motivo de restar 2 unidades es que el primer valor (con todos los bits a cero) es la propia red y el último (con todos los bits a 1) es la dirección de broadcast, es decir, la utilizada para enviar un paquete a todos los nodos de una red. Así pues, una red de este tipo es útil para definir una red para una conexión punto a punto, por ejemplo para una conexión que se usa únicamente para enlazar dos redes.

2.2.2. Enrutamiento dinámico

Hay protocolos de frontera interior como RIP, OSPF, IS-IS y protocolos de frontera exterior, como BGP[6]. El programa recomendado es o bien Zebra o Quagga (este último es un fork surgido de Zebra). Hay otros programas, como Routed (parte de netkit), que sólo es adecuado si no vamos a utilizar más que RIP, Gated (que no es libre) o Bird, con desarrollo menos activo. Un artículo algo antiguo, pero que puede servir de introducción a los protocolos de enrutamiento, se publicó en Linux Magazine: http://www.linux-magazine.com/issue/31/Zebra.pdf

2.2.3. Paquetes ICMP, IGMP

Los paquetes ICMP (Internet Control Message Protocol) son paquetes de control, que van sobre la capa IP pero siguen siendo parte de la capa de red. Por ejemplo son los paquetes que se reciben cuando se recibe un error de que no se ha podido establecer una conexión a un puerto o que no hay ruta para llegar a una red. Son también los paquetes que se envían y reciben al hacer un

[5] En concreto, las redes reservadas para uso privado están fijadas por el RFC 1918 y son 10.0.0.0/8, 172.16.0.0/12 (es decir, de 172.16.0.0 a 172.31.255.255 y 192.168.0.0/16. Hay más rangos reservados para usos especiales, por ejemplo 169.254.0.0/16 para link-local o 224.0.0.0/4 para Ipv4 multicast.

[6] Protocolos de frontera interior son los que se usa para enrutar el tráfico dentro de la organización que tiene asignado un rango de Ips y es responsable cara a Internet de su enrutamiento (típicamente un proveedor de Internet, o por ejemplo RedIris), que es lo que se conoce como un AS (Autonomus System) mientras que los protocolos de frontera exterior son para enrutar entre AS.

ping para comprobar si llegamos a una red. De los paquetes ICMP sólo nos interesa saber que llevan un campo con el tipo de mensaje, que será el campo por el que filtraremos o dejaremos pasar este tipo de mensajes en nuestro cortafuegos. Con iptables -p icmp -h se ven los tipos de paquetes ICMP.

Los paquetes IGMP son paquetes de control para multicast (multidifusión). El origen del modo multicast es que hay comunicaciones en las que los mismos paquetes son de interés de varios nodos destino en lugar de uno solo; el ejemplo más socorrido es una emisión de vídeo. Si se transmite una película usando unicast (el modo normal de comunicación IP) entonces se transmitirá un paquete por cada destinatario, mientras que con multicast en los tramos comunes a todos los destinatarios irá un solo paquete. El multicast se basa en rangos de Ips reservados para este fin, pero no hablaremos más de él porque se usa muy raramente. No hay que confudir multicast con broadcast: este último es enviar a todos los nodos de la red, el multicast es más restringido.

2.2.4. Vinculación entre capa de red y enlace de datos

A nivel de red se utilizan direcciones Ips, pero las redes locales a nivel físico utilizan Ethernet, por lo que en último término para enviar un paquete a una máquina de la red local no basta con utilizar su IP, además en la parte de cabeceras Ethernet habrá que poner su dirección MAC. El protocolo que permite averiguar la dirección MAC de una máquina a partir de su IP es ARP. El funcionamiento de ARP se basa en utilizar una dirección de broadcast Ethernet para preguntar a todos los nodos de la red local quién tiene una determinada IP: la respuesta se cachea en cada host para evitar preguntarla cada vez. Con el comando arp de Unix se puede ver y manipular la tabla que convierte de IP a dirección MAC en una máquina.

Existe un protocolo llamada RARP (reverse ARP) que hace lo contrario que ARP: a partir de la MAC un servidor responde qué Ip tenemos. Se usa para hacer los nodos de la red autoconfigurables, pero se considera preferible usar un protocolo más avanzado como BOOTP o DHCP.

GNU/Linux incluye soporte de ProxyARP. Un proxy ARP es una máquina que responde peticiones ARP en nombre de otras máquinas que estando en la misma red IP que el nodo que hace la petición, no están en la misma red física (ni están unidas por un hub, switch o bridge). El uso más típico es para poner equipos detrás de un cortafuegos, pero usando Ips de la misma red que el resto de equipos. El host que actúa de proxyARP ante una petición ARP a una IP de la que hace de proxy responde con su propia dirección MAC; luego cuando le lleguen paquetes los redirigirá a la máquina. Otro uso importante es para que un nodo que accede a través de una VPN a la red local pueda tener una IP local, sin necesidad de usar un bridge. Otro uso es para el caso de una red Wifi que no soporta WDS y tenemos un nodo que enlaza a varios equipos Ethernet con la red Wifi y los equipos quieren usar direcciones IP válidas en la red Wifi en lugar de NAT para ser accesibles sin tener que hacer mapeos de puertos. Típicamente esto se hace dando la misma IP a la interfaz Wifi y la Ethernet, pero poniendo el enrutado de toda la red por la Wifi salvo los nodos concretos (a los que se accederá en la red Wifi gracias a proxyARP) que se pondrá una ruta concreta usando "dev" a la interfaz ethernet.

En general podemos ver a proxyARP como una alternativa a tener un bridge, para usarlo con tráfico IP y hacerlo con una única IP o unas pocas.

2.2.5. ipalias

Una característica interesante de Linux es que un dispositivo de red Ethernet puede tener más de una IP, puede tener cuantas direcciones queramos. De este modo con una única tarjeta es posible tener varios servidores en el mismo puerto y máquina (por ejemplo varios servidores de correo, que obligatoriamente usan el puerto 25), usando cada uno una IP distinta, sin necesidad de comprar más tarjetas.

Así, si tenemos la interfaz eth0 con la IP 192.168.1.1 y queremos tener otra interfaz con la 192.168.1.2, simplemente creamos:**`ifconfig eth0:1 192.168.1.2`**

Esta funcionalidad se llama ipalias.

No hay restricciones sobre si las Ips tienen que estar en la misma red o en redes distintas: la única restricción es que seguimos teniendo sólo una dirección ARP, pues esto ya es una limitación del hardware de la tarjeta, para que la tarjeta pudiera tener más de una MAC, tendría que ponerse en modo promiscuo, con la consiguiente pérdida de eficiencia: el hardware de la tarjeta está optimizado para filtrar los paquetes que no van a su MAC ni a ninguna de las de difusión.

Técnicamente una interfaz ipalias aunque con ifconfig aparezca como un dispositivo más, no es un dispositivo reconocido por el sistema operativo al mismo nivel que los reales ni los virtuales creados con TUN/TAP. Por ejemplo no tiene entrada específica en /proc/sys/net/ipv4/conf. Esto provoca que no funcione tratar de obtener una dirección de un servidor DHCP con dhcpclient:

falla cuando intenta abrir un socket raw sobre este dispositivo. Técnicamente nada impide que una tarjeta de red obtenga más de una IP, pues como dhcp-client-identifier no tiene por qué usar su dirección MAC, por lo que usa solución sería un programa que obtenga una segunda dirección, utilizando eth0. Una solución que ofrece dhcpclient es usar "interface" para obtener la IP de eth0 y para la de eth0:1 usar la directiva "pseudo", teniendo que proporcionar además un script para configurar eth0:1 con la IP obtenida. Para más información, ver man dhclient.conf.

Ojo, no usar como alias "eth0:0", pues el alias acabado en ":0" lo reserva precisamente dhclient para la opción "alias", que es para asignar otra IP estática a una interfaz mediante ipalias, para posibles usos en entornos de movilidad donde se requiera también una IP que no cambie con la ubicación.

De todos modos la mejor opción es usar otro cliente de DHCP, en concreto udhcp, que sí pemite trabajar con interfaces creadas con ipalias.

2.2.6. NAT trasversal

El NAT provoca sobre todo dos tipos de problemas:

1. Que la IP y puerto vayan dentro del payload del paquete, por lo que no concordará con el de las cabeceras IP que se modifican en el NAT. Este problema se da por ejemplo con SIP.
2. Que haya NAT en los dos lados, por lo que ninguna de las dos partes puede ser la parte que acepta la conexión entrante.

Para el primer problema una posible solución es que en la máquina que hace NAT exista un módulo que reconoce ese protocolo y modifica también esos contenidos dentro del payload del paquete. Es lo que se conoce como un ALG (Application Layer Gateway). Un problema de los ALG es que en protocolos complejos como SIP limitan el protocolo y que funcionen posibles extensiones. Otro problema es que un ALG está funcionando como un "man in middle" que manipula el paquete, por lo tanto no funcionará si por ejemplo la aplicación está usando TLS para evitar manipulaciones.

Otro tipo de solución es utilizar un servidor STUN (creado para UDP, aunque hay alguna extensión para TCP). El cliente detrás del NAT conecta con el servidor STUN para conecer la IP y puerto por el que sale. Esto puede resolver además el segundo problema con un tipo de dispositivos NAT, en los que el mapeo al puerto es el mismo dependiendo del puerto destino y no de la IP destino. Linux no hace este tipo de NAT, pero sí facilita las cosas utilizando el mismo puerto de origen si está libre.

Otro tipo de servidores son los TURN, que hacen relay y por lo tanto son como redirectores. El problema es que mientras que un servidor STUN sólo se usa para establecer la conexión, un servidor TURN se utiliza durante toda la conexión para enviar todo el tráfico a través suyo.

ICE es un mecanismo, originalmente creado para SIP, que consiste en probar distintas vías y métodos de conexión (conexión directa por cada interfaz, conexión vía STUN, conexión vía TURN) y entre los "candidatos" que tengan éxito se escoge al que se haya dado la mayor prioridad (normalmente a TURN es la que menos, luego STUN). Jingle, en Jabber (es tanto una especificación como una librería), utiliza también ICE[7].

Otras soluciones pasan por modificaciones en la máquina que hace NAT (el middlebox), que permita por ejemplo a un programa pedir que se abran puertos o que le asigne un rango de puertos dentro de una IP pública, pero ninguna de estas soluciones son universales (una buena referencia es la página sobre NAT trasversal de la wikipedia, la versión en inglés). Algunas de ellas se mencionan en el apartado sobre Zeroconf y UPnP de este manual. Cabe esperar que algún día salgo algo más estándar de un grupo de trabajo del IETF como midcomm. Otra posibilidad interesante y antigua es SOCKS, que también permite abrir puertos. El atractivo de todas estas soluciones frente a STUN/TURN/ICE es que se basan en la colaboración con la política de seguridad de la empresa, no en tratar de abrir un agujero en el cortafuegos.

Para IPsec una solución específica es NAT-T, pero por ejemplo en Windows XP SP2 hay que activarla editando el registro[8].

[7] Maemo usa también libjingle y cuenta con un documento introductorio sobre NAT-T interesante: http://maemo.org/development/documentation/how-tos/4-x/how_to_use_stun_in_applications.html

[8] En este SP Microsoft desactivó NAT-T por defecto, por considerar un caso que podría ser un riesgo de seguridad. Esta decisión es muy discutida, pues no todo el mundo ve ese supuesto y consideran más inseguras las alternativas que ofrece a IPsec la propia Microsoft.

2.3. Capa de transporte (Layer4)

Sobre los datagramas IP va la capa de transporte. Aquí hay distintos tipos de protocolos, los más famosos son los TCP (que usan casi todas las aplicaciones) los UDP (que se usan para el DNS y para aplicaciones como streaming de vídeo o para construir los túneles de las VPNs).

Hay más tipos de protocolos de transporte que UDP y TCP: DDCP, SCTP... También están implementados en GNU/Linux pero no lo están en todos los sistemas operativos. Hay que señalar así mismo que protocolos como ICMP e IGMP, así como distintos protocolos de enrutamiento, aunque sean posibles protocolos dentro de un paquete IP como TCP y UDP no son protocolos de transporte sino de red.

Nos centraremos pues en la distinción entre UDP y TCP. TCP es un protocolo orientado a la conexión. Esto quiere decir que con TCP todos los paquetes que pertenecen a una conexión se numeran (el número de secuencia inicial se genera aleatoriamente para dificultar ataques de seguridad de falsificar una conexión) y se garantiza que llegan todos a destino y en el orden en que se enviaron. Al comienzo de la conexión, el primer paquete TCP lleva un flaq indicando que se quiere establecer una conexión. El protocolo TCP incluye mecanismos para confirmar la recepción de los paquetes (los paquetes que no se confirmen que se han recibido hay que volverlos a enviar) así como mecanismos de control de congestión (detectar que se están enviando más paquetes de los que permite el ancho de banda de la conexión y en ese caso bajar el ritmo).

UDP en cambio es un protocolo sin conexión. Cada paquete es independiente de los demás. No hay ninguna garantía de que el programa llegue a su destino, ya sea porque hay errores o porque la red está saturada: si estamos saturando la red y por eso no llegan los paquetes nos tendremos que dar cuenta nosotros y tomar las medidas oportunas (bajar el ritmo de envío). Así mismo no hay garantía que los paquetes que lleguen lo hagan en el mismo orden en que se enviaron, dado que los paquetes IP pueden ir por rutas distintas.

¿Qué aporta entonces la capa de transporte UDP sobre lo que proporciona la capa de red IP? ¿por qué no enviar los datos directamente como datagramas IP en lugar de como paquetes UDP? El gran aporte de UDP, que también forma parte de TCP, es que además de incluir IP de origen y destino, incluye puerto origen y destino. El puerto destino es útil para poder ejecutar varios servicios en la misma máquina: por ejemplo en el puerto 80 puede estar el servidor web y en el 25 el servidor de correo. Podemos ver al puerto destino como una extensión y a la IP como un teléfono. El puerto origen también es muy útil: cuando el cliente envía un paquete elige un puerto origen libre y gracias a este puerto el servidor puede distinguir, a la hora de responder, entre dos conexiones procedentes de la misma IP.

Por lo que hemos visto, TCP garantiza fiabilidad en la conexión mientras que con UDP si se quiere enviar una secuencia de paquetes todo se complica todo y además no hay garantía de que no se pierdan los paquetes. ¿Qué sentido tiene usar entonces UDP? Para la mayoría de los protocolos de aplicación, ninguno: la mayoría requieren un canal de transporte orientado a la comunicación (que dicho sea de paso, podría ser TCP o cualquier otro actual como sockets UNIX o que se cree en el futuro, el API de programación de Unix es independiente del protocolo concreto que esté por debajo). Hay casos, sin embargo, en que UDP es útil:

1. Cuando se trata de un envío simple, de un único paquete y se espera otro paquete de respuesta. Aquí no es problema que el paquete se pierda: si el servidor no responde, se vuelve a enviar la petición. Con UDP sólo se envía dos paquetes (el de petición y el de respuesta), mientras que con TCP se envían varios paquetes para primero establecer la conexión y luego para cerrarla, además de implicar reservar/liberar recursos para el control de la conexión. Un ejemplo son las consultas DNS, o el uso de BOOTP/DHCP (en el caso de DHCP cuenta además que UDP es más fácil de implementar que TCP y puede que la petición BOOTP se haga desde un gestor de arranque o la ROM de arranque de una tarjeta de red, por lo que aún no hay sistema operativo.

2. Cuando no pasa nada porque se pierdan paquetes, pero lo fundamental es que los paquetes que lleguen lo hagan a tiempo, no perdiendo el tiempo en retransmisiones, confirmaciones y control de congestión. Es el caso de streaming de vídeo o la VoIP (voz sobre IP). Otro caso son protocolos de sincronización.

3. Cuando se usa el protocolo de transporte para hacer de túnel transportando otros paquetes, por ejemplo para implementar una VPN. Por un lado está que si transporta paquetes de vídeo con UDP porque lo fundamental es que lleguen muchos paquetes a tiempo aunque algunos se pierdan, si usamos TCP por debajo UDP ya no tiene ninguna ventaja: llegarán todos los paquetes, pero algunos demasiado tarde, más tarde de lo que hubieran llegado si no hubiera TCP por debajo. Con los paquetes TCP estaríamos innecesariamente aplicando dos veces el control de errores, de congestión, de que llegan los paquetes ordenados... TCP está diseñado para ir sobre IP, no sobre TCP; sobre UDP también está bien porque sólo añade unas cabeceras, principalmente para los puertos. Es más, los algoritmos de control de congestión de TCP tienen un

indeseable efecto multiplicador cuando se aplican sobre una conexión que ya va sobre TCP, que se traduce en que tarda más en coger velocidad. Ocurre sobre todo sobre medios físicos propensos a errores, como Wifi. El motivo es que cuando por culpa de un error se pierde un paquete, los algoritmos de congestión lo atribuyen a que se está transmitiendo a demasiada velocidad, por lo que reducen el ritmo de envío de paquetes, con lo que la aplicación que crea los paquetes TCP percibe que hay congestión, pero además una severa, pues suele disminuirse la velocidad a la mitad.

4. Cuando la existencia de NAT en los dos extremos es un problema (conexión cliente a cliente). Con UDP, el enrutador que hace NAT sabe que los paquetes que reciba con destino al puerto origen son paquetes de respuesta y debe dejarlos pasar, no hay propiamente un cliente y un servidor: con TCP esto es más complicado al entrar en juego el flag syn y los números de secuencia, por lo que en la práctica es mucho más difícil lograr que dos máquinas tras NAT logren una conexión TCP que UDP.

El protocolo de transporte DCCP ofrece como TCP control de congestión, pero al igual que UDP no garantiza que todos los paquetes se entreguen ni que lleguen en orden. Básicamente es útil para los mismos casos que UDP, pero con la ventaja de no tener que implementar en la capa de la aplicación el control de congestión. Al estar poco extendido no obstante hoy en día no encontraremos ningún programa conocido que use este protocolo.

El protocolo de transporte SCTP ofrece las mismas características que TCP, pero además ofrece algunas ventajas (con todo apenas se usa, en Windows no está implementado por Microsoft y hace falta instalar software de terceros, pero es especialmente útil para usar con SIP, por ejemplo http://sofia-sip.sourceforge.net/):

1. Multistreaming: permite transportar sobre un único canal varios streams, que se tratarán independientemente en temas como garantizar el orden de recepción de los paquetes (incluso a nivel de stream se puede desactivar el tema del orden si no es necesario). Esta característica es muy útil para trasmitir la señalización de la VoIP de varias llamadas, de hecho SCTP se creó para transmitir SS7 (señalización número 7).
2. Multihoming: en los extremos puede haber más de una interfaz, esto permite por ejemplo usar a la vez dos Ethernet o pasar de una Ethernet a una Wifi sin interrumpir la conexión.
3. Mantiene límites de los mensajes: con SCTP si se escribe un mensaje de 100 bytes al leer en el otro extremo se leen 100 bytes; con TCP esto no está garantizado.
4. mecanismos para evitar ataques syn flood; también elimina problemas en cierres de conexiones que presenta TCP.

2.4. Capa de aplicación (Layer5; Layer7 en OSI)

Finalmente la última capa es el protocolo de aplicación, por ejemplo HTTP (web), SMTP (correo) o XMPP (Jabber). En Unix estos protocolos se escriben utilizando sockets, que se utilizan (sobre todo cuando la capa de transporte es TCP) prácticamente igual que los ficheros. Existe una especificación de un framework llamado BEEP (RFC 3080-3081, 3620), con implementaciones libres, que permite crear protocolos de aplicación olvidando temas como la autenticación y negociación de un canal seguro (para lo que usa SASL y TLS respectivamente) así como el modo de multiplexar varios canales sobre una conexión o manejar mensajes asíncronos (en cambio el formato de los mensajes en sí es libre). También estandariza la creación de túneles (proxies).

Dado que con los sniffers es posible espiar el tráfico de red, un mecanismo de seguridad muy utilizado en las aplicaciones es añadir una capa intermedia en la capa de aplicación sobre la que realmente va el protocolo: esta subcapa lo que hace es cifrar los paquetes y añadirles información de control para detectar si alguien los intenta manipular. El protocolo más estándar par implementar esta funcionalidad es TLS (anteriormente conocido como SSL).

Ojo, puede parecer que un switchs elimina la posibilidad de que un nodo en una red local espíe las comunicaciones entre otros nodos, utilizando un sniffer. Al fin y al cabo con un hub todo el tráfico se reenviaba a todos los nodos y el nodo en escucha realiza una actividad totalmente pasiva e indetectable, pero con un switch se supone que cada nodo sólo recibe los paquetes que van destinados a él. Lamentablemente no es así, existen sniffers como http://ettercap.sourceforge.net/ que funcionan también con switchs[9]. Así mismo hay sniffers especializados como oreka.sf.net, que sirven para grabar VoIP, generando un fichero de audio.

[9]Otro sniffer, o conjunto de sniffer revelador de que los switch no son suficientes para evitar que se espíe el tráfico de red es dsniff (www.monkey.org/~dugsong/dsniff/). Este paquete por un lado incluye sniffers pasivos, algunos especializados para tareas como capturas contraseñas, correos, URLs (hay otro programa al margen de dsniff especializado para grabar VoIP en ficheros de audio, oreka.sf.net) y por otro programas para atacar los switchs. Así mismo incluye programas para hacer ataques "man in middle" para servidores SSL y SSH cuando se usan certificados autofirmados o sin comprobar, presentando uno falso sobre la marcha.

El problema es el ARP Spoofing/Port Stealing y ARP Poisoning. El ARP poisoning consiste en enviar al nodo que queremos engañar una respuesta ARP en el que indicamos que nuestra dirección MAC es la correspondiente a la dirección IP del nodo que queremos suplantar. La mayoría de los sistemas operativos no ven nada extraño en que les llegue este paquete aunque no lo hayan solicitado y lo almacenan en su caché ARP y en los que no se puede trucar con un ping. El ARP posioning actúa falsificando la IP no la dirección ARP, por lo que es difícil de detectar y evitar por parte del switch. Otras técnicas (Port Stealing) explotan engañar al switch haciéndole creer que tenemos la dirección ARP de la máquina a suplantar (los swtich que no soporten Safe Port, es decir, que permitan que cambie el puerto asociado a una dirección MAC). Así mismo hay ataques que buscan saturar con entradas falsas la tabla CAM del switch, pues necesariamente esta tabla tiene capacidad limitada.

La buena noticia es que estos ataques ya no son pasivos sino activos y son detectables con herramientas apropiadas como arpwatch (que hace escucha pasiva) o el propio ettercap. Más información: http://en.wikipedia.org/wiki/ARP_spoofing

Si necesitamos un sniffer para hacer comprobaciones de red o aprender sobre protocolos, el sniffer recomendado es Wireshark.

Hay más herramientas útiles para depurar problemas en las redes, además de los sniffers. Por ejemplo iftop (http://www.ex-parrot.com/pdw/iftop/): muestra consumo de ancho de banda, al estilo de top, mostrando origen y destino de la conexión. Utiliza libpcap (como sniffer) por lo que al ponerlo en la máquina que hace de router podemos ver quién está consumiendo en ese momento más ancho de banda.

Una herramienta extremadamente útil para depurar problemas de red y aprender sobre redes es Netcat. Esta utilidad permite conectarse a cualquier socket TCP/UDP y enviar/recibir datos como si estuviéramos escribiendo en un terminal. De igual modo puede escuchar en un puerto TCP/UDP. Es muy útil por ejemplo para probar antes de montar una VPN si hay visibilidad entre las Ips y puertos: escuchamos con netcat en un lado y desde el otro enviamos. Así, para escuchar tráfico UDP en el puerto 9037 (opcionalmente con -s se podría indicar una IP, en el caso de ser un puesto multihome, con más de una interfaz de red): **`netcat -u -l -p 9037`**. Para enviar tráfico a el puerto 9037 desde otra máquina: **`netcat -u 83.59.36.220 9037`**

Otra herramienta aún más versátil que netcat, pero mucho menos conocida, es socat (www.dest-unreach.org/socat/doc/socat.html). Permite usar todo tipo de conexiones, incluso sockets RAW, sockets Unix, ficheros, crear dispositivos TUN/TAP, usar SSL, abrir un PTY, usar un socket HTTP o SOCKS...

Se recomienda usar también un NIDS (sistema detector de intrusiones de red, también suele usarse el acrónimo IDS, pues la mayoría de los IDS son NIDS, pero los hay específicos para una aplicación o para un sistema, que analizan llamadas al sistema). El más conocido es snort (http://www.snort.org/), aunque es programa es polémico debido a que las nuevas reglas para detectar intrusiones ya no son libres, aunque sí gratuitas pero distribuidas con varios días de retraso para los clientes no de pago: un sitio dónde hay reglas que sí son libres es www.bleedingthreats.net/ . Snort es un NIDS reactivo; permite tomar medidas para neutralizar el peligro, frente a los NIDS pasivos que sólo lo detectan. Los sistemas de este tipo se conocen como IPS (Intrusion Prevention System; sistemas de prevención de intrusiones).

También es útil disponer de un sistema de detección de vulnerabilidades como Nessus (lamentablemente desde la versión 3 no es software libre, pero hay un fork que sí lo es http://www.openvas.org/: el problema de todos modos es tener una base de datos de vulnerabilidades que sea de libre uso); ojo con utilizar esta herramienta en una red que no administremos, entre los cometidos de un NIDS como snort es detectar este tipo de monitorizaciones. No confundir un detector de vulnerabilidades o security scanner con un escaneador de puertos (port scanner), como nmap, aunque el escaneo de puertos forme parte del proceso de búsqueda de vulnerabilidades. Nmap no detecta vulnerabilidades, sino localiza qué máquinas hay en la red, qué puertos tienen abiertos, qué sistemas operativos usan y qué versiones de los servidores de red utilizan. Un NIDS también detectará escaneos con Nmap, por lo que sólo deberían usar esta herramienta administradores de redes que tratan de encontrar vulnerabilidades en su propia red, principalmente debidas a máquinas "incontroladas" que no siguen la política de seguridad fijada para la red.

Para redes wireless hay programas como kissmet (http://kismetwireless.net/) que actúa de forma pasiva y no es detectable, permitiendo detectar de hecho ataques de programas no pasivos como netstumbler, además de poder usarse con snort y con otros sniffers (para lo cual deberemos conocer la clave). Un programa de "wardriving" (localización de redes Wifi desplazándose con un vehículo y un portátil con la tarjeta en modo monitor) es swscanner (http://www.swscanner.org/); un uso lícito de este tipo de programas es detectar redes no autorizadas o que no siguen la política de seguridad de la empresa u organización, otro es comprobar temas de cobertura y un tercero depurar problemas de conexión.

Capítulo 3

Autoconfiguración: DHCP, PXE, Zeroconf

3.1. Servidores DHCP

Un servidor DHCP sirve para que los equipos presentes en una red obtengan su configuración de red automáticamente. Así, el servidor asigna a cada equipo su IP y lo comunica su máscara de red, las rutas, el DNS, servidor de impresión, servidor de hora, zona horaria etc. Opcionalmente permite hacer un arranque vía red. Además es posible definir nuevos parámetro de configuración si se extienden también los clientes DHCP para que reconozcan y apliquen estos parámetros.

DHCP es un protocolo surgido con posterioridad a BOOTP; en realidad es una extensión compatible con él, es decir, un cliente BOOTP también puede operar con servidores DHCP.

El servidor DHCP permite al administrador asignar una IP manualmente a un cliente (manual allocation); permite asignarlas automáticamente sin que caduque nunca la asignación (automatic allocation) o permite asignarlas automáticamente por un tiempo limitado (dynamic allocation), de tal modo que si el cliente no vuelve a pedirlas antes que finalice ese tiempo se marcará la IP como libre.

El servidor DHCPD guarda en una tabla las asociaciones entre clientes e IPs. El servidor identifica al cliente a través de su identificador de cliente, si es que lo proporciona, en otro caso usa su dirección MAC.

En un segmento Ethernet puede haber más de un servidor DHCPD; el cliente envía usando broadcast un mensaje DHCPDISCOVER (puede indicar en él la IP que le gustaría tener y durante cuánto tiempo) al que responden con un mensaje DHCPOFFER los servidores presentes; en ese mensaje ya va la IP que le ofrecen y los datos de configuración. El cliente escoge el servidor que quiere utilizar enviando una petición DHCPREQUEST que incluye el identificador del servidor; esta petición también es broadcast, para que sepan el resto de servidores que su ofrecimiento se ha rechazado. En la petición además se pueden solicitar datos de configuración adicionales. Si el servidor acepta la petición del cliente responde con DHCPPACK (dónde irá configuración adicional si la solicitó el cliente), o DHCPNAK si la rechaza, debido por ejemplo a que la IP ya está asignada; a priori no tiene mucho sentido que un servidor rechace la petición del cliente cuando son los datos que le acaba de ofrecer, pero en realidad un cliente puede hacer una petición DHCPREQUEST sin acabar de recibir un DHCPOFFER; por ejemplo una vez que ya se tiene una IP para renovar el "alquiler" se pide directamente la IP al servidor DHCPD que nos la dio sin necesidad de hacer DHCPDISCOVER.

Finalmente, el cliente tiene la opción de rechazar el mensaje DHCPPACK. Así mismo el cliente también tiene la opción de liberar la IP que se le ha asignado, con DHCPRELEASE.

En las peticiones DHCP es muy importante el uso del broadcast. Para permitir que funcione el DHCP sin necesidad de poner un servidor en cada subred se usan BOOTP/DHCP relay agents, que se encargan de reencaminar estas peticiones.

Para casos como una VPN, en la que el nodo que se conecta lo queremos asignar una IP de la red local, posiblemente no nos funcione usar un cliente dhcp normal, ni siquiera acompañado de un relay. Una posible solución es ejecutar un cliente dhcp con la interfaz de la red en la que está el servidor DHCP (no pasa nada porque ya haya otro si se usan identificadores de cliente distintos y distinto fichero para guardar la ip asignada (lease), pero con un script que en lugar de configurar esa interfaz con la IP asignada se encarga de configurar la VPN para que use esa IP.

3.1.1. Software DHCP

1. Clientes: dhcpcd, dhcp-client (dhcp3-client), pump, udhcp (cliente muy pequeño)
2. Servidores: dhcp3-server, udhcp-server (servidor muy ligero)
3. Relays: dhcp3-relay, dhcp-helper. Es más ligero y más fácil de utilizar dhcp-helper.
4. Otros: dnsmasq: servidor caché DNS y servidor DHCP muy básico, útil para sistemas empotrados o para casos en que se necesita una implementación mínima y no un servidor para utilizar por muchos clientes. Ltsp-server: sistema de clientes ligeros, usan DHCP para obtener la configuración y arrancar vía red.

3.2. DHCP vs ZeroConf y UPnP

Zeroconf (www.zeroconf.org) es un sistema de autoconfiguración de dispositivos, que pretende que los dispositivos funcionen con sólo conectarlos a la red, sin necesidad de que para ello existan servidores en la red local (servidores DHCP, DNS o directorio). Hay otras alternativas a Zeroconf como UPnP de Microsoft.

El componente más extendido de zeroconf es Ipv4 Link local Addresses (IPV4LL), que se define en el RFC1397. El propósito de IPV4LL es que un equipo obtenga automáticamente una IP sin necesidad de que exista un servidor DHCP, RARP ni de ningún otro tipo. El modo de lograrlo es elegir al azar una IP dentro del rango reservado para este sistema: 169.254.0.0/16; se comprueba que la IP no está en uso, si lo estuviera se escoge otra hasta encontrar una libre.

La utilidad de este sistema frente a DHCP está en redes "ad-hoc" (sobre la marcha), por ejemplo para comunicar dos equipos a través de un cable cruzado o conectándolos a una red local pero sin afectar al resto de equipos. También puede ser útil en una red pequeña aislada en una oficina o en una casa, dónde no se quiera tener encendido un servidor para hacer de DHCPD. Sin embargo lo normal en una red de empresa o de casa es que haya un acceso a Internet utilizando un router ADSL, que ya integra un servidor DHCP. Otro tipo de redes "ad-hoc" interesantes son las MANET (Mobile Ad-hoc NETwork) como las wireless mech network que usan OLSR, pero quedan fuera del alcance de este manual.

En GNU/Linux existe el proyecto zeroconf.sf.net, que ha creado el programa zcip que implementa Ipv4LL.

UPNP también usa IPV4LL, tras intentar usar un servidor DHCP. En Windows podemos comprobar que cuando configuramos la red para obtener la IP automáticamente, primero intenta usar un servidor DHCP y si no lo logra obtiene una IP aleatoria: eso es porque implementa Ipv4LL.

Otro componente interesante que aporta Zeroconf es mDNS (multicast DNS). Este sistema sirve para reemplazar al DNS. Así mismo zeroconf aporta DNS Service Discovery (DNS-SD) que permite utilizar el DNS para localizar servicios, por ejemplo el servidor Jabber. DNS-SD se puede implementar sobre mDNS o sobre un servidor DNS convencional.

La idea de mDNS es que cada nodo incluye su propio servidor que responde con su IP cuando alguien pregunta por su nombre o hace una búsqueda inversa (el nombre a partir de la IP). En el caso de usar mDNS para implementar DNS-SD, el servidor proporciona información sobre los servicios que ofrece la propia máquina. La consulta se hace utilizando broadcast: la idea es la misma que para obtener el nombre NetBIOS de un equipo de Windows cuando no hay DNS ni servidor Wins.

El utilizar el servidor DNS para localizar servicios no es una idea nueva: el servidor de correo correspondiente a un dominio siempre se ha localizado así (aunque no el servidor de correo saliente). Así mismo el RFC 2782 (Service Types) describe el registro SRV para los servidores DNS que extiende lo que se hacía con el correo a todo tipo de servicios: por ejemplo Jabber usa este tipo de registros. Zeroconf usa también otro tipo de registros DNS: los TXT.

La utilidad de estos registros sobre los SRV, es que los SRV están pensados para que se ofrezca una máquina por tipo de servicio o en todo caso varias, pero a efectos de balanceo de carga o redundancia. Los registros TXT son útiles para servicios como impresoras: en una red puede haber varias impresoras pero normalmente no interesa usar una cualquiera sino que se puede preferir una u otra en función de por ejemplo si imprime o no en color o determinado tamaño o si está en la misma planta que el usuario. En los registros TXT además del nombre del servicio aparece una etiqueta, la instancia (instance) que admite caracteres UTF-8. Cualquier etiqueta DNS está limitada a 63 bytes; ojo, que en UTF-8 caracteres como la eñe o los acentos ocupan dos bytes. Los registros TXT permiten más flexibilidad: por ejemplo se puede poner un servicio para ver una página personal y entre la información no sólo estará el nombre del servidor y el puerto sino la ruta.

La funcionalidad de DNS-SD es utilizable en más casos que Ipv4LL. Por ejemplo en las empresas para descubrir impresoras, servidores de ficheros.. Incluso hay aplicaciones como sistemas para compartir archivos. En el escritorio se integra en el panel de control y permite navegar por los servicios disponibles en la red.

DNS-SD añade además unos nombres de host específicos al DNS para indicar que es navegable para descubrir servicios:

```
b._dns-sd._udp IN PTR @ ; b = browse domain
lb._dns-sd._udp IN PTR @ ; lb = legacy browse domain
```

En GNU/Linux la implementación de mDNS y DNS_SD recomendada es Avahi (http://avahi.org/). Proporciona un demonio y una librería para las aplicaciones.

En Apple la implementación de Zeroconf primero se llamó Rendezvous y luego Bonjour. Lo implementan como software libre y está portado también a Windows. Mucha gente conoce más estas marcas de Apple que Zeroconf, debido a que esta iniciativa ha sido impulsada por Apple, al migrar de AppleTalk a TCP/IP y querer conservar el carácter autoconfigurable propio de AppleTalk.

UPnP trata de ofrecer funcionalidades similares a mDNS y DNS_SD, pero no son compatibles y la implementación es más compleja. Va más allá e incluye posibilidad de hacer llamadas SOAP, o abrir puertos en un firewall para traspasar el NAT. Esto último lo hace IGD (Internet Gateway Device) y hay una implementación para GNU/Linux: http://linux-igd.sourceforge.net/; sin embargo conviene saber que IGD es una característica insegura y que potencialmente un nodo podría hacer que se abran agujeros que suponga acceder desde el exterior a otras Ips si se hace IP Spoofing. Apple también tiene su alternativa a IGD, que es NAT-PMP (NAT Port Mapping Protocol), que implementan también proyectos libres como Stallone.

La parte que más éxito ha tenido ha sido UPnP AV, que trata sobre dispositivos de audio y vídeo con conexión a red. Hay varios proyectos multimedia que pueden hacer streaming para dispositivos UPnP o ser los reproductores: por ejemplo Myth, VideoLan, Geexbox, MediaTomb... UPnP AV distingue entre servidores de contenido (MediaServers, que opcionalmente pueden encargarse también de convertir entre formatos, MediaAdaptor), reproductores (MediaRenders), controladores y mando a distancia. Un servidor libre multiplataforma escrito en C++ es FUPPES; también libre en ese lenguaje es Mediatomb, mientras uShare está escrito en C. Todos ellos funcionan también sobre dispositivos empotrados ARM como el NSLU2. Por supuesto estos protocolos no son los únicos para implementar servidores de streaming: por ejemplo está también icecast o el estándar del IEFT RTSP (recibe comandos como PLAY, luego el vídeo se envía con RTP[1], usándose RTCP como protocolo compañero de RTP para controlar estadísticas sobre la calidad del servicio). Así mismo hay tecnologías "peercasting" basado en uso de peer-to-peer como alternativa al alto coste de usar unicast o la poca flexibilidad de multicast.

SLP: sistema de descubrimiento, a diferencia del de UPnP y el de Zeroconf sí es estándar IETF. No muy extendido, excepto para localizar las impresoras de red. Hay una implementación libre que es OpenSLP. Funciona con un servidor, pero también puede estar más descentralizado, utilizando broadcast/multicast.

3.3. PXE

Una opción interesante en una red para ahorrar costes en los equipos y coste de mantenimiento es utilizar un sistema de clientes ligeros (como TCOS [www.tcos-project.org], LTSP, PXES o ThinStation), con arranque vía red (el arranque vía red también es típico para montar clusters). Para ello se utiliza una tarjeta con una ROM de arranque, que vía DHCP y TFPT descarga el kernel (o más frecuentemente un cargador más potente que es quien descarga el kernel) y luego monta el sistema de ficheros vía algún sistema como NFS o SMB (entornos NAS: Network Area Server) o menos frecuentemente, con un servidor SAN (Storage Area Network, se utiliza un disco externo vía red, directamente).

Hoy en día normalmente ya no hay que sobreescribir una EEPROM para añadir arranque de red, sino que viene integrado en la BIOS de cualquier placa la solución estandarizada por Intel: PXE, que es el acrónimo de Preboot Execution Environment, estandariza cómo descargar un software para descargar un cargador de red (Network BootStrap Program, NBP) que se encargará del arranque (así pues es un sistema de carga en cadena, pues lo que carga es el cargador a utilizar). Un cargador de red hecho para cargarse vía red desde un equipo con una BIOS o tarjeta de red que soporte PXE es LinuxPXE.

Existe un proyecto para crear un cargador de red, Etherboot, que puede grabarse en muchas tarjetas de red del mercado, lo que es útil tanto para sistemas muy antiguos que no soporten PXE, como si se desea instalar directamente un sistema más simple. Con el proyecto gPXE, Etherboot se ha reorientado a crear una versión que extiende PXE, con funcionalidades como iSCI o la posibilidad de descargar un kernel de un servidor web. gPXE no tiene por qué grabarse en la ROM de una tarjeta o en la BIOS de un equipo: también puede usarse en cualquier dispositivo de arranque (CD, disquete, llave USB) o simplemente ser el cargador

[1] RealTime Transport Protocol: se suele encapsular sobre UDP (aunque no tiene un rango de puertos fijos asignados, el hecho de usar un rango dinámico complica su uso con NAT) y puede usarse en multicast además de unicat. Además de para streaming es habitual para videoconferencia, en ese caso para la señalización se usa SIP, H.323 o Jingle en lugar de RTSP. Aunque RTP es el protocolo más usado para transmitir voz y vídeo, otra opción es IAX (Inter-Asterisk Exchange) que tiene la peculiaridad de encapsular en la misma conexión la señalización y los datos y poder multiplexar varios flujos en uno. Esto sobre todo lo hace mucho más adecuado frente a NAT y cortafuegos que RTP, además de más eficiente en consumo de ancho de banda.

de red que carga un sistema PXE. Salvo en el caso de funcionar como un NBP que se carga desde PXE, deberá hacerse una imagen en concreto para la tarjeta que disponga el equipo.

Otro proyecto similar a Etherboot es Netboot.

La BIOS implementada por el proyecto LinuxBIOS (surgido inialmente para OLPC) no sólo permite arrancar Linux sino entre otras cosas pasar el control a Etherboot. Esta BIOS normalmente sólo aparece en equipos específicamente fabricados para llevar Linux, pero puede usarse también con emuladores/virtualizadores como Qemu, aunque normalmente lo que se usa es según la plataforma emulada o una BIOS del proyecto Booch u OpenBIOS, que implementa la especificación OpenFirmware y necesita previamente otro software más dependiente de la máquina como LinuxBIOS: openfirmware es otro de los "payloads" que puede cargar LinuxBIOS, como el kernel, etherboot o GNUFI (que implemente UEFI, United EFI, el remplazo de la BIOS).

Capítulo 4

Servidores de red para empresas

4.1. DNS

Tener un servidor DNS público exige unos requisitos. Por lo pronto no basta con un servidor, como mínimo hay que tener dos, que deberían estar en localizaciones diferentes. La mayoría de las empresas optan por recurrir a una empresa que preste servicios de DNS, al menos para que les mantenga el servidor secundario, que periódicamente se actualiza del servidor maestro haciendo una transferencia de zona. Hay servicios DNS gratuitos como http://www.startssl.net/

En nuestra red necesitamos un servidor DNS pero no para dar servicio a nuestro dominio, sino para que puedan resolver las direcciones DNS los usuarios de nuestra red, tanto las peticiones que van dirigidas a los servidores internos de la empresa con la que estamos conectados (dominio interno .acme), como los de la salida a Internet. Aclaremos que los servidores DNS de la empresa con la que estamos conectados no sólo sirven el dominio .acme sino también los de Internet, pues son los servidores DNS que usan todos los equipos de su red, por lo que podríamos configurar también nuestros equipos para que usen su DNS.

Es más práctico tener un servidor propio, que envíe las peticiones para resolver las direcciones de Internet al DNS de nuestro ISP y las de máquinas del dominio .acme a sus servidores DNS. La primera razón es porque no es lógico que cada vez que un usuario acceda a Internet usemos los servidores de la otra empresa: por privacidad, por buen uso de nuestras propias instalaciones y porque no deberíamos depender de si funciona bien la conexión con la otra empresa para poder usar Internet. Además si un día se añade conectividad con otra empresa más es evidente que habrá que consultar a cada empresa por su DNS.

Ejemplo de fichero de configuración:

```
// Ips que tienen acceso al servidor: sólo la red local
acl redinterna {
192.168.120.0/24;
};

options {
directory "/var/named";
forward first;
// Para resolver direcciones, usamos los servidores de nombres de
// nuestro ISP
forwarders {
80.58.0.33;
80.58.32.97;
};
// Esta línea es para usar la IP de la red local, en lugar de la IP de la
// interfaz de salida (la de la conexión con la red con la otra empresa o
// la de Internet
query-source address 192.168.120.19 port *;
};

// Servidores raíz
zone "." IN {
```

```
type hint;
file "named.ca";
};

zone "localhost" IN {
type master;
file "localhost.zone";
allow-update { none; };
};

zone "127.in-addr.arpa" IN {
type master;
file "named.local";
allow-update { none; };
};

include "/etc/rndc.key";
// aquí ponemos los servidores DNS de la red de la otra empresa
zone "acme" {
type forward;
forwarders {
10.15.8.16;
};
forward only;
};

// Resolución inversa, también para la red de la otra empresa

zone "10.in-addr.arpa" {
type forward;
forwarders {
10.15.8.16
};
forward only;
};

// resolución inversa de nuestra zona
zone "15.168.192.in-addr.arpa" {
type master;
file "/var/named/192.168.120.rev";
};
```

4.1.1. DNS dinámico

Hay muchas web y sistemas para que un equipo con IP dinámica pueda dar de alta su IP en un DNS cada vez que se conecta o cambia. Básicamente lo que se hace es conectarse a un servidor, que analiza la IP de origen: esa será la IP que registrará, tras comprobar que el usuario aporta unas credenciales que le autorizan para cambiar su entrada DNS.

No hay un único sistema, por lo que hay unos programas que sirven para unos sitios y otros para otros, habiendo programas que soportan varios protocolos, como ez-ipupdate.

El paquete http://gnudip2.sourceforge.net/ sirve para implementar nuestro propio servidor de DNS dinámico, junto con el cliente y la descripción del protocolo.

4.2. Servidor de correo

En general no es buena idea tener un servidor propio de correo, si se trata de una empresa pequeña con un ADSL. Hay muchas empresas que ofrecen redirecciones de correo (en unos casos limitadas a 100 o 200 redirecciones distintas, en otros ilimitadas)

por un precio razonable. Es posible usar como destino de las redirecciones por ejemplo cuentas de gmail, que nos permiten usar la dirección que queramos para el correo saliente. Así mismo la propia Google ofrece paquetes para empresas tipo Gmail pero con dominio propio.

Si no obstante nos decidimos, el servidor de correo de un dominio se fija con un registro MX en el DNS; es posible tener varios registros para que si está caído un servidor, se acuda a otro. Las empresas que venden servicio DNS como EasyDNS suelen ofrecer también servidor de correo de respaldo, es decir, añadir uno de sus servidores de correo para que reciban el correo en el supuesto que nuestro servidor esté caído.

Usar como servidor de correo saliente uno propio en lugar de el del proveedor teóricamente permite más control como saber si un mensaje ha sido recibido por el servidor destino; así mismo permite que si la conexión a Internet esté caída el servidor siga aceptando mensajes y los envíe en cuanto se restablezca la conexión, aunque no todo el mundo ve esto como una ventaja. Un inconveniente es que las Ips asignadas a las líneas ADSL a veces están en listas negras para evitar correo basura, por lo que no son las ideales para situar un servidor de correo. En concreto, las Ips de ADSL de Telefónica están en SPAMHAUS PBL http://www.spamhaus.org/pbl

a destacar que sitios como easydns.com bloquean el correo procedente de esas direcciones.

A priori una buena razón para montar un servidor de correo propio es evitar que los mensajes internos salgan a Internet. Sin embargo, recomendamos reemplazar los correos internos por mensajería instantánea utilizando un servidor con Jabber.

Si pese a todo insistimos en tener un servidor de correo propio, recomendamos usar PostFix como software y administrarlo a través de Webmin (www.webmin.com).

Tanto si instalamos nuestro propio servidor de correo como si usamos un proveedor, si tenemos un dominio propio es importante que usemos SPF, Sender Policy Framework (http://www.openspf.org/). Se trata de una medida antispam, consistente en añadir al DNS unos registros indicando qué servidores están autorizados para enviar correo en el que el remitente sea una dirección de nuestro dominio.

softlibre.net INT TXT "v=spf1 mx a:chemahome.softlibre.net include:gmail.com include:easydns.com -all"

En este caso hemos incluido con include las direcciones autorizadas para enviar con gmail.com, puesto que lo uso para enviar correos con @softlibre.net, y easydns.com, pues es quien se encarga de mis redirecciones. Además incluyo todo servidor de nombres del dominio (registros MX) y explíticamente a la máquina chemahome.softlibre.net. Observesé que para simplemente incluir todas las máquinas con registros de nombre A en el dominio bastaría con poner sólo "a". Al final con "-all" se indica que se deben rechazar todos los mensajes que no procedan de ninguno de los sitios indicados. Mientras estamos probando es mejor utilizar "~all", que en lugar de rechazar implica que el servidor añada una cabecera para que lo puedan detectar en los filtros de los programas de correo.

Es importante así mismo que pongamos una regla en el cortafuegos para prohibir conexiones al puerto 25 de otra máquina que el servidor de correo de la empresa (si es externo) o si es interno que sólo se pueda conectar al puerto 25 desde la IP del servidor de correo.

El servidor deberá aceptar correo destinado a su propio dominio y filtrar (o al menos clasificar) el correo que conforme a SPF es falso. Para enviar correo sólo deberá aceptar en el que el from sea el propio dominio y hacerlo autenticado.

Como información complementaria, este artículo explica como añadir a un servidor de correo un fitro ANTISPAM y un antivirus: http://www.howtoforge.com/amavisd_postfix_debian_ubuntu

4.3. Servidor Wiki

Un recurso muy útil para empresas es montar un wiki, una herramienta colaborativa en el que las personas pueden crear páginas de contenidos accesibles vía web, desde el propio navegador, de forma muy sencilla. Estas páginas las podrán modificar el resto de usuarios: en todo momento se podrá revisar el histórico de las páginas, ver quién ha hecho cada cambio... Un wiki además integra un buscador. En definitiva, es una herramienta muy útil para mantener información de uso interno entre un grupo de personas.

Hay distinto software para implementar un wiki. Aunque a la hora de editar editar las páginas de un wiki todas las soluciones son muy parecidas, lo cierto es que son distintos por lo que la gente cuando se acostumbra a uno le cuesta pasar a otro; así mismo no es trivial migrar de un wiki a otro. Vamos a mencionar cuatro soluciones:

1. Mediawiki (www.mediawiki.org). Licencia GPL. Es el software de la wikipedia, pero se usa en muchos otros proyectos, como Mozilla. Realmente no es la opción más flexible para empresas, por ejemplo de las propuestas analizadas es la más floja en el control de acceso para quien edita o ve las páginas; sólo resulta adecuada para un control básico, por ejemplo para que sólo puedan ver las páginas y editarlas los usuarios que están dados de alta en el wiki. Sus puntos fuertes son el atractivo visual de las páginas (además con versión imprimible) y que a muchos usuarios les resulta más familiar por usarse en muchos wikis y en la wikipedia. Otro punto fuerte es que es fácil de editar y muy flexible: se pueden editar fórmulas matemáticas y mediante GraphViz es sencillo crear diagramas. Incluye funcionalidades como página de discusión, posibilidad de notificar al usuario cuando la página cambia, control de páginas huérfanas, posibilidad de subir ficheros... Para instalar mediawiki hace falta Apache, PHP y MySQL.

2. Twiki (http://twiki.org/). Licencia GPL. Este software lo usan muchas empresas, como Michelin, Disney, SAP, Yahoo... Tiene plugins muy interesantes, como ActionTracker (para hacer TO-DO list, que pueden compartirse con otros usuarios), Calendar (calendario, con eventos resaltados) o un editor de gráficos en Java. Para instalar Twiki hace falta Apache y Perl (no hace falta un gestor de base de datos).

3. MoinMoin: (http://moinmoin.wikiwikiweb.de/. Licencia GPL. Lo usan proyectos como Ubuntu, Debian, Apache, Hispalinux... Entre sus puntos fuertes que es fácil de instalar: está escrito en Python y no requiere más que un servidor web, por ejemplo Apache. No guarda las páginas en BB.DD.

4. TikiWiki: (http://tikiwiki.org/) . Licencia LGPL. Destaca por no ser sólo un wiki: también integra funciones de CMS (gestor de contenidos) y solución groupware, por lo que es muy interesante para empresas. Integra webmail, foros, blogs, encuestas, calendario, agenda de direcciones, hoja de cálculo... Para instalarlo requiere PHP y un gestor de BB.DD.: MySQL es el único sistema totalmente soportado, pero otros lo están parcialmente.

4.4. Servidor Jabber

XMPP es el protocolo de mensajería instantánea estandarizado por el IETF, el organismo que establece los protocolos de Internet. Es más conocido como Jabber, el nombre con el que se creó en 1998.

XMPP significa eXtensible Message Presence Protocol. Lo de "extensible" es muy importante: Jabber usa XML y gracias a ello se le han añadido todo tipo de extensiones, como invocar llamadas RPC entre dos clientes. Esto permite todo tipo de aplicaciones; recuerda la revolución que ha supuesto las aplicaciones web, sólo que aquí con más posibilidades: la comunicación no está limitada a cliente servidor sino entre dos clientes, que pueden variar su IP y ubicación. Además la comunicación es totalmente bidireccional, mientras que en la web es petición-respuesta: el navegador efectúa una petición y el servidor web responde, de modo que si es el servidor el que quiere notificar algo al navegador no puede salvo que se haga polling: el navegador envía una petición tipo "respondemé cuando tengas algo que decirme".

De igual modo que con el correo, una organización puede tener su propio servidor Jabber interno o elegir entre distintos proveedores. También como ocurre con los servidores de correo, las comunicaciones entre usuarios del mismo servidor no salen del servidor y por lo tanto de la red local de la organización, mientras que si se contacta con un usuario no local se localiza su servidor a partir de su dirección, que son iguales de las de correo, los servidores se comunican entre sí.

Esta es una gran diferencia respecto a soluciones propietarias como MSN Messenger, AOL o Yahoo, dónde hay un único servidor para todos los usuarios, lo que lo hace muy poco adecuado para empresas: incluso el correo interno pasa por Internet, por el mismo servidor que el de su competencia. Sería como si para tener correo electrónico una empresa sólo pudiera abrir cuentas en Hotmail o en Gmail y al escoger uno de estos dos sitios no pudiéramos enviar ni recibir mensajes del otro.

Otro problema de las soluciones propietarias es que se está en manos del proveedor incluso para elegir el programa de mensajería. En algunos casos se pueden usar distintos programas pero el proveedor tiene la sartén por el mango y puede en un momento dado no dejar que se conecte nadie con otro programa que el oficial. Es el caso de AOL, que durante un tiempo para evitar conexiones de otros clientes preguntaba al programa fragmentos de su propio código, que al estar bajo copyright no podían replicar los otros programas.

De hecho la propia Microsoft tiene una solución de mensajería instantánea para empresas que no usa el mismo protocolo que MSN sino en un software servidor; otro producto muy conocido de mensajería para empresas es Lotus SameTime. En realidad estas soluciones rivales de Jabber se basan en un protocolo que también está siendo estandarizado por el IETF, SIMPLE, pero a pesar de comenzar el proceso antes que con XMPP está más atrasado y hay aspectos básicos sin estandarizar que por lo tanto no son interoperables. De hecho mientras que hay miles de servidores que se comunican a través de Internet al estilo de los

servidores de correo, de modo que puede contactar con cualquier usuario a través de su dirección, no ocurre así con los servidores de SIMPLE.

El paralelismo entre correo y Jabber es tan grande que el formato de direcciones es el mismo: en Google Gmail de hecho cada cuenta es una dirección de correo y de Jabber

Ventajas de tener un servidor de mensajería propio:

1. Los mensajes no salen de la organización. Mayor confidencialidad y seguridad. Mejor aprovechamiento de la conexión a Internet, pues es absurdo que para enviar un mensaje local éste salga a un servidor de Estados Unidos y vuelva.
2. Posibilidad de controlar el flujo de la información y con quien pueden contactar los usuarios.
3. Posibilidad de registrar las conversaciones.
4. Opción de añadir automáticamente contactos a todos los usuarios de un grupo.
5. Posibilidad de usar control de acceso corporativo: todo el mundo accede con su usuario y contraseña que usa para acceder al resto de aplicaciones de la empresa o mediante tarjeta de acceso.
6. Posibilidad de directorio de usuarios (usar uno en Internet hace más difícil las búsquedas y problema de no poder incluir información confidencial.
7. Posibilidad de enviar mensajes a todos los usuarios o poner mensajes del día
8. Posibilidad de añadir funcionalidad al servidor o de integrar otros servicios como VoIP.
9. Uso de un dominio propio. Mejor imagen corporativa. No se pierde disponibilidad si cae conexión a Internet o servidor del proveedor.

XMPP no es un protocolo que sólo sirva para mensajería, es realmente un protocolo que puede substituir en muchos usos a HTTP como protocolo para aplicaciones de Internet. Por ejemplo está estandarizado todo el tema de la autenticación, el mecanismo publish-subscribe, comandos ad-hoc, XML-RPC y SOAP, comunicación in-band (envío de datos arbitrarios usando el propio canal XMPP) y out-band con SOCKS5... Google ha aportado Jingle, que es para establecer sesiones entre dos clientes (por ejemplo para establecer una videoconferencia), usándose XMPP para la señalización y negociar el canal por el que irán los datos fuera de la conexión Jabber.

4.4.1. Qué servidor instalar

Hay varios servidores Jabber bajo licencia libre. Uno que viene con muchas distribuciones es jabberd, escrito en C. Uno de los más usados y fáciles de instalar (hay un vídeo de cómo instalar y empezar a usar este servidor en 180 segundos; en Debian y Ubuntu se instala con apt-get install ejabberd y ya está configurado para aceptar conexiones locales) es ejabberd (http://ejabberd.jabber.ru/). La única pega de ejabberd es que está escrito en Erlang, por lo que para quien no esté familiarizado con este lenguaje requiere cierto aprendizaje para escribir nuevos módulos. Otra opción es un servidor escrito en Java: OpenFire (anteriormente se llamaba WildFire): http://www.igniterealtime.org/projects/openfire/index.jsp.

Capítulo 5

Acceso remoto seguro

5.1. Distintas soluciones de acceso remoto

Para conectarse desde casa o desde otra ubicación a la red del trabajo, hay distintas alternativas:

1. Usar una VPN, de modo que nuestro ordenador de casa virtualmente esté en la red de la empresa, pero a través de una conexión cifrada vía Internet. Esta es la opción más potente, pero también la más compleja. Una empresa que tenga una política estricta de seguridad sobre las máquinas instaladas en la red, con el filtrado de red, se encontrará con que cada vez que se conecte el usuario se añade a su red local un nodo que puede estar en una red insegura (aunque esto depende también de la configuración, un equipo puede conectarse a la red remota y desconectarse del resto). Es una opción especialmente interesante cuando el usuario usa un portátil tanto cuanto está en la empresa como en su casa.

2. Conectarnos remotamente a nuestro escritorio del PC del trabajo, usando un túnel con SSH (también se puede hacer con SSL), con lo que sólo habría que abrir un puerto en el cortafuegos de la empresa. Es una solución más conservadora, pues virtualmente el usuario está usando su monitor, teclado y ratón para acceder a la máquina del trabajo, pero por ejemplo no puede ni transferir ficheros entre su equipo local y el de la oficina (aunque algunas soluciones de escritorio remoto sí lo permiten) o conectarse desde su equipo de casa a ningún servicio. Para el usuario también es una solución sencilla, pues está usando su equipo de la oficina en el que ya está todo el software instalado y configurado. Sin embargo no es una solución idónea si por ejemplo el usuario tiene que escribir un documento y luego enviarlo: para eso sería más razonable que lo pueda editar en su equipo y conectarse a la intranet de la empresa. Además esta posibilidad no está disponible cuando el trabajador que se conecta remotamente no tiene un PC de sobremesa en la oficina sino que usa un portátil que se lleva a casa.

   Técnicamente usando SSH se pueden abrir más túneles para por ejemplo acceder a la web interna o transferir ficheros, pero eso ya supone mayor conocimiento por parte de los usuarios y es así mismo posible limitar estas posibilidades. De todos modos en el momento que se permite al usuario crear un túnel cifrado potencialmente se le está dando acceso total a la red local de la empresa, pues el usuario puede utilizar el túnel para encapsular lo que quiera, incluyendo por ejemplo una sesión PPP para implementar así una VPN completa.

3. Usar lo que se ha dado en llamar, no con demasiada propiedad, SSL VPN. Es el caso del programa SSLExplorer Community Edition. Este tipo de solución es intermedia entre las dos anteriores. Ofrece una interfaz web, que completada con código Java permite navegar por el disco duro de la máquina remota, transferir ficheros entre las dos máquinas, navegar por la intranet de la empresa... Se implementa también con un túnel SSL y permite crear túneles para distintas aplicaciones y lanzar la aplicación en cuestión, por ejemplo rdesktop para conectarse a un escritorio remoto de Windows.

   Un detalle de nomenclatura, es que OpenVPN se define también como una SSL VPN pero no tiene nada que ver con esto, puesto que es una VPN real, que permite todo lo que ofrece cualquier otra VPN, mientras que las aplicaciones como SSLExplorer no son realmente VPNs. Se define como SSL VPN porque la VPN se encapsula en un túnel SSL y sólo hay que abrir un puerto.

5.1.1. Una solución práctica: crear un túnel con openSSH[1]

Supongamos que queremos conectarnos desde casa con el puerto 8080 en la máquina 192.168.1.100 de la red local de nuestra empresa. Para ello contamos con una máquina SSH que está dentro de la red local de la empresa y a la que llegamos a través de un DNAT que hemos hecho en el cortafuegos. En este caso hemos abierto el puerto 34321 del cortafuegos, que tiene la IP pública 87.15.10.120: los paquetes con este destino cambiarán de 87.15.10.120 a 192.168.1.1:22. Así pues, cuando nos conectemos vía ssh a 87.15.10.120:34321, llegamos en realidad al puerto SSH de 192.168.1.1.

La solución está en utilizar la posibilidad que ofrece SSH de crear túneles. SSH es un protocolo que no sólo cifra y garantiza la integridad de los datos, sino que es capaz de encapsular cuantas conexiones necesitemos en una sola. Así, nuestro propósito es que SSH cree un servidor local en nuestra máquina, por ejemplo con el puerto 8081, recoga todo lo que escribamos en él, lo envíe cifrado hasta la máquina 192.168.1.1 (la máquina remota a la que nos hemos conectado vía SSH) y desde allí los descifre y los envíe al puerto 8080 de la máquina 192.168.1.100; los paquetes de respuesta seguirán el proceso inverso, hasta llegar al puerto 8081 local de nuestra máquina. Observesé que en el caso del túnel los paquetes pasan por la capa de aplicación, mientras que en el NAT simplemente se modifican las cabeceras a nivel IP y TCP para hacer el NAT. La máquina 192.168.1.100 no verá como IP origen de los paquetes a la IP de nuestra casa, sino a 192.168.1.1, la máquina SSH que ha retransmitido el paquete.

En la práctica, hacer el túnel es tan sencillo como: **`ssh -N 801020.13 -p 34231 -o ServerAliveInterval 1200 -L8081:192.168.1.100:8080`**

La opción -N es para crear el túnel pero no iniciar una shell. La opción ServerAliveInterval es para enviar cada 2 minutos un paquete en caso de que la conexión esté inactiva, para evitar que el cortafuegos asuma que la conexión esté muerta y olvide la asociación NAT. Es posible añadir más opciones interesantes. Por ejemplo por defecto el puerto 8081 que ha creado ssh es sólo accesible dentro de nuestra máquina, si quisiéramos que pudieran conectarse a él otras máquinas de nuestra red local, añadiremos la opción -g.

Es posible también crear túneles en los que el puerto servidor se abre en la IP a la que nos conectamos remotamente (en este caso 192.168.1.1) y que se enviara a una IP y puerto de nuestra red local. Esto nos permite crear servidores en la red local de nuestra empresa (para que el puerto sea accesible desde todas las Ips de la red local de la empresa y no sólo desde 192.168.1.1, habrá que usar la opción -g).

Así, si queremos crear el puerto 8087 en 192.168.1.1 y que todo lo que se reciba allí vaya al puerto 9000 de la IP 192.168.7.3 (en nuestra red local de casa) usaríamos: **`ssh -N 801020.13 -p 34231 -o ServerAliveInterval 1200 -g -R192.168.1.1:8087:192.168.7.3:9000`**

Como puede verse, los túneles SSH son muy potentes y un tanto inquietantes desde el punto de vista de la seguridad para un administrador: podemos hacer un túnel para usar el escritorio remoto, para conectarnos a la web de la empresa, para enviar correos desde casa utilizando el servidor SMTP de la empresa, e incluso para ejecutar una aplicación en local que usa una BB.DD. de la red, cambiando la configuración para que use un puerto en nuestra máquina local y usando un túnel. Es más, es posible crear toda una VPN como veremos más adelante, con sólo la posibilidad de crear túneles de SSH.

Por este motivo el administrador de SSH puede desactivar el uso de túneles o especificar exactamente qué túneles se pueden usar, así como permitir o denegar obtener una shell o permitir ejecutar sólo un comando en concreto. Puede así mismo no permitir el acceso al sistema no dándole una contraseña y autenticando por clave pública: el propio fichero de clave pública almacenado en el servidor tendrá las restricciones que se aplican cuando se accede con esa clave, pudiéndose tener distintas claves para un mismo usuario según se le permita ejecutar una tarea u otra. El administrador también puede limitar lo que hagan los usuarios en el fichero de configuración global sshd_config mediantve la directiva Match, por usuario, grupo, dirección o host de origen.

La funcionalidad de SSH no acaba aquí. Es posible que un usuario se conecte a una máquina y desde ahí a otras. Si utiliza claves públicas para autenticarse, SSH incorpora un sistema, el ssh-agent, que permite autenticar al usuario estando conectado en otra máquina y sin que nuestra clave privada salga de nuestra máquina. Esta funcionalidad también es necesaria si se usa una tarjeta criptográfica, pues la clave privada no puede extraerse de la tarjeta ni siquiera por su propietario.

Para usuarios con Windows, en lugar de utilizar el cliente en línea de comandos pueden recurrir a Putty, un programa con interfaz de usuario basada en ventanas y que en realidad también se puede ejecutar en GNU/Linux.

Un pequeño detalle con SSH: el lanzar programas para ejecutar en modo batch (segundo plano) y que ssh retorne inmediatamente, sin esperar a que el programa lanzado termine. Si se abre una sesión y se lanzan con nohup funciona, pero si se invoca directamente con ssh un programa no; el cliente de ssh se queda esperando hasta que el programa acabe. El motivo por el que

[1] Openssh no es la única implementación libre de SSH. A destacar dropbear (http://matt.ucc.asn.au/dropbear/), usado sobre todo en sistemas empotrados como PDAs, teléfonos, routers por ser mucho más ligero que openssh... La mayoría de lo expuesto en este manual sobre openssh funciona igualmente en dropbear.

espera es porque la entrada/salida está redirigida a sockets y hasta que ssh no detecta que esos sockets están cerrados, no retorna. En la shell interactiva esto no pasa, porque al ejecutar nohup, aparte de no considerarse la señal HUP al morir la shell, que es algo que a SSH ni le va ni le importa (y que se puede hacer en cualquier momento aunque el programa no se lanzara con nohup, mediante el comando interno de la shell disown), ocurre que salida estándar y de error se redirige al fichero nohup.out y la entrada estándar a /dev/null, cerrándose la que hasta ahora era E/S estándar y de error, por lo que ssh ya no tiene que esperar. ¿Pero por qué la diferencia de comportamiento? La diferencia está en que en con una sesión interactiva se usa un ptty (un terminal virtual) mientras que con un comando aislado no, se usan los sockets directamente y nohup sólo hace la redirección si la E/S está asociada a un terminal, pues esa es su función, que el programa no utilice un terminal que ya no está disponible, mientras que sin son ficheros se supone que aunque muera la shell los ficheros siguen existiendo.

La solución por lo tanto está o bien en manualmente redigir la E/S estándar y de error a nohup.out o /dev/null, o invocar con la opción -t para que se cree un TTY virtual.

Capítulo 6

Redes Privadas Virtuales (VPN)

6.1. Soluciones VPN para GNU/Linux

VPN es el acrónimo en inglés de red privada virtual. El objetivo es lograr, mediante la creación de un túnel IP cifrado sobre una red pública como Internet, que los nodos a conectar por la VPN virtualmente estén en una misma red local. Es posible establecer una VPN directamente entre dos nodos, pero lo más habitual es o bien una VPN entre un nodo remoto y la red local de la empresa (esta configuración en inglés se llama con frecuencia road warrior, guerrero de la carretera, hace mención a que un usuario móvil se conecta a la empresa con su portátil y una conexión a Internet desde allí dónde está desplazado) o una VPN que une dos redes, por ejemplo dos filiales de una empresa o una filial con la central.

Hay distintas tecnologías y productos para crear una VPN con implementaciones libres para GNU/LInux:

1. IPsec: el estándar por excelencia, será obligatorio en IPv6 y está implementado en muchas pilas TCP/IP también para la actual generación IPv4, incluyendo el kernel Linux. Se implementa directamente sobre la capa 3 (IP) mientras que el resto de soluciones suelen implementarse a nivel de capa de aplicación (aunque conceptualmente PPTP y L2TP son protocolos de capa de datos, se implementan como túneles, en el caso de L2TP sí es sobre UDP, en el caso de PPTP se usa GRE, que es un protocolo sobre IP). A diferencia de otras soluciones que se implementan siempre mediante un túnel, IPsec permite tanto túneles (modo túnel, entre dos security gateways, SG) como directamente cifrar la capa de transporte (modo transporte), lo que es útil cuando sólo se quiere seguridad entre dos máquinas o ya se implementa el túnel sobre una capa posterior (típicamente la de aplicación, utilizando paquetes UDP, es el caso que enseguida veremos de L2TP sobre IPsec).

2. PPTP: protocolo impulsado por Microsoft. Su implementación original tenía varios fallos graves y además existe el problema de que no está tan probado como IPsec siendo también relativamente compleja. No es un estándar, aunque se publicó un RFC informativo describiendo el protocolo. El nombre completo es Point to Point Tunneling Protocol. Hay clientes de PPTP en Windows desde Windows 95, aunque Microsoft trata ahora de substituir PPTP por LLTP e IPsec, si bien PPTP ha tenido cierto éxito por su sencillez de configurar: no se usan claves públicas, sino las propias cuentas de Windows para autenticar el acceso.

   Hay implementaciones para GNU/Linux interoperables con la implementación de Microsoft tanto del cliente (http://pptpclient.sour
) como del servidor (http://pptpclient.sourceforge.net/) . A destacar que los desarrolladores de estas implementaciones recomiendan no usar PPTP más que si no queda otro remedio: en su lugar recomienda OpenVPN o IPsec.

   En Ubuntu 7.10, el cliente PPTP forma parte de los paquetes oficiales. Hay además un applet para network-manager, aunque este ya no es oficial.

3. L2TP: Layer 2 Tunnel Protocol: promovido por Microsoft y CISCO, sigue la vía de estandarización del IETF (es el RFC2661, con la versión 3 más reciente en el RFC 3931). Es un protocolo que crea un túnel a nivel de la capa de aplicación (mediante paquetes UDP, aunque en principio puede encapsularse sobre otro tipos de redes no IP) para transportar trápico layer2. Mediante L2TP se crea un túnel entre el LAC (L2TP Access Concetrator) y el LNS (L2TP Network Server): una vez creado el túnel, cualquiera de las dos partes puede abrir sesiones (también denominadas en L2TP como llamadas) y estas sesiones se multiplexarán sobre el túnel; un uso típico es crear una sesión PPP. L2TP garantiza fiabilidad en los paquetes de control, mientras que en los de datos como es propio de UDP podrán perderse. Podemos pensar en el LAC como la parte cliente de la VPN, aunque hay una configuración, compulsory tunnel remote dial en que no es así; para ser

más precisos normalmente funciona siendo el software al que se conecta el cliente PPP para que la encapsule en un túnel L2TP, en algunas implementaciones en realidad el cliente PPP y el LAC se implementan con un solo programa, pero hay configuraciones en las que el LAC lo implementa el ISP y otro caso típico es que lo implemente un router en la red local. En este caso el ordenador del usuario no tiene que saber nada de L2TP, sino que utiliza por ejemplo PPPoE para conectarse con el router o con el ISP, estando ese tramo sin cifrar.

En realidad L2TP no implementa por sí sólo una VPN, dado que no cifra la información, por lo que o bien usa PPTP o lo recomendado, usa IPsec (es decir, se establece un transporte seguro entre los dos extremos y a su vez se encapsula el túnel L2TP: esto también se estandariza por el IETF, en el RFC).

¿Qué sentido tiene usar L2TP sobre IPsec, pudiendo usar directamente IPsec? una razón es usar la autenticación PPP (por ejemplo un servidor Radius) y la parte de autoconfiguración (por ejemplo proporcionar la información DNS); no obstante con IPsec también es posible autenticación y asignar IP y DNS sin necesidad de L2TP. Así mismo L2TP crea un túnel a nivel 2, por lo que puede ir todo tipo de tráfico, no necesariamente IP, por ejemplo puede ir tráfico IPX o incluso se puede transportar directamente paquetes de nivel 2 como ATM. Así mismo L2TP en gran medida es más interesante para proveedores de Internet que tengan que encapsular información (por ejemplo puede multiplexar varias sesiones L2TP sobre un túnel L2TP/IPsec, o aprovechar esta funcionalidad para revender ancho de banda), que para VPNs entre oficinas, pero en cualquier caso se agradece que haya implementaciones para GNU/Linux pues habrá servidores que no permitan conectarse directamente usando IPsec (quizás por desear usar la autenticación PPP) y requieran L2TP. En concreto usan L2TP además de Microsoft, CISCO y MacOSX . En el artículo de la Wikipedia hay enlaces a las implementaciones existentes así como a un tutorial: http://en.wikipedia.org/wiki/L2TP. También se comenta la dificultad con Windows de usar IPsec sin L2TP.

4. OpenVPN: es la solución recomendada, por su sencillez: hace un túnel TLS sobre el que se encapsulan paquetes layer 3 (IP) o layer 2 (Ethernet), pero usando UDP en lugar de TCP, aunque también se puede usar sobre TCP e incluso pasar a través de proxies HTTP. En realidad TLS no se puede implementar directamente sobre UDP, porque requiere un protocolo fiable como TCP y con UDP se tiene que permitir la pérdida de paquetes. Por ello en realidad TLS se usa para la funcionalidad de control, como negociar la sesión TLS o renegociarla en un momento dado (implementando sobre UDP un sistema para pedir de nuevo los paquetes que se puedan perder y son importantes por ser de control), sustituyendo a los protocolos PKI de IPsec. Pero además se han imitado los mecanismos presentes en las cabeceras [1] de IPsec para dar seguridad a nivel de paquete y con independencia de si se pierde el paquete anterior: para ello hay que añadir una secuencia a los paquetes, así como un HMAC y contemplar varios mecanismos. Una idea similar es la que se sigue en la implementación del estándar DTLS (Datagram TLS: http://es.wikipedia.org/wiki/Datagram_Transport_Layer_Security) que se define en el RFC4347 y cuya única implementación actual conocida es la que hace el proyecto OpenSSL. En http://unix.freshmeat.net/projects/-dtls_example/ hay código de ejemplo. Es posible que en un futuro se modifique OpenVPN para usar DTLS en lugar de su propio protocolo.

5. Montar una VPN sobre la marcha con OpenSSH y pppd o OpenSSH con TUN. Es una solución de compromiso ante una necesidad puntual. Si hemos instalado SSH y dejamos que pase el cortafuegos por si necesitamos conectarnos o lo usamos para cifrar un tunel con la conexión el acceso a escritorio remoto, en un momento dado podemos instalar una VPN. La idea es utilizar el software de PPP (pppd) para crear los dispositivos de red en cada extremo y comunicarlos por un túnel SSH: el software PPP escribe por salida estándar lo que lee del dispositivo de red e igualmente lo que lee por salida estándar lo pone en el dispositivo de red. Con SSH se pueden comunicar las entradas salidas de dos programas, formando un túnel. Si lo preferimos, podemos hacer el túnel utilizando Stunnel en lugar de OpenSSH. Otra opción es utilizar la opción de hacer un túnel con TUN de SSH. Una ventaja es que es algo más sencillo, se puede lanzar sin privilegios y existe TUN incluso para Windows, aunque no por ejemplo para Mac OS X. Otra ventaja es que se puede usar TAP en lugar de TUN y usarlo para hacer un bridge: pppd necesita parchearse para funcionar en un bridge (soporte BCP), pues no transmite tramas Ethernet, sino por defecto tramas IP (aunque en Linux puede transmitir IPX en lugar de IP). Otra utilidad que puede hacer el papel de openssh +TUN pero más ligera es socat (la mencionamos al hablar de netcat). Socat es una utilidad realmente versátil que también puede reemplazar a stunnel.

6. Usar un cliente sobre la VPN proporcionada por un encaminador CISCO. El cliente, al igual que OpenVPN, no necesita de ningún soporte en el kérnel. salvo TUN/TAP. Web: www.unix-ag.uni-kl.de/~massar/vpnc/

[1] IPsec define dos protocolos para dar seguridad a los paquetes, AH y ESP. AH garantiza la integridad y previene replay attacks, mientras que ESP además de eso cifra. AH garantiza la integridad también de las cabeceras (por lo que no puede usarse con NAT), pero en general no es muy útil para una VPN frente a ESP, pues normalmente IPsec se usa en modo túnel, no en modo transporte, y en este caso las Ips origen y destino van dentro del paquete. No obstante sí podría ser útil para que los routers estén seguros que el tráfico proviene de determinado router, suponiendo que se use un certificado que reconozcan y que efectivamente los "secure gateway", es decir, los extremos del túnel, coincidan con los enrutadores. En modo transporte sin AH la IP de origen se puede falsificar. En modo túnel se podría falsificar la dirección del router origen o destino entre los que se establece la VPN, pero no las IP origen y destino de los paquetes, que es lo que importa a los usuarios finales. Si se desea, se puede encapsular ESP en un paquete AH, pero rara vez se hace. Bruce Schneier recomienda simplificar IPsec y eliminar el modo de transporte y AH. dejando sólo ESP y modo túnel.

7. En grandes empresas, usar la infraestructura de VPN que ofrezca el operador para unir dos sedes (no es por tanto solución general para teletrabajo, especialmente si no tiene por qué coincidir el operador), de modo que se tiene una red privada virtual como si se tuviera una línea dedicada o una conexión dedicada a través de una red de conmutación de circuitos como las líneas telefónicas analógicas. Típicamente esto es posible si el operador utiliza MPLS, u otras tecnologías más antiguas como ATM o Frame Relay. Obviamente, requiere que el operador comercialice la posibilidad de conectar dos sedes entre sí en lugar de sólo dar servicio a Internet.

El paquete NetworkManager (www.gnome.org/projects/NetworkManager/) dispone de plugins para OpenVPN, PPTP y VPNC (para este último programa soporta los ficheros .pcf, que sería lo único que tendría que distribuir el administrador). En el caso de la versión incluida en Ubuntu 7.10, sólo aparece la opción de configurar VPN si la tarjeta está en modo "itinerante", lo que excluye la configuración manual.

6.2. IPsec

Hay implementaciones de IPsec para cualquier sistema Unix, MacOS X, Windows XP/2000... Una implementación de IPsec consta de dos partes: la implementación sobre la pila de protocolos TCP/IP, en el kérnel y el demonio y herramientas que se ejecutan en el espacio de usuario para establecer las claves[2] (esto es la parte IKE: Internet Key Exchange[3]). En principio es posible utilizar la parte IKE de una implementación sobre la pila de protocolos de otra. Así, la implementación integrada en el kernel Linux (llamada NETKEY) no incluye herramientas IKE, por lo que hay que usar las de otra implementación: habitualmente se usa la implementación del proyecto Kame (surgido de un consorcio de empresas japonesas como Fujitsu y Toshiba para hacer una implementacion de Ipv6 e IPsec), que es la implementación usada en Free/NetBSD y en MacOS X; el demonio IKE de Kame se llama Racoon y el conjunto de utilidades portadas para Linux que además de Racoon incluye otras como setkey es IPsec-tools (http://IPsec-tools.sf.net); en Ubuntu Dapper existen por separado los paquetes IPsec-tools y Racoon (sólo necesitamos el demonio si la asociación de seguridad se negocia vía IKE (puerto UDP 500) entre las dos partes, no lo necesitamos si la SA se establece manualmente manipulando la SAD con setkey.

Setkey puede ser necesaria aunque se use IKE, para establecer la política (la SP), pues esta herramienta es tanto para manipular la SAD como la SPD (basa de datos de asociaciones de seguridad y políticas de seguridad, respectivamente). Las reglas de las políticas recuerdan las de iptables: establece que por ejemplo para comunicaciones entre dos rangos de IP, usando determinados protocolos, debe usarse un túnel con esp. Sin embargo la política se puede crear directamente sin necesidad de setkey desde un programa a nivel de conexión, mediante setsockopt y libipsec (llamada ipsec_set_policy), librería incluida en el paquete racoon pero que por supuesto sólo es válida para esta implementación: no está estandarizada la sintaxis de las políticas.

También se puede usar por ejemplo como demonio IKE isakmpd (de la implementación de OpenBSD) o Pluto. Pluto es el demonio IKE de la antigua implementación de IPsec para Linux, FreeSWAN, que nunca llegó a formar parte del kernel; FreeSWAN dejó de desarrollarse pero tomaron el relevo dos proyectos: OpenSWAN (www.openswan.org) y StrongSWAN (www.strongswan.org). Mientras que OpenSWAN sigue desarrollando la pila de protocolos KLIPS como alternativa a la integrada en el kernel, NETKEY(pero su parte IKE funciona tanto con KLIPS como con NETKEY), en StrongSWAN ya sólo soportan NETKEY. StrongSWAN es compatible no sólo con IKE1 sino con IKE2[4]: de la compatibilidad con IKE1 se encarga el viejo demonio Pluto, mientras que para IKE2 hay un nuevo demonio llamado Charon. La flexibilidad de programas como StrongSWAN o OpenSWAN son inmensas: incluyendo toda una infraestructura de clave pública basada en certificados X.509, con la posibilidad de utilizar dispositivos como smartcards y teniendo en cuenta aspectos como que los certificados pueden revocarse y ya no ser válidos, que pueden obtenerse de servidores LDAP... Otra característica interesante es "oportunistic encryption", que se basa en situar los certificados en servidores DNS. Ahora bien, para que esto sea seguro, hay que usar DNSSEC, pero esta especificación está poco implementada y hay problemas de seguridad y privacidad por permitir obtener todas las entradas DNS de un servidor[5].

[2] En terminología IPsec se habla de Security Associations (SA) que incluyen los parámetros de seguridad como claves, certificados, algoritmos, duración de claves, datos para evitar replay attacs, etc.. En una comunicación hay una SA por sentido. En los paquetes se incluye un número, el SPI (Secure parameter index) que junto con la IP destino y el protocolo (AH o ESP) sirve para que la otra parte encuentre la SA en una tabla, la SADB. En cada nodo hay una SPD (Security Policy Database) que básicamente es una tabla que dados unos selectores (Ips/puertos origen/destino, por ejemplo) aplica una acción, que puede ser dejar pasar el paquete tal cual o usar AH/ESP, modo transporte o modo túnel y la SA a aplicar. La implementación de Linux almacena en el kernel estas dos tablas.

[3] En la versión anterior de los RFC de IPsec, en realidad se hablaba de ISAKMP, para la gestión de las SA, que para la parte del intercambio de claves podía usar distintos protocolos, aunque en la práctica se establecía IKE. En la versión actual, IKE2, engloba todo.

[4] StrongSWAN no es la única implementación libre de IKE2. Existe también racoon2, ikv2 y el proyecto español openikev2 (http://openikev2.sourceforge.net/)

[5] Hay más detalles de esto en la wikipedia. La idea básica es que los errores de no encontrado también hay que firmarlos, pero como la clave privada no tiene por qué estar online, estos mensajes pueden estar prefirmados; para que estos mensajes sean finitos, se indica que entre tal nombre y tal otro no existe esa entrada.

Para usar IPsec hay que abrir en los cortafuegos el puerto 200 UDP, pues lo usa IKE. También hay que tener en cuenta que los paquetes tienen un número de protocolo distinto al IP (en concreto usa 50 y 51), por lo que también podría haber problemas en el cortafuegos.

El lado negativo de IPsec es que es una solución muy compleja[6], sobre todo lo que tiene que ver con la parte IKE. Al margen que esa complejidad podrá afectar más o menos al usuario según los asistentes para configurar una VPN (el problema está en que todo se complica más si los sistemas operativos de los nodos a conectar son distintos[7], frente a la sencillez de usar una implementación de VPN más simple que tenga versión para distintos sistemas operativos) desde el punto de vista de la seguridad son preferibles las soluciones sencillas. Hay gente que no le gusta tampoco de IPsec frente a otras soluciones que se ejecute mucho código en el espacio del kernel (por ejemplo la parte de cifrado, que en el caso de OpenVPN se ejecuta en el espacio de usuario) dado que un error de seguridad ahí es fatal al comprometer todo el sistema, si bien tiene un impacto positivo en el rendimiento. Esta objeción es opinable: es cierto, pero también la información sensible está más protegida a nivel del kernel.

6.2.1. Caso práctico: IPsec en modo transporte, usando preshared keys

La IP de nuestra máquina es 10.0.0.1 y queremos que haya conexiones seguras con otra máquina de la red local, la 10.0.0.2. Vamos a crear la SA (Security Association) directamente con setkey, por lo que no necesitamos el demonio racoon, dado que no usamos IKE. Usaremos PSK (pre-shared keys). Para ello, creamos el siguiente fichero y lo grabamos como "politica.txt":

```
add 10.0.0.1 10.0.0.2 esp 7320 -E 3des-cbc "i92dfafgehogexxxx1112222";
add 10.0.0.2 10.0.0.1 esp 6698 -E 3des-cbc "xxxXXXXXXrjZZZ(xx111z<<3";
spdadd 10.0.0.2 10.0.0.1 any -P out ipsec esp/transport//require;
```

Los números 7320 y 6698 son los SPI para identificar la SA; nos basta con elegir un entero de hasta 32 bits al azar, sabiendo que del 0 al 255 están reservados y que estos números tienen que ser únicos por cada nodo.

La clave en cada sentido la hemos especificado para 3DES, que son 192bits, o lo que es lo mismo, 24 bytes; si especificamos la cadena como texto (otra opción es como número hexadecimal, comenzando por 0x y sin entrecomillar) deberá tener exactamente esa longitud.

Hay que ejecutar en la máquina **`setkey -f politica.txt`**. Tras transferir el fichero a la máquina 10.0.0.2 haremos lo mismo, pero tras invertir las direcciones en la última línea del fichero.

Más casos prácticos, en el HOWTO: http://lartc.org/howto/lartc.ipsec.html

6.3. SSH + PPPD / SSH + TUN

¿Qué inconvenientes tiene usar OpenSSH + PPPD o TUN? el principal inconveniente es que no es buena idea implementar el túnel de una VPN sobre TCP, es mejor utilizar UDP. El motivo es porque TPC tiene mecanismos de control de congestión que no funcionan de la mejor forma cuando una conexión TCP a su vez se encapsula sobre una conexión TCP: se produce un efecto multiplicativo que hace que ante una congestión se reduzca la velocidad enormemente y que tarde mucho en coger de nuevo velocidad. Hay una explicación de este fenómeno en http://sites.inka.de/sites/bigred/devel/tcp-tcp.html. Otro argumento para usar UDP es que podemos tener aplicaciones UDP que usan este protocolo por la necesidad de que los paquetes lleguen siempre a tiempo, sin importar si se pierde alguno: como se usa TCP, los paquetes nunca se descartan y se gasta tiempo en retransmisiones.

Paradójicamente, hay casos en los que usar un túnel TCP sí puede tener alguna ventaja sobre un túnel UDP. Si un protocolo UDP está mal diseñado frente a un ataque que selectivamente hace que determinados paquetes se pierdan, el que vaya sobre TCP sería una ventaja puesto que entonces TCP nos garantiza que ningún paquete se pierda ni llegue fuera de orden.

[6] Hay 9 documentos para describir IPsec (RFC 4301-4309, de Diciembre de 2005, todavía es frecuente encontrar referencias a la versión anterior de estos documentos, RFC 2401-2409 y aún hay otra más antigua, totalmente incompatible) más otros complementarios para explicar cómo traspasar los cortafuegos, así como por ejemplo para especificar IPKMOBIL (para clientes con IP móvil y para máquinas multihome) . De todos modos IKE2 simplifica IKE, a la par que evita algunos de sus problemas, en concreto denegaciones de servicio. Si oímos hablar de cosas como main vs agressive mode, se refieren a la versión vieja, IKE.

[7] Los problemas de interoperabilidad se dan practicamente siempre en la parte IKE. Son muchos los parámetros negociables y no hay unos valores por defecto que todas las implementaciones tengan que proporcionar. En la mayoría de los casos si algo sale mal no se muestra claramente la causa y si se muestra no es comprensible para alguien que no conozca a fondo IPsec. Para complicar más las cosas, IKE2 no es compatible con IKE1, aunque se pueden implementar los dos usando el mismo demonio y puerto UDP. Así mismo hay extensiones, por ejemplo en IKE con X-AUTHse pueden utilizar mecanismos adicionales de autenticación; en IKE2 se usa EAP (como en WPA/WPA2-Enterprise en las redes Wifi y en general en dispositivos que usen la autenticación de IEEE802.1X).

Otro inconveniente es que PPPD requiere privilegios de superusuario, aunque se pueden hacer cambios en la configuración para que no sea necesario, atendiendo a que el programa se ejecuta con el bit setuid. Crear un dispositivo TUN/TAP también requiere privilegios, pero es distinto, porque una vez creado con un programa, se puede hacer que esté disponible para otros programas lanzados sin privilegios.

6.3.1. Caso práctico: SSH + PPPD

Supongamos que la IP pública dónde escucha el servidor SSH es 157.88.66.102 y el puerto es 8088 . Vamos a crear una conexión punto a punto, en el que la IP local del enlace punto a punto (esta IP será la que tendrán como origen los paquetes lleguen a la red remota a través del túnel y tengan como origen la máquina local) será 172.16.16.15 y el remoto 172.16.16.16. Realmente podemos escoger el par de Ips que queramos, con tal que la IP remota no esté en uso en la propia red, teniendo en cuenta que en el otro extremo la IP local será la remota y por lo tanto la que no tiene que estar presente; no pasa nada porque la IP local sea la IP asignada a otra interfaz de red de la máquina. Ejecutamos: **`pppd updetach nobsdcomp nodeflate usepeerdns noauth connect-delay 10000 pty "ssh root@157.88.66.102-o ServerAliveInterval=120 -p 8088 -t -t pppd noauth 172.16.16.16:172.16.16.15"`**

La opción updetach es para que el programa pase a segundo plano tras lograr ejecutar la conexión (es decir, se ejecuta como si fuera un demonio, liberando el terminal). Las opciones nobsdcomp y nodeflate son para evitar que se use compresión, que no se llevaría bien con el usar un túnel SSH. Basta ponerlas en el lado cliente, pues se negocian para los dos lados de la comunicación. La opción "usepeerdns" es para que se actualice nuestro /etc/resolv.conf con los datos del servidor DNS del extremo con el que conectamos; a fin de usar su DNS en lugar del nuestro. Sin embargo es posible que esta opción no nos funcione, pues requiere que haga su trabajo el script local /etc/ppp/if-up, que no siempre tiene en cuenta estos datos, por lo que es posible que tengamos que modificarlo manualmente. La opción noauth se usa para que no use la autenticación PPP: no hace falta, dado que ya usamos la de SSH. La opción connect-delay en este caso indica que no intente establecer la conexión PPP hasta transcurridos 10 segundos; el motivo es que el pppd en el otro extremo se ejecuta al lograr autenticarnos con el servidor SSH, para lo que tendremos que introducir la contraseña antes; es decir, nos damos 10 segundos de margen para lograr ejecutar en el otro extremo pppd vía ssh. Finalmente la opción pty es para usar como script de conexión, con el se comunicará usando un pseudo-tty, lo que va entrecomillado, que no es otra cosa que lanzar pppd en la máquina remota. Entre las opciones de SSH, a destacar ServerAliveInterval: le estamos indicando en este ejemplo que envíe un paquete cada 2 minutos si no se transmiten datos, a fin de garantizar que el posible NAT que haga algún enrutador no se pierda por considerar que la conexión ya no está activa.

En realidad es posible simplificar la instrucción anterior, así como mejorar la seguridad, usando para autenticarse con el servidor un certificado y configurando en el servidor que al conectarse con ese certificado en lugar de poder ejecutar cualquier programa ejecute una orden determinada, que en nuestro caso sería la invocación a pppd.

Tras ejecutar esta operación, tendremos un nuevo dispositivo ppp0, con IP local 172.16.16.15 e IP remota 172.16.16.16. Para llegar a las máquinas remotas caben dos opciones: una es que perdamos la conectividad local (nuestro acceso a Internet), usando como ruta por defecto 172.16.16.16; esto lo podemos hacer con **`route del default && route add default gw 172.16.16.16`**, aunque también podíamos haber pasado directamente la opción "defaultroute" a pppd para que hubiera aplicado este cambio automáticamente. Sin embargo, al ejecutar esta operación nos encontramos con un problema: no funciona, porque hasta el servidor SSH tenemos que llegar usando nuestra antigua ruta por defecto (la de nuestro router que nos permite salir a Internet). Así pues, suponiendo que nuestro entrutador a Internet es 192.168.1.1, añadimos una ruta: **`route add host 157.88.66.102 gw 192.168.1.1`**

La otra opción es que sólo añadamos una ruta específica para llegar a las Ips de la red remota y que para la ruta por defecto sigamos usando la ruta que tenemos. Así, si la red remota es 192.168.10.0, ejecutaremos: **`route add -net 192.168.10.0 -netmask 255.255.255.0 gw 172.16.16.16`**

Esta segunda alternativa encierra más riesgo de seguridad, pues si hay agujeros en nuestra máquina y estamos conectados a Internet, estaremos arriesgando la red remota, pues los paquetes que vienen de nuestra máquina no pasan por el cortafuegos de la red remota.

Con la IP que hemos escogido como dirección local, los paquetes que enviemos llevarán como IP origen 172.16.16.15 (la IP local en el enlace punto a punto); posiblemente en la red destino no sepan cómo llegar a esta IP desde otro nodo que el del servidor SSH, por lo que sería una conexión host a host, salvo que precisamente la máquina del servidor SSH sea el enrutador por defecto. Aún así, si por ejemplo este router a su vez pasa al testigo a otro para llegar a Internet o determinadas redes, con esa IP no llegaremos porque no se esperan que la IP 172.16.16.15 esté en ese router.

Hay dos formas de solucionarlo. Una es utilizar como IP local una IP de la red destino que no se esté usando y pasar al servidor la opción proxyarp. En ese caso cuando desde un nodo de la red local se pregunte por la IP, responderá la máquina donde está el servidor SSH con su propia dirección ARP y hará una redirección. La otra solución es hacer NAT; supongamos que la IP en la red remota del nodo que es servidor SSH es 192.168.10.7; entonces habría que ejecutar en la máquina remota (para lo cual podemos usar nuestro ssh) la orden para añadir la regla NAT: **`ssh 81.33.18.197 -p 9436 iptables -t nat -A POSTROUTING --source 172.16.16.15 -j SNAT --to-source 192.168.120.19`**

Esta solución tal como está es sólo para los paquetes procedentes del equipo local: si lo que hemos hecho es usar el túnel VPN para unir a la red remota toda la red local ya es más complicado, pues la IP origen será la de la máquina de procedencia.

6.3.2. Más sencillo: servidor SOCKS de SSHD

En realidad, si se tiene acceso a una máquina de la red local remota vía SSH, hay un método más sencillo para que nuestras aplicaciones de red accedan a los nodos de la red remota como si nuestra máquina estuviera allí. Se trata del servidor de SOCKS5 que integra OpenSSH. Este servidor se arranca en nuestra máquina local en el puerto que le indiquemos con la opción -D. Cualquier cliente que utilice ese proxy SOCKS accederá a la red remota, pues el proxy hace las peticiones en la máquina remota: todas las peticiones y respuestas van tuneladas y cifradas en la conexión SSH. De este modo, es como si las aplicaciones pudieran crear túneles SSH a petición, pero utilizando un protocolo estándar, SOCKS.

Una característica interesante de SOCKS5 es que para establecer una conexión se puede pasar al proxy en lugar de la IP directamente el nombre de la máquina. Esta posibilidad es interesante, porque así usamos el DNS de la red remota, en lugar de el nuestro local, algo muy deseable si es un DNS privado.

Hay aplicaciones que se pueden configurar para usar SOCKS, como Firefox. No obstante Firefox por defecto no usa el DNS remoto. Para que sí lo haga, bastará con teclear about:config y cambiar la opción network.proxy.socks_remote_dns.

Muchas aplicaciones no soportan SOCKS. La buena noticia es que se pueden "sockificar": la idea es reemplazar en tiempo de ejecución la librería estándar de C para que ante un connect o un bind llame en su lugar al proxy SOCKS. La única pega es que esto no sirve para usar el servidor DNS remoto, aunque la alternativa es abrir un túnel con SSH al servidor remoto y usarlo en lugar del local. En algunos casos parece que también resuelve el caso del DNS, alterando la llamada para hacer la consulta para hacerla a través del servidor SOCKS. Ejemplo (sólo para TCP) http://proxychains.sourceforge.net/. Más programas en http://en.wikipedia.org/wiki/SOCKS.

"Sockificar" una aplicación es muy sencillo en cualquier programa que no tenga bit setuid: basta con usar LD_PRELOAD.

6.4. SSH con TUN

Las versiones recientes de SSH permiten crear túneles TUN o TAP de forma muy sencilla. Para ello hay que comenzar por modificar sshd_config para autorizar que se creen túneles. Se trata de la directiva PermitTunnel y deberemos dar valor yes para autorizar ambos tipos de túneles, "point-to-point" para autorizar sólo los TUN y ethernet para autorizar sólo los TAP.

Sólo el usuario root puede crear túneles: tiene que ser root tanto quien ejecute el cliente ssh, como la cuenta destinataria a la que conectar. El túnel existe mientras dure la conexión.

El cliente puede indicar que quiere crear el túnel de tres maneras distintas:

1. Mediante la opción -w <número_dispositivo_tun_local>[:<número_dispositivo_tun_remoto>, por ejemplo -w 0,0 si queremos que tanto en local como en remoto se use tun0.
2. Indicando en ssh_config que queremos crear el túnel al conectarnos como root. Hay que rellenar la directiva Tunnel (mismos valores que PermitTunnel en el servidor) y TunnelDevice, con la misma sintáxis que la opción -w
3. Usar un certificado con la opción tunnel="<núm_dispositivo>", idealmente además fijado el comando a ejecutar a uno que configure el dispositivo TUN, dado que todas estas opciones crean los dispositivos pero no les asignan una IP ni les configuran en absoluto.

6.5. OpenVPN

El hecho de usar protocolos conocidos y una librería muy probada como OpenSSL da mucha fiabilidad a este programa: es relativamente fácil fallar estrepitosamente a la hora de tratar de crear un nuevo sistema de VPN, como revela este interesante escrito de Peter Gutmann (el autor de cryptolib y la impresionante presentación/tutorial de criptografía de más de 800 transparencias)

que habla de CIPE, VTUN y TINC: http://www.cs.auckland.ac.nz/~pgut001/pubs/linux_vpn.txt. Del mismo autor, un artículo más elaborado sobre el tema: http://www.linux-magazine.com/issue/39/VPN_Insecurity.pdf

OpenVPN consta de un único ejecutable (para ambos extremos es el mismo) y un fichero opcional de configuración, además de los ficheros de claves.

Uno de los puntos fuertes es que es multiplataforma: el poder usar exactamente el mismo programa en GNU/Linux, cualquier BSD, Solaris, Mac OS X o Windows XP/2000 supone hacer una VPN en la que intervengan equipos con cualquier sistema operativo sin problemas de interoperabilidad.

OpenVPN permite hacer túneles a nivel de la capa 3 (la opción por defecto y la recomendada, se usa un dispositivo TUN) o a nivel de la capa 2 (usando un dispositivo TAP). Si usamos la capa 2, podemos utilizar un puente. Sólo tiene sentido usar la capa 2 en casos especiales, principalmente si necesitamos un puente o vamos a transferir distintos tipos de tráfico (IP, IPX...) simultaneamente. Por ejemplo es útil utilizar un puente para tener una IP propia asignada por un servidor DHCP en la red remota, aunque es posible lograrlo (con más trabajo) utilizando proxyarp o usar un proxy de DHCP.

Una funcionalidad interesante de OpenVPN desde su versión 2.0 es crear un único servidor al que pueden conectarse distintos clientes, para crear cada uno su propia red privada virtual con la red local en la que está el servidor; en este modo además el servidor puede pasar configuración al cliente, incluyendo opciones DHCP. El modo servidor de OpenVPN requiere el uso de certificados, en cambio en el modo tradicional (peer2peer) se pueden usar certificados o claves compartidas; en este último caso no se utiliza TLS ni criptografía de clave pública.

OpenVPN no soporta STUN ni TURN para superar problemas en el caso de que las dos partes estén tras NAT, pero en su defecto soporta SOCKS, como proxy UDP.

OpenVPN se adapta muy bien al caso de sitios con IP cambiante que usan DNS dinámico. Tiene una opción para reiniciarse si la otra parte no responde a unos pings periódicos, para volver a establecer la conexión.

El programa incluye un gran número de opciones: limitar ancho de banda, autenticar por usuario y contraseña además de por certificado, listas de revocación de certificados... Así mismo se puede usar con una smartcard (actualmente en la versión beta sólo).

6.5.1. Caso práctico: OpenVPN

Supongamos que queremos conectarnos las máquinas con IP públicas 83.59.36.220 y 81.33.18.197. La primera tiene abierto en el cortafuegos el puerto UDP 9037, mientras que la segunda tiene el 1198. Vamos a crear una red privada virtual, en el que el primer nodo tendrá la IP 10.4.0.2 y el segundo el 10.4.0.1.

La configuración más sencilla de manejo de una clave común es utilizar una clave compartida. Para ello en primer lugar la generamos en uno de los nodos: **`openvpn --genkey --secret static.key`**

A continuación la subiremos al otro nodo utilizando algún procedimiento seguro. Por ejemplo con scp (parte de SSH). Hecho esto ya podemos crear la VPN:

En el nodo 81.33.18.197, ejecutamos: **`openvpn --remote 83.59.36.220 --rport 9037 --lport 1198 --dev tun1 --ifconfig 10.4.0.1 10.4.0.2 -secret static.key 0`**

En el nodo 83.59.36.220 ejecutamos: **`openvpn --remote 81.33.18.197 --rport 1198 --lport 9037 --dev tun1 --ifconfig 10.4.0.2 10.4.0.1 --secret static.key 1`**

El número que sigue a static.key simplemente tiene que ser distinto en cada lado; se utiliza para que se utilicen dos pares de claves en la sesión en lugar de un solo par.

En ambos nodos ejecutamos: **`echo 1 > /proc/sys/net/ipv4/ip_forward & iptables -I FORWARD -i tun+ -j ACCEPT`**

La primera línea es para activar el forwarding entre interfaces de red, o lo que es lo mismo, la posibilidad de enrutado. La segunda línea es para evitar que rechace el cortafuegos el tráfico procedente de la interfaz tun. En esta línea hemos puesto I

(Insert) en lugar de A (append) como viene en la documentación. Realmente es más apropiado append, puesto que así si hay reglas específicas para filtrar no nos las saltaremos, pero para probar primero que podemos establecer conexión puede ser buena idea poner la regla la primera.

Una vez que veamos que todo funciona, podemos invocar estos programas con la opción --daemon.

Si no hemos logrado conectar, posiblemente haya un problema con el cortafuegos. Podemos probar con la herramienta netcat a enviar/escuchar paquetes en estos puertos UDP.

Así, en 83.59.36.220 ejecutamos: **`netcat -u -l -p 9037`**

en 81.33.18.197 ejecutamos: **`netcat -u 83.59.36.220 9037`**

A continuación escribimos algo: deberá aparecer en 83.59.36.220 y lo que escribamos allí aparecer en 81.33.18.197.

El uso de una clave compartida no es el método más seguro. Por ejemplo si se compromete la clave se comprometen todas las conversaciones pasadas. No es un sistema de "perfect forward security". Aclaremos que hay sistemas basados en PSK (Preshared keys) que sí lo son, pero no así el de OpenVPN. La diferencia es que en algunos sistemas las PSK hacen las mismas funciones que los certificados en TLS: sirven para autentificar a la otra parte, no para obtener la clave de sesión, que se determina utilizando Diffie-Hellman. En cambio con OpenVPN sirven para establecer la clave de sesión.

Es posible usar proxyarp también con OpenVPN, lo que ocurre es que la forma de activarlo sí que es dependiente del sistema operativo y habrá sistemas que ni siquiera lo soporten. En el caso de Linux sería con:

`echo 1 > /proc/sys/net/ipv4/conf/all/proxy_arp`

Por supuesto también es posible activar proxy_arp a nivel de cada dispositivo de red, igual que en el caso del fordwarding. Como en el caso del forwarding, deben activarse todas las interfaces implicadas, esto es, tanto la interfaz por la que llega la pregunta como la interfaz de la que se recibe la respuesta.

No es necesario especificar --remote y --rport en los dos extremos, basta con hacerlo en uno (esto es útil para especificarlo sólo en el extremo que no tiene un puerto abierto en el router, se recomienda en este caso usar así mismo --nobind). El extremo en el que lo especifiquemos trata de contactar con el otro (se puede indicar varias veces con datos distintos y lo intentará secuencialmente con cada uno hasta lograrlo). Ahora bien, si lo especificamos OpenVPN dará un error si recibe un paquete de una IP o puerto distinto a alguno de los especificados. Si la IP de la otra parte puede cambiar es conveniente no usar --remote o usar --float. Un detalle cuando se usa --remote y el puerto remoto está redirigido a un puerto UDP distinto al del router, es que entonces los paquetes tendrán un puerto origen distinto al que ve la otra parte como destino, salvo que enviara antes la otra parte un paquete y el NAT del router modifique el puerto origen.

Otra razón para no usar --remote en uno de los dos extremos, es si por ejemplo accedemos desde nuestra casa y unas veces usamos un PC y otras otro, por lo que no podemos utilizar siempre el mismo puerto. Pero ojo, no debemos conectarnos desde dos clientes distintos a la vez, salvo que usemos el modo servidor.

Usar certificados en lugar de claves compartidas no es complicado en OpenVPN, gracias a la mini autoridad de certificación de prueba que incluye (en Ubuntu está en el directorio de documentación, en el subdirectorio easy-rsa). Situados en este directorio, ejecutamos: **`source vars ; ./clean_all ; build-ca`**

Con esto ya tenemos creado el certificado y clave privada de la autoridad de certificación. Para generar el certificado y clave privada del servidor (en el supuesto que queramos modo servidor, de otro modo da igual generar todos los certificados clientes): **`./build-key-pkcs12 --server server`**

Para generar un certificado para un cliente: **`./build-key-pkcs12 cliente1`**

Generamos los parámetros DH, que deberemos pasar al lado que funciona como servidor TLS: **`./build-dh`**

En la máquina que va a funcionar como servidor TLS ejecutamos: **`openvpn --dev tun2 --remote 192.168.1.131 --ifconfig 10.5.0.2 10.5.0.1 --pkcs12 server.p12 --tls-server --dh dh1024.pem`**

En la máquina que va a funcionar como cliente TLS ejecutamos: **`openvpn --dev tun2 --remote 192.168.1.130 --ifconfig 10.5.0.1 10.5.0.2 --pkcs12 keys/cliente1.p12 --tls-client --ns-cert-type server`**

Para finalizar, un ejemplo de uso del modo servidor. Máquina servidor: **`openvpn --dev tun2 --server 10.5.0.0 255.255.0.0 --pkcs12 server.p12 --tls-server --dh dh1024.pem`**

Máquina cliente: **`openvpn --dev tun2 --remote 192.168.1.130 --client --pkcs12 keys/cliente1.p12 --tls-client --ns-cert-type server`**

El cliente obtendrá una IP en el rango indicado por el servidor; la IP del servidor será la x.x.x.1. Por defecto dos clientes que se conecten no podrán verse, salvo que se active una opción o el servidor añada reglas de enrutado. La opción ns-cert-type está para evitar que otro cliente con un certificado válido firmado por la CA se haga pasar por el servidor.

Con certificados, una opción recomendable es añadir la opción -tls-auth, con una clave compartida. Esta opción sirve para que el cliente tenga que proporcionar la clave compartida para poder continuar. Con esto se evitan ataques de denegación de servicio y posibles incidencias si un día se descubre un bug en openssl.

OpenVPN permite crear/usar/borrar un dispositivo TUN persistente, en lugar de crearlo al establecer la conexión y destruirlo al acabarla.

Capítulo 7

Escritorio remoto

7.1. Escritorio remoto: X-Window

El sistema X-Window desde sus orígenes se diseñó con vistas a la transparencia de red: una aplicación se visualiza y maneja desde un terminal, pero puede estar ejecutándose en cualquier máquina de la red con la que haya conexión TCP/IP. En la terminología X-Window, el PC del usuario ejecuta el servidor X, mientras que las aplicaciones son los clientes: la idea es que las aplicaciones hacen peticiones al servidor del tipo dibuja este mapa de bits o notificamé determinados eventos. El servidor notifica a los clientes eventos como pulsaciones de teclas o eventos del ratón.

Para ejecutar una aplicación desde una máquina remota a la que nos hemos conectado por ejemplo a través de un telnet o un ssh, lo primero que hace falta es que esté definida la variable DISPLAY, que indica el servidor a utilizar. Así, un valor :0 indica usar el servidor local (si hubiera más de uno, se incrementaría el número); con el valor 192.168.7.10:0, indicaríamos que conecte con el servidor X de la máquina 192.168.7.10.

El siguiente requisito, es que si la conexión es vía red, tiene que estar abierto en el cortafuegos el puerto TCP del servidor X. Este puerto por defecto es 6000 para el primer servidor (:0) y se va incrementando conforme se ejecutan más servidores sobre una misma máquina.

Finalmente, el servidor X tiene que permitir que el cliente se conecte. Hay dos formas:

1. método poco sutil, xhost. Ejecutando xhost desde nuestro servidor X damos permiso para que desde las Ips que indiquemos se puedan conectar a nuestro servidor. Así, con xhost +192.168.7.1 permitiríamos conectar desde esa IP. Con xhost + permitiríamos conectar desde cualquier IP. Ojo, no conceder este permiso con ligereza: un cliente puede espiar eventos de teclado, enviar eventos de teclado (nada divertido si tenemos una terminal abierta y envían un comando destructivo seguido de l evento de la tecla enter) o cuanto menos molestar bastante. A esto hay que añadir que al abrir una IP se abren a todos los posibles usuarios de la máquina si es multiusuario o al posible malware que tuviera instalado; así mismo no hay que olvidar la posibilidad del IP Spoofing.
2. método más adecuado, xauth. Para que el cliente pueda conectarse, salvo que se haya permitido explícitamente utilizando xhost como se describe en el método anterior, tiene que utilizar una "cookie" que está almacenada en la cuenta de usuario del servidor que ejecuta el servidor X-Window, en el fichero que indique la variable de entorno XAUTHORITY. Este es el motivo por el que en algunas distribuciones al cambiar de usuario, incluso al cambiar a root, no permite ejecutar aplicaciones X: la variable de entorno ya no apunta a este fichero, por lo que no se envía la cookie. Hay distribuciones en las que sí funciona al pasar a root, porque sudo, su o la orden que hayamos ejecutado para cambiar de usuario usa un módulo PAM para que al establecer una nueva sesión de usuario se copie la cookie. Como comentario adicional, con xhost es posible dar permiso para que abra conexiones cualquier cliente local con independencia de su usuario, pero no clientes remotos, mediante xhost +local:. Este mecanismo se apoya en que cuando cliente (aplicación) y servidor (terminal X) están en la misma máquina, no usan el socket TCP sino un socket Unix, que es sólo accesible localmente. De hecho si sólo vamos a permitir conexiones locales, es buena idea ejecutar el servidor X con la opción -nolisten tcp; es lo que hacen por defecto distribuciones como Ubuntu.

Mediante la herramienta xauth, podemos leer la cookie en el servidor X; luego también mediante la herramienta xauth, en la máquina remota, podemos añadir la cookie.

En realidad, el método recomendado para ejecutar aplicaciones remotas es conectarnos con ssh con la opción -X o con la opción -Y[1] (si recibiéramos algún -Y estamos sujetos a algunas restricciones de seguridad, a decir verdad no totalmente efectivas). Al ejecutar aplicaciones a través de SSH se usa para cada conexión un túnel encapsulado dentro de la conexión SSH, con todas las ventajas que implica: seguridad, al estar cifrado y protegido contra modificación y versatibilidad, pues no hay que abrir ningún puerto en el cortafuegos ni hay problema si nuestra máquina está tras una ADSL con NAT. La idea en que se basa es utilizar un proxy: crea un servidor X en la máquina remota y redirige todo lo que recibe por medio del túnel SSH al servidor X real que tenemos en nuestra máquina local. Gracias a este sistema funciona incluso aunque el servidor X se ejecute con la opción -nolisten tcp, pues la comunicación es local gracias al túnel.

Muchas veces la forma de ejecutar las aplicaciones remotas no es ejecutando aplicaciones sueltas, en nuestro escritorio local, sino ejecutando entero el escritorio remoto. Es decir, nos sale la pantalla de GDM o KDM que nos pide nuestro usuario y contraseña y a continuación entramos en un escritorio Gnome o KDE, pero que en realidad no es el de nuestra máquina, sino el de una máquina remota. Esta funcionalidad se apoya en el protocolo XDMCP, que por razones de seguridad suele estar desactivado (lo implementa el propio GDM/KDM). El protocolo usa el puerto UDP 177, pero sólo inicialmente; para mostrar la propia pantalla de Login ya recurre a la habitual conexión TCP con el puerto 6000, si bien automáticamente configura todo, incluyendo el fichero con la cookie, para que el usuario no tenga que hacer nada especial. Se puede establecer una conexión con un servidor XDMCP bien directamente desde el gdm de la máquina local, o pasando opciones como -query (ver man Xserver) a un servidor X ejecutado en otra consola, por ejemplo **`X :1 -query 192.168.10.7`** o bien a un servidor ejecutando en una ventana, como xnest o el más moderno Xephyr . En cualquier caso usar XDMCP es inseguro, habría que tener un túnel que proteja tanto la conexión UDP con el servidor XDMCP, como las conexiones TCP de los clientes X-Window. Una opción más sencilla de lograr una sesión sin perder seguridad, sobre todo teniendo en cuenta que openssh no permite túneles UDP, es olvidarse de XDMCP e invocar en la máquina remota vía SSH por ejemplo /etc/X11/Xsession[2] (teóricamente podría valer con ejecutar x-session-manager, pero suelen arrancarse aparte algunos demonios, como D-BUS).

Hay servidores X libres disponibles también para Windows: recomendamos http://x.cygwin.com/.

X-Window es la tecnología pionera de escritorio remoto, pero con el tiempo han surgido otras como RFB (VNC), RDP, Citrix, Adaptive Internet Protocol (Sun), Tarantella,

7.2. Escritorio remoto: RFB (VNC)

VNC es un sistema que permite conectarse a un escritorio remoto. El servidor se ejecuta en el ordenador remoto a cuyo escritorio nos queremos conectar y el cliente en nuestra máquina. Toda la pantalla del escritorio se maneja dentro de una ventana, no es posible como con X-Window ejecutar cada aplicación en su ventana, como si fueran aplicaciones locales. Así mismo es un sistema mucho menos avanzado que X-Window, basado simplemente en un framebuffer, retransmitiendo los cuadros que cambian (en cambio en X-Window se almacena información en el servidor, de modo que no hay que retransmitirla cada vez que se muestra por pantalla: como muestra un botón: el protocolo baso X-Window, sin extensiones, tiene 160 tipos de peticiones, respuestas y eventos).

Si VNC es menos avanzado que X-Window, ¿por qué hay gente que usa VNC en lugar de X-Window?

1. conectividad: VNC funciona sobre cualquier escritorio, incluyendo Windows: las aplicaciones Windows no son clientes X-Window, por lo que no podemos esperar usarlas remotamente utilizando este sistema.
2. conexión sin estado: cuando iniciamos una sesión remota con X-Window, no podemos en un momento dado decir, vale, me voy a casa, apago el monitor y me conecto luego desde casa y tengo todo como lo dejé, con el OpenOffice abierto editando un fichero y el visor de PDF por la página 85. El motivo es que los clientes están ligados a esa conexión con el servidor y en el servidor hay información de estado: si cerramos el servidor mueren las aplicaciones. En cambio con VNC realmente funciona la analogía monitor, teclado y ratón remoto.

[1] La opción -X ejecuta usando un modo restringido de seguridad, que impide por ejemplo acceder a determinadas propiedades y que puede impedir que algunas aplicaciones de escritorio se ejecuten. Las restricciones de seguridad que impone -X tampoco son demasiado efectivas por sí solas. Si al usar ssh con -X o con -Y, en la máquina remota no aparece definida la variable de entorno DISPLAY y por lo tanto no funcionan las aplicaciones X, posiblemente se deba a que en la máquina remota no está instalado xauth.

[2] Por ejemplo, desde una consola se puede lanzar **`startx /usr/bin/xterm -- :1`** y luego desde ese terminal (que no debe cerrarse o de lo contrario se cerrará la sesión) ejecutar **`ssh -Y user@host /etc/X11/Xsession`**. Con Xephyr la idea sería primero crear una cookie de autenticación invocando xauth; en la subshell hay que ejecutar una orden tipo **`add :1 . ad753146ead4c2e60031c60f22139da9f`** (:1 sería el DISPLAY) y para grabar salir con exit. A continuación se puede lanzar el servidor, por ejemplo con **`Xephyr :1 -auth .Xauthority &`**, finalmente se establece **`export DISPLAY=:1`** y se invoca ssh.

3. rendimiento aceptable en una conexión vía Internet. Este punto es desconcertante, porque por diseño X-Window es mucho más eficiente que VNC y debería ofrecer mejores resultados con menos ancho de banda. De hecho usando una red local con X-Window realmente las aplicaciones parecen locales. ¿Entonces que es lo que ocurre? El motivo es el round-trip. Conforme pasaban los años, los desarrolladores se despreocuparon más de la posibilidad de ejecutar las aplicaciones remotamente vía Internet, sobre todo en un punto: las aplicaciones e incluso los propios toolkits hacen muchas consultas al servidor X y esperan una respuesta, pese a que en general X-Window funciona asíncronamente. Estas peticiones-respuestas tardan en procesarse cuanto menos el tiempo de latencia de la red multiplicado por dos (uno para enviar, otro para recibir). En una aplicación que se ejecuta en local, la latencia es de 0,1 ms, por lo que aunque haya miles de round-trips el efecto es inapreciable. En una red local la latencia es de 1ms, todavía asumible. Pero si usamos por ejemplo una ADSL, ya encontraremos latencias de 50ms, en una línea serie 200ms y estos valores subirán aún más en conexiones GSM o por satélite. Afortunadamente hay una tecnología que trata de soluciones estos problemas, NX, de la que también hablaremos.
4. algunos programas como TightVNC (www.tightvnc.com/) soportan también transferencia de ficheros.
5. puede usarse para mostrar escritorio a otros usuarios o para solicitar asistencia; además de protegerse por contraseña se puede hacer que salga una ventana pidiendo autorización si de lo que se trata es de compartir escritorio.

VNC se ejecuta sobre una única conexión TCP, por lo que es muy fácil de securizar utilizando un túnel SSH, que es así mismo una solución interesante para permitir que los traajadores de una empresa puedan manejar en caso de necesidad su equipo de la oficina desde casa.

Tanto KDE como Gnome incluyen una opción de permitir administrar remotamente el equipo (en Gnome el paquete Vino, en KDE es krfb). Esta opción se implementa con VNC. También se implementa con VNC el "Apple Remote Desktop". Probablemente también se use en VMWare Server, que permite desde una consola VMWare acceder a máquinas virtuales ejecutándose en otra máquina.

A veces en lugar de VNC leeremos RFB. Es el nombre del protocolo: Remote Frame Buffer.

Relación de software que implementa VNC: http://en.wikipedia.org/wiki/Vnc. Como visores los más habituales son tighvnc y realvnc (este último de una empresa creada por los programadores originales, que ahora desarrollan una versión privativa además de la libre; las distribuciones suelen incluir este paquete como vnc a secas, sin el "real" en el nombre). Como servidores también estos, pero ya es más habitual usar el integrado en el escritorio, al permitir hacerlo funcionar en cualquier momento sobre un servidor X ya arrancado; en cambio en los otros el servidor VNC es también un servidor X-Window, de tal modo que primero hay que arrancar ese servidor VNC y desde otro servidor X ejecutar un cliente VNC para poder usar las aplicaciones. Este enfoque sigue siendo interesante cuando se quieren lanzar aplicaciones X-Window sobre una máquina remota que se sigan ejecutando cuando apagemos el ordenador desde el que nos estamos conectando. Así mismo hay programas interesantes como x2vnc, que permite (al estilo de x2x) manejar dos ordenadores de modo que al llegar el ratón al borde de la ventana "continúa" en la del otro PC.

La funcionalidad de ejecutar una aplicación interactiva y poder seguir manejándola desde otra máquina también existe para texto. Lo más habitual es lanzar la aplicación con screen. Si no lo hemos hecho, queda el recurso de linuxvnc, que se une a una sesión de terminal ya existente y permite manejarla vía VNC. Otra opción es conspy. Una alternativa a screen mucho más ligera pero que sólo aporta la funcionalidad de desvincular del terminal y no por ejemplo la de ejecutar varias aplicaciones es dtach

7.3. RDP. Escritorio remoto de Windows

Microsoft añadió soporte a su sistema operativo para poder ejecutar aplicaciones remotamente. En concreto lo implementó para Windows 2000 Server en el componente Windows Terminal Services, un sistema que permite al estilo Unix que varios usuarios estén conectados a la misma máquina. Evidemente para ello hay que pagar licencias, las de Windows 200x Server y la de Windows Terminal Services, esta última en función del número de clientes (una licencia por cliente que potencialemente se pueda conectar; fórmula menos ventajosa que licenciar por usuarios concurrentes y además propensa a abusos, pues la licencia cliente está incluida con algunas versiones de Windows, pero no con otros sistemas potencialmente clientes como GNU/Linux o Mac OSX). Como excepción, hay una licencia especial para usuarios que se conectan a través de Internet, que es la que se usa tienen en Hacienda para permitir que los usuarios de Linux y Windows hagan la declaración de la renta con el programa PADRE: la conexión es utilizando un cliente Cytrix (por Cytrix también se paga) pero a su vez Cytrix requiere Windows Terminal Services.

De todos modos por lo que es más conocido RDP es porque Windows XP Professional (no así Windows XP Home) incluye un servidor RDP limitado a una sola conexión y permitiendo una única sesión (es decir la máquina es monousario: o se usa en local

o en remoto, pero no se puede tener a la vez una sesión local y otra remota). Lo mismo ocurre con las versiones más caras de Windows Vista (lo incluyen las versiones Enterprise y Vista Ultimate, pero no ninguna de las ediciones de Windows Home); también se incluye en el antiguo Windows Media Center, Windows 2003 Server y Tablet Edition 2005. Es el famoso escritorio remoto, para el que existe un cliente libre también para GNU/Linux, rdesktop. con una interfaz gráfica muy intuitiva, tsclient. Teóricamente Microsoft podría cobrar también licencias por clientes

RDP ofrece mejor rendimiento que VNC. Sigue por detrás de X-Window, pero recordemos que X-Window tiene un problema práctico con el round-trip, que RDP. Hasta RDP 6, novedad en Windows Vista (también disponible al parecer para XP SP2) tenía la limitación de VNC: sólo permitía manejar el escritorio entero en una ventana, no una ventana por aplicación. El soporte para ejecutar aplicaciones cada una en su ventana (SeamlessRDP) lo permite excepcionalmente rdesktop con versiones más viejas de RDP utilizando un componente que se ejecuta en el servidor.

Otra ventaja común con VNC es que permite desconectar la sesión e iniciarla desde otra máquina dejando todo como estaba.

Añade alguna innovación como poder utilizar remotamente más dispositivos que el ratón, teclado y pantalla: por ejemplo la tarjeta de sonido o la impresora. Esto en Unix ha sido también siempre posible, a través del demonio de sonido y de impresión, la diferencia aquí es que todo se hace con un solo producto y en RDP 6 se han ido añadiendo más posibilidades.

RDP se ejecuta también sobre una conexión TCP, por lo que al igual que VNC es fácil de enrutar usando openssh o stunnel. En las últimas versiones permite cifrado usando TLS, mientras que el sistema de cifrado de las versiones viejas basado en usar RC4 no es seguro.

En Linux existe también un servidor RDP que va sobre VNC o directamente X-Window: http://xrdp.sourceforge.net/

La versión propietaria del virtualizador VirtualBox incluye un servidor RDP para manejar remotamente el escritorio que se ejecuta dentro del virtualizador.

7.4. NX. Nueva tecnología de escritorio remoto.

NX (http://en.wikipedia.org/wiki/NX_technology) es una tecnología de escritorio remoto desarrollada por una empresa Italiana, NoMachine. NoMachine comercializa su implementación propietaria de esta tecnología. Los clientes son freeware y hay también un servidor freeware, NX Free Edition, para GNU/Linux y Solaris, mientras que el servidor para Windows y MacOSX es de pago. NX Free Edition está limitado respecto a la versión de pago: sólo permite dos sesiones por servidor

Lo más interesante es que Nomachine además publica el código fuente necesario para implementar esta tecnlogía bajo licencia GPL (lamentablemente sólo bajo GPL2): http://www.nomachine.com/documents/technology/building-components-3.x.php. El lado negativo es que no hay ninguna implementación libre que venga con las grandes distribuciones actuales, por lo que esta tecnología a pesar de ser muy interesante, tiene una presencia marginal en GNU/Linux.

NX se implementa sobre X-Window para acelerarlo sensiblemente y eliminar round-trips. Así, gracias al uso de caché o el utilizar formatos tipo PNG en lugar de bitmaps se logran ratios de compresión del orden de 10:1 a 100:1. Gracias a estas compresiones es utilizable un escritorio remoto incluso a través de un modem.

En una serie de artículos sobre NX publicados en Linux Journal, se muestra como el inicio de sesión en KDE la primera vez supone transferencias de 4,1MB, pero gracias a NX y su caché las siguientes veces se queda en tan solo 35Kb.

NX aplica a X-Window mejoras que ofrece RDP, como el poder desconectar la sesión y conectarse desde otra máquina con todo como estaba (lo logra haciendo que el servidor X se ejecute en la máquina remota, mediante el agente nxagent, a la que se conecta el cliente NX) o el permitir transferencias de ficheros o utilizar por las aplicaciones remotas nuestra impresora o tarjeta de sonido local.

Desde la versión 3.0, NX permite ejecutar aplicaciones en modo rootless, es decir, como aplicaciones normales en lugar de todas dentro de una ventana que actua de escritorio.

En realidad NX también se puede utilizar sobre RDP o VNC para acelerar también estos protocolos, pero los resultados que se obtienen son inferiores respecto a X-Window: como hemos comentado la tecnología de X-Window es superior a RDP, sólo que había problemas como los round-trips que en la práctica lo hacían poco utilizable sobre redes no locales.

Una limitación de NX utlizado sobre X-Window frente a VNC, es que no es posible iniciar un servidor NX sobre un servidor X ya arrancado, si se quiere usar remotamente hay que arrancarlo vía NX desde el principio.

7.4.1. Cómo funciona NX

En la implementación gratuita aunque privativa de NoMachine, NX Free Edition for Linux, hay paquetes nxserver, nxnode y nxclient. El servidor necesita los tres, mientras que el cliente necesita nxnode y nxclient; la mayoría de los programas están dentro del paquete nxnode; el motivo por el que para un servidor hace falta nxclient es simplemente porque este paquete incluye también las librerías de NX. Uno de los programas más importantes que incluye el paquete nxnode es precisamente el ejecutable nxnode, que un proxy que se ejecuta tanto en el lado local como en el remoto y que crea el túnel entre los dos lados. En el lado de la aplicación remota, nxnode ejecuta un agente que es el intermediario con la aplicación: para aplicaciones X-Window es nxagent, para RDP es nxdesktop y para VNC nxviewer. Nxagent funciona en la máquina remota como un servidor X-Window local. En principio las aplicaciones podrían funcionar sin necesidad de nxagent, utilizando el túnel de nxnode al estilo de un túnel SSH, pero nxagent es lo que permite eliminar los round-trips

Un tema que desconcierta a un nuevo usuario de NX, es que el servidor nxserver no es un demonio que tenga que lanzar el usuario y permanezca en ejecución a la espera que se conecten clientes. En realidad nxserver es un programa presente en la máquina de las aplicaciones remotas, que ejecuta el cliente (nxclient) vía SSH, bajo la cuenta de usuario nx: la idea es similar al servidor de sftp que viene con OpenSSH. Otro dato desconcertante es que a priori cualquiera puede ejecutar nxserver, pues se ejecuta accediendo con una clave SSH que la clave privada es conocida por todo el mundo, pues viene con el cliente NX. Esto no debería ser un problema de seguridad, pues nxserver autentica al cliente antes de hacer nada (que es invocar nxnode), pero es mejor si cambiamos la clave por otra, aunque entonces habrá que cambiarla también en todos los clientes. Por defecto nxserver para autenticar al cliente usa su usuario y contraseña en el sistema, pero se puede mantener una lista de usuarios y contraseñas propia para no tener que usar la del sistema. Con nxserver --help podemos ver las ordenes disponibles para entre otras cosas administrar usuarios.

Si usamos el cliente oficial, es muy recomendable en configure/advanced la opción "enable SSL encryption of all the traffic". Esta opción tiene un nombre confuso, porque realmente no se usa SSL: lo que hace es utilizar la conexión SSH para entubar todas las conexiones entre el nxnode local del cliente y el del servidor; estas conexiones irán cifradas, pero no con SSL sino con el protocolo de SSH, que es similar pero no es SSL. El interés de esta opción no está sólo en que se cifre sino en pasar los cortafuegos: sin ella habría que abrir varios puertos, que además van cambiando conforme se abren nuevas conexiones.

Algunas ideas para probar por qué no funciona la conexión:

1. probar primero con el cliente oficial, con la opción "enable SSL encryption of all the traffic". El problema más frecuente es de cortafuegos.
2. probar a conectar al servidor con **`ssh -i /usr/NX/share/keys/server.id_dsa.key nx@ip_servidor`** ; ejecutar **`login`** ; si no autentifica probar a añadir el usuario manualmente. Si no ha sido posible conectar, comprobar si es problema de SSH; podemos por ejemplo conectarnos a la máquina y ejecutar **`su -- nx`**; si al entrar en la cuenta del usuario "nx" no se abre una shell en la que podemos ejecutar login, es que no está correctamente instalado.

La información proporcionada como se ha dicho es para el caso "NX Free Edition", que es software privativo. En el caso del software GPL liberado por NoMachine, no existe por ejemplo nxnode sino nxproxy (tiene la misma función, utilizar un túnel que se podrá utilizar directamente o mejor mediante nxagent).

7.4.2. NX y software libre

El proyecto FreeNX (http://freenx.berlios.de) permite instalar un servidor NX (i.e. el lado de las aplicaciones remotas) utilizando software libre. FreeNX lo que hace es compilar el código GPL que ofrece NoMachine y añadirle scripts para lograr una solución funcional, pues el software GPL de no machine implementa la tecnología pero no por ejemplo las herramientas para administrar las conexiones. FreeNX no forma parte de Ubuntu, ni siquiera se incluye en el repositorio Universe (sólo se incluye el componente nxproxy, para usarse sólo en el lado cliente), pero hay un repositorio externo que se puede utilizar: https://help.ubuntu.com/community/FreeNX . FreeNX tiene fallos de diseño que lo hacen difícil de mantener y por ello se está rediseñando. Es significativo que hace años Ubuntu incluyera FreeNX en su distribución y decidiera dejar de hacerlo, aunque el código actual ha mejorado.

FreeNX incluye el servidor, pero no un cliente. Como cliente es habitual utilizar el cliente freeware pero no libre de NoMachine; por ejemplo es el utilizado por la distribución para hacer thin clients ThinStation. La única alternativa libre "usable" es QtNX/Nxcl; NXCL es una librería, con una implementación de referencia en GTK+. El mismo autor de QtNX y NXCL está integrando la tecnología NX en más clientes y empaquetando versiones más recientes del código libre de NoMachine que el

proporcionado por el propio proyecto FreeNX; es muy recomendable buscar en su blog sobre NX en http://blog.gwright.org.uk/-articles/search?q=NX. Otra implementación completa de NX (incluido el cliente) bajo GPL, obtenida tras comprar una licencia a NX de todo el código fuente de la versión 1.5 y liberarlo bajo GPL: http://code.2x.com/linuxterminalserver/downloads. Sin embargo este código ha quedado anticuado, dado que NX ha ido incorporando novedades en sus versiones 2.x y 3.x.

Capítulo 8

Proxies, cortafuegos, software de filtrado

8.1. Proxies y software de filtrado

Este tipo de uso es de interés para los padres preocupados por el acceso de sus hijos a contenidos inadecuados en Internet, pero también para gente que le molesta que le salga pornografía y para los puestos de trabajo. Es típico el uso de listas negras (blacklisted) pero en algunos casos se usa lo contrario, listas blancas: el usuario sólo puede navegar por esas webs.

Para los puestos de trabajo, es importante que los sistemas permitan relajar las normas a determinadas horas: por ejemplo si un trabajador llega antes de la hora no debería haber ningún problema en que lea el periódico, o que a la hora de descanso o en determinados momentos del día se permitan hacer gestiones personales por Internet. También permiten reglas distintas según la IP de origen.

Una funcionalidad importante de este tipo de software no es sólo filtrar sino registrar los accesos. En cierto modo su efectividad está más en el efecto disuasorio de pensar que si se está navegando por dónde no se debe quizás el jefe se entere que por la habilidad del programa para filtrar.

Las implementaciones existentes de software de filtrado para GNU/Linux se implementan sobre el proxy Squid. SQUID se puede configurar para usarlo de modo transparente (y por lo tanto obligatorio) utilizando DNAT, por lo que no es posible saltarse el proxy salvo que se pueda usar un túnel y salir por un puerto que no esté redirigido al proxy transparente.

Un software basado en filtrado por listas negras es Squidguard (http://www.squidguard.org/). Hay varios sitios dónde es posible conseguir listas negras, pero hay que tener en cuenta que en muchos casos para obtener listas actualizadas hay que subscribirse y cuesta dinero. Una relación de listas gratuitas y de pago está en http://www.squidguard.org/blacklists.html; una de las gratuitas se actualiza regularmente, con un millón y medio de entradas. Una lista de pago es http://urlblacklist.com/; la primera vez deja descargar gratuitamente y realmente no hace demasiado control para asegurarse que no descargamos más veces sin subscribirnos, confiando en la honradez del usuario. Tiene unas dos millones de entradas.

El software de control parental más usado en GNU/Linux es DansGuardian. También va sobre Squid y utiliza listas negras, pero además de URLs soporta palabras claves, de tal modo que bloquea las webs en función de la puntuación que obtiene por las palabras presentes.

http://www.censornet.com/ Distribución de GNU/Linux que integra DansGuardian, administrable vía web. Comercializan subscripción a lista negra (como curiosidad el precio depende del país, más pagan los países desarrollados, los países subdesarrollados es gratis) y a un filtro de imágenes.

Puede no ser buena idea utlizar un proxy para todo. Sobre el papel usar SQUID para la web y SOCKS para otros protocolos obligaría a un uso responsable de la red, al estar autenticadas las conexiones. En la práctica la realidad puede ser más compleja y funcionar mejor soluciones más adaptativas, que simplemente tratan de ajustar el ancho de banda para que los usuarios que más consumen tengan menos prioridad. Pero si se usa un proxy SOCKS estas soluciones no funcionan, pues todos los paquetes proceden de la misma IP (si ya se ha pasado el proxy) o van a la misma IP (si es antes del proxy).

8.2. Netfilter/IPtables. Solución para crear cortafuegos, auditar red y hacer NAT

El filtrado en Linux lo hace un componente del kernel llamado netfilter. La utilidad para configurarlo es iptables, aunque es muy habitual no crear las reglas de un cortafuegos a mano sino utilizando una herramienta con interfaz gráfica.

Netfilter maneja distintas tablas: hay una tabla para filtrado (filter), es decir para el cortafuegos, que es la tabla que iptables entiende que queremos utilizar si no le indicamos otra cosa con la opción -t. Otra tabla que se usa mucho es nat, para hacer SNAT (sobre todo para salir a Internet con una sola IP todas las máquinas de una red local) o DNAT (para abrir puertos en un cortafuegos, para hacer proxies transparentes, para hacer balanceo de carga entre varias máquinas...). Hay más tablas que no veremos, como mangle, que permite alterar los paquetes.

Cada tabla contiene cadenas. Los paquetes pasarán por unas cadenas u otras. Así, en la tabla filter existen estas cadenas:

INPUT: por aquí pasan los paquetes que van destinados a la máquina

OUTPUT: por aquí pasan los paquetes que proceden de esta máquina, pero van destinados a otra

FORWARD: por aquí pasan los paquetes que ni proceden ni se dirigen a nuestra máquina, es decir, los paquetes que estamos enrutando.

En cuanto a la tabla nat, también tiene tres cadenas, aunque mientras que en filter cada paquete iba a una y sólo una de esas tres reglas básicas, aquí un paquete puede pasar por dos reglas:

PREROUTING: esta regla es para hacer DNAT de paquetes antes de enrutarlos, tan pronto como llegan a la máquina desde otra máquina.

OUTPUT: esta regla es para hacer DNAT de paquetes generados por la máquina con destino a otra máquina, antes de enrutarlos, para hacer DNAT

POSTROUTING: esta regla es para hacer SNAT de los paquetes justo después de determinar su ruta de salida.

Con iptables se añaden reglas a las cadenas (las reglas se pueden añadir al final, con -A o insertar al principio, con -I). Cada regla tiene unas condiciones de aplicación; por ejemplo una condición puede ser que la IP destino sea la 192.168.7.10 y el puerto destino sea el puerto TCP 8080. Así mismo cada regla tiene un target (objetivo) que es qué hacer con el paquete en caso de que se den las condiciones de aplicación de la regla: se establece con la opción -j, por ejemplo -j ACCEPT. Cuando se determina que a un paquete le es aplicable una cadena (por ejemplo llega un paquete con destino a un servidor de la máquina, luego le corresponde la regla INPUT) se va recorriendo la cadena secuencialmente hasta alcanzar una regla que tenga unas condiciones de aplicación que se ajusten al paquete. En ese caso se aplica el target de la regla.

El target puede ser un destino final: se ha decidido qué hacer con el paquete y ya no se miran más reglas. Por ejemplo ACCEPT para dejar pasar el paquete, REJECT para rechazarlo o DROP para no dejarlo pasar pero sin retornar un error ICMP, una práctica recomendable para no dar pistas a las aplicaciones que hacen escaneos de puertos, aunque para el caso TCP es más recomentable el target TARPIT (ver man iptables para ver cómo se usa). Otro target final es QUEUE, que permite pasar a una aplicación el paquete para que haga con él lo que quiera.

El target puede ser también un destino no final. Por ejemplo LOG registra el paquete; tras ejecutar esta target se siguen recorriendo las reglas.

El target puede ser otra cadena. El usuario pude crear sus propias cadenas, a las que añadirá sus reglas. En caso de poner como target una cadena, el resultado es que se salta a esa regla. Si se termina de recorrer la regla sin alcanzar ningún target final, se continuará en la regla siguiente al punto dónde se produjo el salto. También se puede retornar antes de llegar al final, con el target RETURN.

Las cadenas predefinidas (INPUT, OUTPUT, FORWARD... pero no las definidas por el usuario) pueden tener un target por defecto (que deberá ser un target final como ACCEPT o DROP, no un salto a otra cadena). El target por defecto es el que se aplica sobre los paquetes que han llegado al final de la cadena sin que les fuera aplicable ninguna regla con un target final.

Este target por defecto se establece con la opción -P. Esta opción habla de "policy" (política) pues permite establecer una política por defecto: rechazar todo salvo lo que explícitamente se permita (si el target por defecto es DROP) o aceptar todo salvo lo que explícitamente se prohíba (si el target por defecto es ACCEPT). Obviamente la política más segura es rechazar por defecto, pero es más difícil de llevar, sobre todo en una red grande: si hay poca comunicación entre los usuarios y los administradores y los usuarios ven cómo las políticas de seguridad de la empresa son un obstáculo para su trabajo buscarán formas de saltarse el cortafuegos, por ejemplo mediante el uso de túneles, por lo que a veces una política muy estricta si es también rígida puede ser contraproducente.

Ejemplos:

Permitir el tráfico saliente de la propia máquina y el enrutado, con destino al puerto 80 TCP

```
iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT
iptables -A FORWARDING -p tcp --dport 80 -j ACCEPT
```

La opción -p indica el protocolo (tcp, udp) y la opción --dport (sólo puede aparecer dport si aparece -t) el puerto destino; para indicar el puerto origen sería con sport).

No permitir conectarse a ningún servidor de la máquina, excepto al servidor SSH

```
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -p tcp --syn -j DROP
```

Otras opciones interesantes:

- -s: ip de origen
- -d: ip destino
- -i: interfaz de entrada
- -o: interfaz de salida
- -p icmp -icmp-type: para filtrar por tipo de mensaje ICMP. Ver con -p icmp -h la lista de tipos.

Recomendamos leer man iptables para conocer las impresionantes posibilidades de este programa. Por ejemplo hay una opción para limitar el número de conexiones a un puerto por IP de origen (o incluso por red de origen, mediante una máscara de bits). Otra opción permite poner como selector de reglas para paquetes generados localmente el usuario o el proceso que lo generó. Hay opciones para detectar escaneos de puertos, para ejecutar o dejar de ejecutar cuando la regla se ha dado una serie de veces. El módulo connbytes es perfecto para localizar conexiones que consumen mucho ancho de banda.

8.2.1. Cómo hacer SNAT

Supongamos que la IP pública de nuestro proveedor es 81.10.20.15; que la salida a Internet es por eth2 y que queremos que todas las conexiones salgan a Internet con esta IP. En este caso ejecutaríamos: **`iptables -t nat -A POSTROUTING -o eth2 -j SNAT --to-source 81.10.20.15`**

Si nuestra IP no es fija, en lugar de SNAT utilizamos MASQUERADE y no incluimos el parámetro --to-source.

Observesé que SNAT no se usa sólo para hacer NAT con la IP de la conexión a Internet. Otro caso típico es si tenemos una VPN: establecemos una red privada entre nuestra máquina y la de la empresa; para acceder al resto de máquinas de la empresa una posible solución es hacer en la máquina en la red de la empresa SNAT a su propia IP de los paquetes procedentes de la red virtual, de modo que los paquetes procedentes de nuestra casa parecerán que salen de la IP a la que nos hemos conectado en la red local.

8.2.2. Cómo hacer DNAT

La regla se añade a PREROUTING si el paquete procede de fuera, OUTPUT si el paquete se ha generado localmente.

Un uso típico de DNAT es abrir puertos en un cortafuegos y redirigirlos a Ips y puertos de la red local.

Otro uso típico es reemplazar un servidor por otro temporalmente, o incluso balancear entre varias Ips. En este último caso en lugar de DNAT usamos BALANCE y especificamos un rango de Ips.

El tercer uso corriente es hacer un proxy transparente. Ahora bien, mientras que hay protocolos en los que en la petición aparece de nuevo la IP o hostname más puerto (por ejemplo en HTTP aparece en la cabecera Host) esto no tiene por que ser así, de modo que el proxy no sabría nada sobre la IP y puerto destino. Por este motivo este caso de DNAT se hace con un target especial (REDIRECT) que sólo funciona sobre la propia máquina, porque así netfilter le pasa la IP y puerto destino a la aplicación a través del sistema operativo.

SQUID es un proxy que puede funcionar como proxy transparente. Es posible ejecutar SQUID en una máquina distinta que el enrutador, utilizando la tabla mangle de iptables para marcar los paquetes con destino a SQUID y utilizar la herramienta ip (del paquete iproute; este programa es necesario para realizar operaciones avanzadas que no se pueden hacer con otras herramientas como route, por ejemplo enrutar por IP de origen o por políticas. En este caso se usa una política que es que el paquete haya sido marcado desde netfilter, de modo que lo que hacemos es enrutar a la máquina de SQUID el tráfico que queremos que procese.

8.2.3. Cortafuegos a nivel capa de enlace de datos

Hay dos posibilidades para montar un cortafuegos sobre un bridge (lo que permite tener un cortafuegos transparente), en lugar de sobre un router, filtrando a nivel de información de la capa de enlace (además podemos seguir usando filtrado a nivel de capa IP).

1. Usar ebtables en lugar de iptables (en realidad se pueden usar los dos, cada uno con sus tablas). La fórmula más avanzada, permite por ejemplo considerar la VLAN (IEEE 802.1Q)
2. Usar en iptables las extensiones mac (para filtrar por dirección MAC) y physdev, para tener en cuenta en las reglas la interfaz de entrada o salida en el bridge.

Para más información: http://ebetables.sf.net/

8.2.4. Contadores con netfilter

Hay diversas herramientas para monitorizar redes y dar información sobre el ancho de banda consumido. Estas aplicaciones normalmente funcionan como sniffers, poniendo la tarjeta en modo promiscuo y recompilando información a través de libpcap. Una aplicación visual muy vistosa es Etherape. Una de las aplicaciones más recomendadas es iptraf (http://iptraf.seul.org/). Tiene interfaz de consola, con ncurses. Una relación extensa de herramientas para monitorizar el consumo de ancho de banda está en www.ubuntugeek.com/bandwidth-monitoring-tools-for-linux.html.

La otra posibilidad es utilizar el propio netfilter. Cara al rendimiento tenemos la ventaja que se ejecuta en el espacio del kernel.

Si ejecutamos iptables -L -v (se puede especificar una regla en concreto si no queremos ver todas) aparece junto con cada regla el número de paquetes que se han ajustado a esa regla y el total de bytes. Así, podemos escribir un script que cree una nueva cadena; en esa cadena añadimos una regla por cada IP de la red local, precisamente con su IP como selector (origen o destino, según lo que queramos monitorizar) y como target, RETURN. De este modo los contadores reflejarán los paquetes y bytes enviados/recibidos por/para cada una de esas Ips. Esta información es útil por ejemplo para detectar virus que tratan de hacer escaneos de puertos y que a veces provocan que la tabla NAT del router ADSL se sature. En este caso apreciaremos que desde esa IP se envían muchos paquetes, aunque probablemente con pocos bytes.

Para poner los contadores de una cadena a cero se usa la opción -Z.

Ejemplo de script para crear y borrar reglas para tener contadores:

```
#!/bin/sh
ipmasalta=198
start() {
iptables -N contadores
contador=1
while [ ${contador} -le $ipmasalta ]
do
iptables -A contadores --source 192.168.15.$contador -o eth2 -j ACCEPT
contador=`expr $contador + 1`
done
iptables -A FORWARD -j contadores
}

stop() {
iptables -D FORWARD -j contadores
contador=1
while [ ${contador} -le $ipmasalta ]
```

```
do
iptables -D contadores --source 192.168.15.$contador -o eth2 -j ACCEPT
contador=`expr $contador + 1`
done
iptables -X contadores
}
```

Existe un proyecto (http://ipac-ng.sourceforge.net/) que crea las reglas, recopila la información y la guarda, para finalmente visualizarla.

Otro sistema basado en netfilter es el módulo de webmin para monitorizar el ancho de banda.

Además con el módulo account es posible añadir una sola regla para registrar la información de toda una red y que desglose el resultado a nivel de tcp, udp, icmp y resto. Para más información ver www.svn.barbara.eu.org/ipt_account/wiki/Usage

Capítulo 9

Caso práctico: router red local, red otra empresa, Internet

Supongamos la red local de una empresa, dónde hay un enrutador que tiene que comunicar estas redes entre sí:

1. red local con los equipos de los usuarios y servidores de impresión. Están todos en la red 192.168.120.0/24
2. red local de otra empresa del parque tecnológico con la que colaboran un grupo de usuarios. Esta red es 10.2.0.0/16. La otra empresa ha pedido que los equipos de la red local no vean directamente su router, sino que haya un enlace punto a punto entre los servidores de las dos redes. Las Ips serán visibles en ambas redes, es decir, desde 192.168.120.* se verán las Ips de 10.2.*.* y viceversa.
3. acceso a Internet con router ADSL. Hay una única dirección IP, luego habrá que hacer NAT con configuración multipuesto, aunque también podríamos ponerlo en multipuesto y que el NAT lo haga la máquina Linux que actúa de router.

Para esta configuración necesitamos tres tarjetas de red para la máquina Linux que hará de router: una será para la red local (192.168.120.0/24), otra para conectarse con la red de la empresa y una tercera para el router ADSL para la conexión a Internet. ¿Nos las podíamos haber apañado con una sola tarjeta utilizando ipalias? pues realmente sí. El problema es que no es la configuración más adecuada, pues lo deseable es que desde la red local no accedan directamente a Internet ni a la red de la otra empresa y desde luego separar también Internet y la red de la otra empresa y poder poner cortafuegos en medio. Si se usa una única red física no hay tal separación y cualquier nodo podría tratar de espiar la red. En una infraestructura de red con switchs que soporten VLANs sí es viable usar una única tarjeta de red y sin necesidad de ipalias, sino creando tres VLANs distintas y configurando el puerto del switch al que está conectado el router como un Trunk port con acceso a las tres VLANs.

Tanto para conectarnos con la red de la otra empresa como con el router ADSL, utilizamos sendas redes con máscara de 30 bits, es decir de dos nodos. Por ejemplo con la red 192.168.200.84/30 nuestro nodo tendrá la IP 192.168.200.85 y el otro router la 192.168.200.86.

9.1. El problema de la IP de origen en máquinas multihome (con más de una IP).

Los encaminadores tienen más de una dirección IP, dado que tienen una IP en cada interfaz de red que encaminan. Ahora bien, cuando una máquina que tiene varias direcciones IP se comunica con otra máquina ¿qué IP origen figura en los paquetes? la IP de la interfaz de red por la que saldrá el paquete.

Esto puede ser problemático en algunos casos. Por ejemplo en nuestra configuración, en la interfaz eth1 que conecta nuestra red con la red de otra empresa. Resulta que los equipos de la red de la otra empresa no saben cómo llegar a 192.168.200.86: los equipos de ambas redes conocen las Ips 192.168.120.0 y 10.0.0.0, pero no conocen la red 192.168.200.84/30, que es un artificio para comunicar los dos enrutadores. Así pues, cualquier paquete que proceda de nuestro enrutador y vaya a una dirección de la red 10.0.0.0 llevará como IP origen 192.168.200.86, por lo que podrá llegar a destino pero luego no llegarán los paquetes respuesta. Podemos hacer una prueba muy sencilla: un simple ping a una máquina de la red 10.0.0.0 y veremos que la IP origen es 192.168.20.86 y no nos llega la respuesta, mientras que desde otra IP de la red sí.

La solución pasa por utilizar como IP origen 192.168.120.19 también cuando los paquetes vayan a la red 10.0.0.0 y no sólo cuando vayan a la red local. Ante esto caben dos enfoques:

1. Configurar cada programa que necesita contactar con la red 10 para que use la IP 192.168.120.19. Cada programa se configura de forma distinta. Por ejemplo con un ping basta indicar la IP con la opción -I, con SSH se indica mediante la opción BindAddress...
2. Hacer NAT: mediante iptables ponemos una regla para que todo paquete con la dirección origen 192.168.200.86 que vaya a salir por la interfaz eth1 cambie su IP a 192.168.120.19. Esta regla sería: **`iptables -t nat -A POSTROUTING --source 192.168.200.86 -o eth1 -j SNAT --to-source 192.168.120.19`**

¿Qué solución es más apropiada? Depende. La segunda obviamente es más directa, pues no hay que cambiar programa a programa. Ahora bien, por seguridad desde el enrutador no se debería acceder a más equipos de la otra red que los imprescindibles, por ejemplo los servidores DNS. Obligarse a reconfigurar cada programa que hay lo necesita es obligarse a mantener controlado a qué servicios de la otra red estamos accediendo desde el router.

Capítulo 10

Soluciones de virtualización/redes con virtualizadores

10.1. Virtualización

Mediante la virtualización en una sola máquina es posible tener varios sistemas, como si en lugar de un PC tuviéramos varios. Así, un ISP que ofrezca servidores dedicados puede ofrecer varios servidores virtuales sobre una máquina: cada uno funcionará como un servidor independiente con su propia IP, se podrán instalar todo tipo de servicios (DNS, mail, Jabber, web, PostgreSQL) y reiniciar y administrar independientemente.

Al sistema sobre el que se ejecuta el virtualizador se le llama host; a veces también se habla de domain 0, porque cada sistema que se ejecuta se le considera un dominio. A cada uno de los sistemas que se ejecutan sobre el virtualizador se les denomina guest. En algunos casos no hay sistema host porque el virtualizador se ejecuta directamente sobre el hardware.

Hay hardware como los mainframes de IBM, que soportan virtualización. El diseño original del PC no soporta virtualización: hay instrucciones de modo protegido que impiden ejecutar dos sistemas operativos simultáneamente, pero mediante software se pueden suplir (con bastante complejidad) las carencias de hardware. Así mismo mediante sofware se emulan componentes del hardware como discos, tarjeta gráfica, de red y de sonido.

Además de la virtualización de la CPU, está la del resto del hardware: tarjeta de red, discos, tarjeta gráfica... Para esta parte la mayoría de los proyectos libres toman código de Qemu.

Hay varios métodos de implementar la virtualiación del procesador:

1. Emuladores totales: emulan totalmente el hardware, incluyendo el procesador. Es el caso de Booch, o el de Qemu cuando emula una arquitectura distinta (por ejemplo ARM). Su rendimiento es muy pobre.
2. Emuladores con compilación JIT: es el caso de Qemu, cuando se ejecuta sin el módulo kqemu. El código necesita "compilarse" para la máquina virtual, si bien como los compiladores JIT de Java se hace sólo la primera vez que se ejecuta el código, luego ya está compilado. Mucho más rápidos que los emuladores totales, pero más lentos que el resto de soluciones. Se puede ejecutar como usuario normal sin instalar nada como root. Se pueden emular procesadores distintos.
3. Virtualizadores completos: es el caso de Vmware (software privativo), Qemu con el módulo de aceleración Kqemu y Virtual Box (software con una versión libre y otra más funcional privativa). Se basan en un VMM (Virtual Machine Monitor, también conocido como hypervisor) que mediante complicadas técnicas (como traps y análisis del código antes de su ejecución) detecta en tiempo de ejecución el código que no puede ejecutarse directamente porque afectaría a todo el sistema, modificando dinámicamente las instrucciones conflictivas. El rendimiento varía mucho según lo avanzado que sea el VMM: Vmware tiene varias patentes. Estos tres programas permiten instalar un sistema operativo sobre un virtualizador del mismo modo que sobre un PC: la ventana del virtualizador asemeja el monitor de un PC, podemos entrar en la BIOS, reiniciar el ordenador...
4. Paravirtualizadores: es el caso de Xen y UML (User mode Linux). En lugar de tener que detectar un VMM los casos conflictivos en tiempo de ejecución, se modifica el código fuente de los sistemas operativos para evitar esos casos conflictivos.

En lugar de el VMM tener que analizar el código, es el código quien invoca al VMM cuando sea necesario. Esta técnica simplifica muchísimo el VMM y ofrece muy buen rendimiento, aunque en el caso concreto de UML el rendimiento es mediocre. La pega es que haya que parchear el sistema operativo, sobre todo para poder ejecutar Windows. Xen parcheó un Windows XP en un programa de investigación que permitía acceso al código fuente de Microsoft, pero ese tipo de licencias no permitía distribuir el resultado. UML también se puede considerar dentro de esta categoría, pues es una modificación del kernel de Linux para que pueda ejecutarse dentro de otro Linux. Xen no emula una tarjeta gráfica SVGA "completa" como Qemu, Vmware o VirtualBox, pero utiliza VNC que para el sistema guest se ve como una tarjeta VGA.

Vmware y VirtualBox no son paravirtualizadores, pero utilizan esta técnica para virtualizar la E/S en los drivers especiales que se ejecutan en el sistema guest (por ejemplo el driver de red y el de la tarjeta gráfica: se comunican con el hipervisor en lugar de ser drivers normales sobre el hardware emulado). Lo mismo planea hacer KVM. En el caso de Vmware estas herramientas y drivers que se instalan sobre el sistema guest son las vmware tools; las versiones para Unix de estas herramientas, no así las de Windows, las ha liberado Vmware en un alto porcentaje bajo licencias libres, en el proyecto Open Virtual Machine Tools

5. Virtualizadores apoyados en el hardware (un tanto pretenciosamente llamados nativos): tanto Intel con sus extensiones VT como AMD con AMD-V (también conocido como Pacífica) ofrecen extensiones de virtualización desde hace tiempo. Hay que decir que mientras que AMD soporta virtualización en casi todos sus procesadores actuales, Intel juega más a la segmentación del mercado y hay modelos modernos que no lo incluyen, como muchos modelos de Atom e incluso algunos de Core2 Duo; ver la página sobre virtualization en la Wikipedia para obtener una relación completa. Gracias a estas instrucciones ya no es necesario un complicado VMM ni parchear el sistema operativo, si bien sigue siendo necesario virtualizar otros dispositivos y emular discos, tarjetas gráficas, de red... Ejemplos de estas soluciones son KVM (integrado en el kernel desde la versión 2.6.20, utiliza qemu para la virtualización del resto del hardware) y Virtual Iron (basado en Xen, pero ya no es paravirtualizador; en cualquier caso es una solución propietaria, con algo de código bajo GPL). Además Xen soporta también estas instrucciones, como método para poder ejecutar Windows o simplemente sistemas sin paravirtualizar. Los virtualizadores apoyados en el hardware son más lentos que los paravirtualizadores e incluso pueden ser menos eficientes que los virtualizadores completos con un VMM avanzado. Algunos virtualizadores de hecho soportan estas extensiones pero no las usan salvo para determinadas configuraciones, si bien por ejemplo VirtualBox inicialmente no las usaba por defecto y ahora sí.

6. Virtualización a nivel de sistema operativo (OS level Virtualization): no permite ejecutar dos sistemas operativos simultáneamente, sino servidores privados virtuales (SVP) dentro de un único servidor, es decir, es un único kernel, pero que permite aislar (isolate) los servidores. Cada servidor tendrá su propia red, espacio de disco, de memoria, se podrá reiniciar... así mismo tendrá limitación de uso de CPU con el fin de evitar que un servidor virtual esquilme recursos de los otros. También esta tecnología se denomina como Jail, pues es extender el concepto de chroot. Dentro del kernel de Linux ya existen namespaces de red (Ver http://lxc.sourceforge.net/network.php), que permiten que una interfaz sea privada a un proceso y sus hijos, pero las soluciones que están disponibles como parches van más allá. Tanto Linux VServer como Virtuozzo lo vienen ofreciendo proveedores de hosting desde hace años. Estas son las dos soluciones libres más destacadas:

   a) Linux VServer: http://linux-vserver.org (no confundir con linux virtual server, que es sobre clusters). Software libre. Una limitación es que no admite usar iptables dentro de cada SVP, sino dentro del host. Tampoco admite migración de procesos. Así mismo está más limitado en lo que se virtualiza, por ejemplo no se virtualiza nada de /proc). Por otro lado la distribución host parece menos restringida que en OpenVZ, que como host sólo admite Fedora, algunas versiones de CentOS y algunas de RHEL. Hay varias distribuciones que funcionan como guest directamente (las instalamos en su partición y luego las usamos, con sus programas y librerías, pero no obviamente con su kernel) aunque otras como Gentoo no.

   b) OpenVZ (http://openvz.org). Virtuozzo es un virtualizador bajo una licencia privativa; OpenVZ es el producto de la misma compañía que es software libre y que es la base de Virtuozzo. Un hecho positivo es que OpenVZ no es un conjunto de código GPL difícil de integrar y sin soporte como ocurre con otros productos en los que hay versión GPL y privativa: también venden soporte para OpenVZ. Admite distintas distribuciones como host, a través de templates, que son repositorios para bajar los paquetes necesarios para esa distribución. Hosting con OpenVZ: http://www.vpslink.com/vps-hosting/. En algún caso es más caro con OpenVZ que con Virtuozzo, por ser menos maduro y requerir más recursos...

Entre las posibilidades más avanzadas de algunos virtualizadores está el migrar en caliente con una indisponibilidad inapreciable un dominio de un servidor a otro. Dentro de los productos libres lo permiten Xen, KVM y OpenVZ. Vmware lo permite a través de productos de pago. Para saber más sobre este tema: http://www.linux-kvm.org/page/Migration

Un debate interesante es el de los virtualizadores que se ejecutan "bare metal", es decir, directamente sobre el hardware, en lugar de sobre un sistema operativo "host". Vmware ESX y ESXi (el segundo es gratuito con ciertas restricciones, ninguno de los dos libres; Vmware vSphere funciona sobre ESX o ESXi) sigue esta fórmula de forma pura, de tal modo que sólo funciona con determinado hardware. En el caso de Xen, se ejecuta sobre un host, pero asume parte de las funciones de un sistema operativo, al tener código por ejemplo para planificar la CPU. En cambio KVM utiliza todo lo que aporta Linux, lo que para muchos desarrolladores es la opción preferible pues difícilmente una empresa que desarrolla un producto de virtualización tendrá más experiencia en elementos de sistemas operativos como un planificador, que el grupo de personas que llevan desarrollando estos componentes desde hace años en un sistema tan extendido como el kernel Linux.

Software interesante:

http://virt-manager.org/: Interfaz gráfica para administrar máquinas virtuales con Xen, KVM y Qemu. Incluye parte para gestionar redes.

Información muy completa (navegar por los enlaces del marco de la izquierda): http://virt.kernelnewbies.org/TechOverview

10.2. Virtualización para aprender sobre redes

Para aprender temas de redes, es buena idea usar un simulador implementado sobre un virtualizador. La plataforma más adecuada es User Mode Linux, que como virtualizador de producción es la solución más pobre, pero como herramienta educativa es muy interesante. Destacaremos estas tres implementaciones:

1. Netkit: http://www.netkit.org/
2. VNUML (Virtual Network UML): http://www.dit.upm.es/vnumlwiki/index.php/Main_Page

   Es un proyecto activo español, de una universidad participado por Telefónica I+D. Lo más interesante es que se crea la configuración en XML y un intérprete se encarga de construir toda la configuración de UML. También se puede utilizar desde Windows mediante CoLinux.
3. VDE (Virtual Distributed Ethernet): http://wiki.virtualsquare.org/index.php/VDE_Basic_Networking (simula uno o varios switch). Realmente es un programa (vde_switch) que hace de switch al que se le añaden máquinas virtuales, ya sean de UML, Qemu o KVM. Así mismo se pueden encadenar switch, que pueden estar en distintas máquinas. Forma parte de un proyecto más amplio, Virtual Square. Más orientado a temas de virtualización y menos a educativos e investigación en redes que los anteriores, que son sólo sobre un PC con UML.

10.3. VMWare y la red

En VMWare a la hora de crear un dispositivo de red se puede elegir entre:

1. NAT: en el host aparecerá como la interfaz vmnet8, pero no podremos hacer nada con esta interfaz, por ejemplo usar Iptables para restringir las Ips que se pueden alcanzar desde el sistema que se ejecuta dentro de Vmware. El motivo es que no se usa realmente el NAT del núcleo sino un demonio, vmnet-natd, por lo que los paquetes no pasan realmente por vmnet8. Si se quiere abrir puertos, basta con editar el fichero /etc/vmware/vmnet8/nat.conf
2. Bridge: también se implementa utilizando un demonio privativo, en este caso vmnet-bridge, en lugar de utilizar el soporte del núcleo, por lo que no se puede restringir la red con iptables sobre la interfaz vmnet2. Como en el caso de vmnet8, realmente el tráfico no pasa por esta interfaz.
3. Host only: aparentemente este modo es el más limitado. Pues bien, en realidad es el más flexible, pues por la interfaz de red que crea, vmnet1, sí que pasan todos los paquetes. De este modo podemos utilizar esta interfaz para hacer NAT a través de iptables, o crear un bridge con brctl. Al usar iptables, podemos restringir el tráfico como con cualquier otra unidad de red.

Las interfaces de red vmnet1, vmnet2, vment8, se crean al ejecutar vmware-config, en la parte de configuración de red. Una posibilidad interesante si vamos a ejecutar varios sistemas simultáneamente o si queremos que un sistema tenga más de una interfaz es crear más de un dispositivo de red para "host only", de modo que aparte de vmnet1 haya otros. De otro modo todas las

máquinas virtuales estarán conectadas a la misma red, la de vmnet1. Así, una máquina podrá tener la IP 192.168.152.128, otra la 192.168.152.129 y el host la 192.168.152.1; las dos máquinas virtuales se verán la una a la otra y podrán comunicarse, aunque eso sí, con un sniffer una máquina no verá el tráfico de los otros nodos.

La red vmnet1 que crea Vmware es de máscara de red 255.255.255.0; ejecuta un servidor DHCP automáticamente, cuya configuración y arrendamientos pueden verse en /etc/vmware/vmnet1. Pero nada impide que podamos cambiar esta configuración, pues se emula a una interfaz ethernet. Por ejemplo un nodo puede cambiar su IP a otra de la red o incluso crear un alias y usar una red distinta, tanto en el host como en las máquinas virtuales.

Si usamos vmnet1 para hacer NAT utilizando iptables, hay que tener en cuenta que habrá que configurar la red de la máquina virtual para añadirle una ruta por defecto y la configuración del DNS y que el firewall deberá permitir que llegue al servidor DNS. Para pasar la configuración de DNS y ruta por defecto podemos usar el servidor DHCP de Vmware: "option routers" y "option domain-name-servers".

10.4. Qemu/UML y TUN/TAP

En Qemu hay dos formas de utilizar la red. Por defecto se usa -net socket, que sería el equivalente al modo NAT de Vmware. Mediante la opción -redir se pueden abrir puertos de servidor. Una diferencia interesante sobre Vmware es que esta solución se implementa enteramente en espacio de usuario, por lo que no se crea interfaz de red ni se precisa cargar ningún módulo del kernel, lo que es bueno porque en el caso de Vmware es un módulo privativo que activa la marca "tained" del kérnel, con lo que perdemos toda opción de soporte. Además así no hay que tener privilegios de superusuario para instalar Qemu (Vmware no requiere privilegios para ejecutarse, pero sí hace falta para insertar el módulo en el kernel).

Un inconveniente de -net socket es que como ocurre con los modos NAT y Bridge de Vmware los paquetes no pasan por ninguna interfaz de red, por lo que no se puede utilizar iptables para restringir la red.

La solución está en el uso del soporte de TUN/TAP del kernel. Consiste en que una aplicación puede abrir el fichero de dispositivo /dev/net/tun y con eso crear una nueva interfaz de red (por defecto tun0). Todo lo que la aplicación escriba en ese dispositivo se recibirá en la interfaz de red recién creada; de igual modo todo lo que llegue a esa interfaz de red (por ejemplo a través de enrutamiento, un ping, un servidor que escucha en esa IP y recibe paquetes de otro programa) lo leerá la aplicación del fichero de dispositivo.

Los dispositivos TUN operan a nivel IP[1] y son punto a punto. Los dispositivos TAP operan a nivel 2 y son multipunto, como las interfaces eth*. Un dispositivo tap0 y un dispositivo vmnet1 vienen a funcionar de forma muy similar y a nivel de ejemplos de configuración con iptables o brctl donde aparezca un tap0 podría aparecer un vmnet1 y viceversa. Por lo general se usa tun0 en lugar de tap0; para la mayoría de los usos son equivalentes por lo que es mucho más habitual utilizar tun0 que resulta más sencillo y directo.

Normalmente un dispositivo tun/tap sólo existe mientras el programa no cierra el fichero /dev/net/tun. Esto a veces es problemático, especialmente porque para crear un dispositivo TUN/TAP hacen falta privilegios de superusuario. Afortunadamente, con root se puede abrir un dispositivo en modo persistente para un determinado usuario, de modo que luego un programa ejecutado por ese usario sin privilegios podrá abrir ese dispositivo tun/tap creado para él por el root. Este se puede hacer con el programa tunctl, que forma parte del paquete uml-utilities.

¿Por qué no usa Vmware TUN/TAP en lugar de vmnet1? quizás por unicidad entre plataformas, o por diferencia de implementación; es posible que vmnet1 también permita driver de red de la máquina virtual directamente en espacio del kernel.

TUN/TAP es la solución utilizada también por otros virtualizadores, como UML. Muy interesantes los documentos sobre red avanzada en la web de Virtual Box, aunque desde la versión 3.0 ya no es necesario utilizar TUN/TAP: http://www.virtualbox.org/-wiki/User_HOWTOS.

10.4.1. Ejemplos

Permitir acceso sólo a red local 192.168.10.0 y además excluir el nodo 9 de esa red:

[1] En realidad no tiene por qué ser tráfico IP, puede ser también IPX o cualquier otro protocolo de red con tal que sea todo del mismo. Es decir, es a nivel del "payload" (contenido) del paquete Ethernet, mientras que TAP es a nivel del paquete Ethernet entero.

```
iptables -A FORWARD -i vmnet1 --destination 192.168.10.9 -j REJECT
iptables -A FORWARD -i vmnet1 --destination 192.168.10.0/24 -j ACCEPT
iptables -A FORWARD -i vmnet1 -j REJECT
iptables -t nat -A POSTROUTING -j MASQUERADE -o eth0
echo 1 > /proc/sys/net/ipv4/conf/eth0/forwarding
echo 1 > /proc/sys/net/ipv4/conf/vmnet1/forwarding
```

Crear un bridge, pero prohibiendo el acceso a la IP 157.88.10.20. Se puede filtrar con ebtables o con iptables (ojo, que hay tarjetas wireless y AP que no soportan bridges:

```
ifconfig eth0 0.0.0.0
ifconfig vmnet1 0.0.0.0
brctl addbr puente
brctl addbr puente
brctl addif eth0
brctl addif vmnet1
ifconfig puente 192.168.10.1
iptables -O FORWARD -m physdev --physdev-in vmnet1 --destination 157.88.10.20 -j REJECT
```

10.4.2. Instalar QEMU

Para compilar kqemu no hace falta versión vieja de compilador gcc, ni las fuentes de qemu. Es bastante rápido: **`./configure && make && && make install && modprobe kqemu`**

Por defecto, necesitamos permisos de root para leer /dev/kqemu

1. creamos grupo qemu: **`sudo addgroup qemu`**
2. añadimos nuestro usuario (jomar) al grupo: **`sudo gpasswd -a jomar qemu`**
3. configuramos udev para que cree fichero de dispositivo con permisos para grupo qemu. Para ello editamos fichero /etc/udev/rules.d/ kqemu.rules con este contenido:

   ```
   KERNEL=="kqemu", NAME="%k", MODE="0666", GROUP="qemu", RUN="/root/prueba.sh"
   ```

4. hacemos lo propio con el fichero /dev/net/tun. A partir del kernel 2.6.18 no pasa nada por dar permiso para todo el mundo, pues nadie sin privilegios puede crear una nueva interfaz si no se ha creado antes por el root para ese usuario. Esto se puede hacer con la herramienta tunctl, que forma parte del paquete uml-utilities. También se puede usar el programa que adjuntamos más adelante, cambiando el tipo de dispositivo de tun a tap. Para crear el dispositivo con tunctl se usa **`tunctl -u jomar -t tap0`**; para borrarlo **`tunctl -d tap0`**.

Ejemplos:

```
./qemu -net nic -net tap,script=no ~/linux-0.2.img -net nic,vlan=1 -net socket,vlan=1, ↩
    listen=:8081 centos.qcow

ifconfig tap0 up
ifconfig eth0
brctl addif puente tap0
ifconfig eth0 0.0.0.0
brctl addif puente eth0
ifconfig eth0 192.168.15.45
./qemu -net nic -net tap,script=no ~/centos.qcow2 -no-kqemu
```

10.4.3. Crear una imagen con Qemu

qemu-img create -f qcow2 centos.qcow2 3G

Con esta orden se crea una imagen de un disco de 3GB; en realidad con GNU/Linux este fichero no ocupa 3GB; el espacio sin usar del fichero no ocupa espacio en el disco.

Para arrancar del CD de instalación podemos ejecutar: **qemu -kernel-kqemu -cdrom /home/jomar/centos1of6.iso -boot d -m 512 centos.qcow2**

La opción -kernel-kqemu es para obtener la máxima aceleración: acelera también el código del espacio del kernel; por defecto sólo se acelera la de usuario. Si se produce algún problema, reintentaremos sin esta opción; si sigue habiendo problemas podemos probar con -no-kqemu a quitar incluso la aceleración de espacio de usuario. Tras la instalación, podemos probar a volver a utilizar aceleración: a veces los posibles problemas sólo se dan durante la instalación, aunque esta situación es más propia de Windows que GNU/Linux.

Con la opción -cdrom le indicamos que el CDROM es en realidad esa ISO; así evitamos el tener que tostar un CD. Con la opción -boot d indicamos que arranque de CD en vez de disco duro. La opción -m 512 establece que la máquina virtual tenga 512 MB de memoria. La cantidad de memoria puede cambiarse de una ejecución a otra, pero hay que tener en cuenta que los instaladores suelen crear una partición de intercambio con el doble de la memoria RAM.

¿Cómo cambiar de CD durante la instalación? Con ctrl-alt-2 (ojo 1, no F1) pasamos al monitor, en el que ejecutamos el comando: **change cdrom /home/jomar/centos2of6.iso**

Tras escribir la orden (qemu no indica ni errores ni que la operación se ha realizado con éxito) volvemos a la pantalla de la máquina virtual con ctrl-alt-1.

Una característica interesante de Qemu es que permite crear a partir de un fichero de imagen, otra imagen que sólo contendrá los cambios que se produzcan en la primera, si la primera permanece sin modificar. Esta característica es útil por ejemplo para crear dos imágenes muy parecidas para hacer pruebas de redes entre las dos imágenes, sin que ocupe tanto en el disco. **qemu-img create -b xubuntu.qcow2 xubuntu1.qcow2 & qemu-img create -b xubuntu.qcow2 xubuntu2.qcow2**

10.4.4. Listados de ejemplos de TUN/TAP

Listado para usar el dispositivo creado para este usuario (en kernels viejos lo crea si no existe): muestra primer paquete que recibe, una salida similar a la de tcpdump -x -i tun0. Para probarlo configuramos con ifconfig el dispositivo tun0, usando pointtopoint para indicar una IP supuestamente destino; podemos hacer un ping a esa dirección y ver lo que recibe el programa.

```
/*
 Este programa abre un dispositivo TUN (intenta que sea tun0), lee paquetes de
 él y los muestra por pantalla en formato hexadecimal. Tras ejecutar el
 programa, en otro terminal configuramos la interfaz tun0, por ejemplo:
 ifconfig tun0 192.168.13.1 pointopoint 192.168.13.2

 Podemos utilizar la IP 192.168.13.1 para poner un servidor en ella y enviar paquetes:
 este programa no recibirá ningún paquete hasta que sea un paquete que pretenda ir por
 el enlace punto a punto, por ejemplo si intentamos hacer un ping a 192.168.13.2

 Con TAP es similar, con la diferencia que no es punto a punto. El programa no verá
 paquetes que vayan para la propia IP, pero sí los destinados a otras. Esto es
 comportamiento del kernel, que no envía al dispositivo (por eso tampoco lo veremos
 con un sniffer) los paquetes que pueda procesar directamente; nos pasa lo mismo
 con dispositivos "reales" como eth0.


Información sobre TUN/TAP en:
 http://www.mjmwired.net/kernel/Documentation/networking/tuntap.txt
Y también en el fichero de cabecera:
 /usr/include/linux/if_tun.h

Este programa requiere que exista /dev/net/tun y que o bien seamos root, o si no lo
```

```
somos que además de tener permiso de lectura/escritura sobre ese fichero el dispositivo
ya ha sido creado persistente por el root (o por alguien con privilegio CAP_NET_ADMIN)
y nos haya asignado a nuestro UID como dueños; esto se puede hacer por ejemplo con tunctl
(de User Mode Linux) o con el otro programa de ejemplo que acompaña a este.

En versiones anterioes al kernel 2.6.18 bastaba con tener derecho de lectura/escritura en
/dev/net/tun, el sistema actual es más adecuado, permitiendo dejar /dev/net/tun con
permiso para todo el mundo sin problemas de seguridad.

(c) 2007 Chema Peribáñez
    Pongo este fichero bajo dominio público.
*/
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <linux/if.h>
#include <linux/if_tun.h>
#include <string.h>

int main() {
   struct ifreq ifr;
   int fd;
   int err, leidos, i;
   char *nombre="tun0";
   char dev[IFNAMSIZ];
   char buffer[1024];

   fd = open("/dev/net/tun", O_RDWR) ;
   memset(&ifr, 0, sizeof(ifr));

   /* Flags:
    * IFF_TUN   - Crear dispositivo TUN
    * IFF_TAP   - Crear dispositivo TAP
    * IFF_NO_PI - No incluir información del paquete (TUN): son 4 bytes, los dos
                  primeros flags y los dos siguientes el protocolo del paquete
                 (si es IPv4, IPv6, etc. este código es el número de protocolo
                  en un paquete ethernet, por ejemplo 0x0800 es un paquete IP.
                  Una lista de constantes podemos encontrarlas en
                 linux/if_ether.h
    */
    ifr.ifr_flags = IFF_TUN|IFF_NO_PI;
    strcpy(ifr.ifr_name, nombre);
    err= ioctl(fd, TUNSETIFF, (void *) &ifr);
    if( err<0) {
  perror("falló");
        close(fd);
        return err;
    }
    strcpy(dev, ifr.ifr_name);
    printf("%s\n",ifr.ifr_name);
    while (1) {
      leidos=read(fd,buffer,1024);
      printf("Longitud datagrama: %d\n",leidos);
      for (i=0;i<leidos;++i) {
    printf("[%02hhx]",buffer[i]);
      }
      printf("\n");
```

```
   }
}
```

Listado para crear como persistente el nodo:

```
/*
 Este programa crea un dispositivo TUN  persistente (intenta que sea tun0) y lo asigna
 al usuario con uid=1000. Este programa hay que ejecutarlo con privilegios de root (o al
 menos con privilegios CAP_NET_ADMIN)

 Información sobre TUN/TAP en:
 http://www.mjmwired.net/kernel/Documentation/networking/tuntap.txt
Y también en el fichero de cabecera:
 /usr/include/linux/if_tun.h

(c) 2007 Chema Peribáñez
    Pongo este fichero bajo dominio público.
*/
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <linux/if.h>
#include <linux/if_tun.h>
#include <string.h>
int main() {
   struct ifreq ifr;
   int fd, err;
   char dev[IFNAMSIZ];
   char *nombre="tun0";

   fd = open("/dev/net/tun", O_RDWR) ;
   memset(&ifr, 0, sizeof(ifr));

   /* Flags:
    * IFF_TUN   - Crear dispositivo TUN
    * IFF_TAP   - Crear dispositivo TAP
    * IFF_NO_PI - No incluir información del paquete (TUN): son 4 bytes, los dos
                  primeros flags y los dos siguientes el protocolo del paquete
                 (si es IPv4, IPv6, etc. este código es el número de protocolo en
                 un paquete ethernet, por ejemplo 0x0800 es un paquete IP. Una
                 lista de constantes podemos encontrarlas en linux/if_ether.h
  */
    ifr.ifr_flags = IFF_TUN|IFF_NO_PI;
    strcpy(ifr.ifr_name, nombre);
    err= ioctl(fd, TUNSETIFF, (void *) &ifr);
    if (!(err<0)) {
      // Hacer dispositivo persistente, es decir, que permanece
      // creado aunque se cierre dispositivo; normalmente el
      // dispositivo sólo existe mientras está abierto el
      // descriptor de fichero a /dev/net/tun.
      // Cambiar 1 por 0 para borrar un dispositivo persistente.
      err= ioctl(fd, TUNSETPERSIST, 1);

      // Poner como dueño del dispositivo al UID 1000; de este
      // modo puede crear dispositivo proceso ejecutado por
```

```
      // usuario con este UID en lugar de root, si además tiene
      // permiso de lectura y escritura sobre /dev/net/tun
      // antes de kernel 2.6.18 no era necesario hacer esto, pero
      // ahora ya sí. Con Qemu otra alternativa es hacer un
      // programa lanzador setuid que abra fichero, revoque
      // privilegios y lance Qemu pasándole handler abierto; o que
      // un programa con privilegio de root abra fichero y pase el
      // handler vía un socket UNIX (ver man cmsg)
      if (!(err<0)) {
          err= ioctl(fd, TUNSETOWNER, 1000);
          strcpy(dev, ifr.ifr_name);
          printf("%s\n",ifr.ifr_name);
      }
    }

    if( err<0) {
  perror("falló");
        close(fd);
        return err;
    }
    return 0;
}
```

Apéndice A

GNU Free Documentation License

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent

copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document’s license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

Ñ. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (C) YEAR YOUR NAME.
>
> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.