Manual sobre software libre

ÍNDICE

Configuración del interfaz de red en Linux Debian.

Activación en arranque.

El script */etc/init.d/networking* es el encargado de levantar la red en el arranque. Toma su información del archivo */etc/network/interfaces*. Este archivo contiene los datos necesarios para invocar los comandos anteriores por parte del script networking. El contenido de */etc/network/interfaces* podría ser:

```
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)
# The loopback interface
iface lo inet loopback
auto eth0 eth1
iface eth0 inet static
      address 192.168.1.1                  //dirección IP.
      netmask 255.255.255.0                //máscara
      broadcast 192.168.1.255              //dirección de broadcast
      network 192.168.1.0                  //dirección de red
      gateway 192.168.1.200                //puerta de enlace
iface eth1 inet dhcp
```

Si no se ha configurado la red durante el proceso de instalación, editar este archivo con un editor de textos; respetar las indentaciones. Para que los cambios tengan efecto debemos relanzar el script /etc/init.d/networking con el comando:

```
service networking restart
```

ifup, ifdown.

Estos comandos son la forma más fácil de levantar y bajar una interfaz de red. Toman su información del archivo /etc/network/interfaces ya mencionado. El script /etc/init.d/networking usa estos comandos para levantar la red en el arranque.

ifup eth0

levanta la interfaz de red eth0, primera tarjeta de red.

ifdown eth0

baja la interfaz de red eth0, primera tarjeta de red.

ifconfig

para comprobar estado de las interfaces activas.

Activación manual.

El comando ifconfig nos pemtie ver la configuración de la red (sin argumentos) o activarla y desactivarla en cualquier momento por ejemplo:

ifconfig eth0 192.168.1.1 netmask 255.255.255.0 broadcast 192.168.1.255

configura una interfaz de red sobre la primera tarjeta (eth0), con número IP de máquina 192.168.1.1, máscara de red 255.255.255.0, y dirección de difusión 192.168.1.255.

ifconfig eth0 down

baja la interfaz de red correspondiente a la primera tarjeta.

ifconfig eth0 up

levanta la intefaz de red correspondiente a la primera tarjeta.

route.

Además de configurar la interfaz es necesario indicar a qué red se accede a través de ella. Esto se hace con el comando route .

route

muestra las rutas configuradas. Una salida típica es

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.1.0	*	255.255.255.0	U	0	0	0	eth0

route add -net 192.168.1.0 netmask 255.255.255.0 dev eth0

agrega ruta correspondiente a la red 192.168.1.0.

Si la máquina accede a otras redes a través de otra máquina ("gateway"), debe indicarse el número IP de esta máquina "gateway".

route add default gw 192.168.1.200 metric 1

agrega ruta por defecto hacia una máquina gateway de número IP 192.168.1.200, sobre el mismo cable.

route

muestra las rutas habilitadas.

route del -net 192.168.1.0 netmask 255.255.255.0

borra la ruta creada antes.

ping.

Este comando verifica el estado de una conexión de red enviando un paquete hacia una máquina destino y esperando su respuesta. Si no se le indica una cantidad con la opción -c, continúa enviando y recibiendo paquetes hasta digitar Ctrl-C.

ping -c3 tisanuro

verifica conexión hacia la máquina tisanuro enviando y recibiendo 3 paquetes. Produce esta salida:

```
PING tisanuro.nsk.com.uy (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=32 time=0.3 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=32 time=0.2 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=32 time=0.3 ms
--- tisanuro.nsk.com.uy ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.2/0.3 ms
```

ping 127.0.0.1
verifica la conexión a la propia máquina ("loopback").

ping -c3 192.168.1.1
verifica la conexión al número IP de la propia máquina fijado para la tarjeta eth0 en el ejemplo anterior.

ping -c3 192.168.1.200
verifica la conexión a la máquina fijada como gateway en el ejemplo anterior.

ping –c3 -R tisanuro.nsk.com.uy
verifica conexión a la máquina indicada, con 3 paquetes, pidiendo registro de la ruta de ida y vuelta con la opción -R; esta opción no es aceptada por todas las máquinas.

Nota. Por razones de seguridad, muchas máquinas conectadas a Internet han deshabilitado la respuesta al comando ping.

Instalación de paquetes Debian.

102.4.1. Introducción

En esta sección estudiaremos como se gestionan los **paquetes (programas)** en Debian y sus distribuciones derivadas. La forma en que se hará será muy similar a la gestión de paquetes **rpm** que se verá más adelante. Los paquetes, que realmente son ficheros binarios, se llaman así porque no solo contienen el programa que se desea instalar, sino que además incluyen los ficheros (scripts) de configuración, la documentación y sus dependencias.

Es muy importante conocer lo que significa "**dependencias**", este término pone de manifiesto la necesidad que tienen los programas que otros estén presentes para poder funcionar. Por ejemplo, si pretendemos instalar el paquete "X" que es un entorno gráfico, posiblemente tenga dependencias, es decir, que necesite de otros paquetes sobre los que apoyarse para funcionar correctamente, por ejemplo librerías.

Los nombres de los paquetes Debian tienen la siguiente estructura:

nombre-del-paquete_nombre-del-paquete-build_arquitectura.deb

donde…:

- **nombre-del-paquete**: Será corto y descriptivo, si esta formado por varias palabras suelen estar separadas por guiones.
- **nombre-del-paquete**: Varía en cada revisión y suele ser numérica siguiendo el esquema siguiente: major.minor.patchlevel.
- **build**: Indica la versión del paquete.
- **arquitectura**: Plataforma hardware para la cual fue diseñada la compilación del paquete.
- **deb**: Es la extensión del fichero que, para los paquetes Debian, es siempre **.deb**

Ejemplos:

ethereal_0.8.13-2_i386.deb; arj_3.10.22-9_i386.deb

102.4.2. Base de datos de los paquetes

La información correspondiente a los paquetes que hay instalados se conserva en una base de datos que suele estar en***/var/lib/dpkg***. Dentro de este directorio tenemos el fichero ***/var/lib/dpkg/status*** que contiene la totalidad de paquetes conocidos por dpkg con su estado y también el fichero ***/var/lib/dpkg/available*** que contiene los paquetes que hay disponibles y se pueden instalar. Estos ficheros son muy similares y nos darán bastante información: quién mantiene el paquete, su tamaño, versión, dependencias, descripción, etc. Es muy útil para saber que uso tiene un paquete o ponerse en contacto con su desarrollador.

102.4.3. Herramientas para la gestión de paquetes .deb

El sistema Debian tiene varias herramientas para el manejo de los paquetes, pero los cuatro comandos principales son **dpkg, apt-get, deselect y alien.**

102.4.3.1. dpkg

Esta herramienta permite manejar los paquetes de forma individual tratando directamente con los ficheros **.deb**. Es el núcleo del sistema de empaquetado Debian y equivalente al comando **rpm** en otras distribuciones.

Permite instalar, actualizar, suprimir y gestionar los paquetes. Su uso se suele limitar para forzar instalaciones, arreglar dependencias rotas y el más común, ver los paquetes instalados.

Este comando no gestiona las dependencias. Si al instalar un paquete con **dpkg** faltaran dependencias informará de ello y tendremos que instalarlas previamente o en la misma línea de comando que nos ha dado el aviso. **Un aspecto importante de esta orden es la necesidad de haber descargado previamente los paquetes a instalar.**
La herramienta dpkg usa muchos de los archivos de la carpeta ***/var/lib/dpkg*** ya mencionada anteriormente. Su sintaxis es:
dpkg orden [paquete/s]
Donde **orden** puede ser:
-i; --install Sirve para instalar el/los paquete/s que se le pase/n como segundo parámetro. (Debe escribirse el nombre completo del paquete). Si el paquete que se pretende instalar tiene dependencias de otros paquetes que no se encuentren instalados el programa lo detectará, y será necesario instalar previamente (o en la misma línea) aquellos abortando la operación.
La opción también se puede usar para actualizar un paquete, pero hay que tener en cuenta que si el paquete estaba instalado lo actualizará si hay una versión más nueva disponible **y si no estaba instalado lo instalará**. Por eso hay que tener cuidado si sólo se quieren actualizar los existentes sin instalar paquetes nuevos.
-r; --remove Elimina el/los paquete/s que se le indique/n como segundo parámetro. No borra los ficheros de configuración para evitar reconfigurar la herramienta si se vuelve a reinstalar. Esta opción también comprueba las dependencias y si el paquete que se pretende desinstalar es una dependencia de otro que esté instalado, abortará la operación.
-P; --purge Elimina todo incluyendo los ficheros de configuración.
-R Esta opción junto con –i instala todos los paquetes contenidos en un directorio que se indique.
-l; --list Da el estado de de los paquetes Debian que aparecen en la base de datos. Si le añadimos una cadena nos muestra los paquetes que en su nombre contienen esa cadena. Es una opción muy usada que devuelve una sola línea por paquete y permite usar comodines.
En el listado resultante aparecerá a la izquierda de cada paquete unas indicaciones que podrán ser, por ejemplo, “un”, “ii” o “rc”. Esos valores indicarán respectivamente paquete sin instalar, paquete instalado o paquete desinstalado pero del que se conservan sus ficheros de configuración.
-s; --status Da información sobre el paquete que le indicamos a continuación. Si esta instalado, tamaño, etc.
-S; --search Busca un paquete de los instalados que contenga la cadena que le indiquemos a continuación. Si en vez de una cadena le pasamos un fichero nos indicará a que paquete corresponde dicho fichero. Este segundo uso es el más común, encontrar un paquete que contiene un fichero, o lo que es lo mismo, el paquete propietario de un fichero
-L; --listfiles Muestra la lista de ficheros que usa el paquete cuyo nombre indiquemos a continuación, vamos, los ficheros que instala. La mayoría de las veces basta con añadir en nombre del paquete, sin versión, pero a veces pueden existir varias y entonces habría que especificarla en el nombre del paquete.
--configure Ejecuta los scripts de configuración del paquete que indiquemos a continuación.
--get-selections Imprime **sólo los nombres** de los paquetes que estén instalados. Permite comodines.
Ejemplos:
#dpkg -i paquete1.deb paquete2.deb (Instala los dos paquetes indicados. Obviamente los archivos **paquete1.deb** y**paquete2.deb** deben existir en el directorio desde donde se invoca el comando **dpkg)**

#dpkg -r zip (Elimina el paquete ya instalado en el sistema de nombre zip)
#dpkg -P apache (Elimina completamente el paquete apache del sistema, incluyendo los ficheros de configuración, algo que no ocurre con la opción anterior: -r)
#dpkg --purge arj (Igual que la anterior pero aplicado al paquete arj)
#dpkg -i -R /var/tmp/packs/ (Instala todos los paquetes que haya en el directorio /var/tmp/packs/ así como los que pudieran encontrarse en subdirectorios que cuelguen de él a todos los niveles de profundidad)
#dpkg -l (Imprime por la salida estandar la relación ordenada alfabéticamente de todos los paquetes instalados en el sistema)
#dpkg -l apache (igual que la anterior pero sólo la información correspondiente al paquete *apache*)
#dpkg --get-selections xserver* zip* (Imprime por la salida estandar la relación ordenada alfabéticamente de todos los paquetes cuyos nombres coinciden con los patrones proporcionados)
#dpkg -s wawk (Imprime por la salida estandar el estado del paquete proporcionado así como información complementaria de interés)
#dpkg -S /usr/bin/basename (Busca entre los todos ficheros de todos los paquetes instalados el archivo proporcionado o patrón de coincidencia)
#dpkg -L coreutils (Imprime por la salida estandar la relación completa de archivos que el paquete -en este caso coreutils- a depositado en el arbol de directorios después de instalarlo)
Nota: La ejecución y posterior salida de un comando en pantalla puede variar dependiendo de la distribución Linux, pero básicamente contendrá la misma información con distinto formato.

102.4.3.2. Gestor APT (Advanced Packaging Tool)

Con la herramienta anterior, cuando se quiere instalar algo, puede resultar que se necesite a su vez 3 ó 4 paquetes más y, cuando se van a instalar estos, estos segundos necesitan de otros pudiendo la historia convertirse en una auténtico laberinto existiendo incluso dependencias cruzadas. Como ya dijimos antes, **dpkg** instala los paquetes individualmente sólo informando de las dependencias pero no instalándolas. Esta nueva herramienta gestiona esas dependencias instalándote todo lo necesario.
El programa **apt-get** sirve para automatizar la gestión de paquetes en las distribuciones que derivan de Debian. Su principal ventaja es que resuelve él mismo las dependencias y, si quieres instalar un paquete que tiene dependencias y estas a su vez otras resultando que hay que instalar un número determinado de paquetes, nos informará de ello mostrándonos la lista de paquetes que se instalarán, nos consultará si deseamos proseguir y, si se contesta afirmativamente, lo instalará todo. Esta herramienta instala todas las dependencias requeridas (**depends**), pero ignora los paquetes recomendados (**recommends**) y sugeridos (**suggests**) que también puede haberlos.
Para poder controlar todo, **apt-get** en vez de trabajar directamente con paquetes lo hace con **repositorios de paquetes,** que satisfacen las dependencias de manera automática. Leerá una lista de paquetes del repositorio, creará un árbol de dependencias, y determinará que paquetes son requisitos previos obligatorios y que todavía no están instalados. También puede ser que le **sugiera**instalar algún otro paquete.
Estos repositorios contendrán todos los paquetes necesarios y pueden estar en local (en un cd, dvd o directorio), aunque lo más habitual es que sean remotos, estén online y que sean mantenidos y actualizados a diario siendo el inconveniente de esta segunda opción el tiempo de descarga, pero con la gran ventaja de que con un solo comando podemos actualizar todos los paquetes del sistema.
Los paquetes contenidos en los repositorios dependerán unos de otros, o de otros paquetes procedentes de otros repositorios. El sistema **APT** puede gestionar varios repositorios de distintos sitios, y cuando instala un paquete, también instala sus dependencias (si las encuentra).

102.4.3.2.1 El fichero /etc/apt/sources.list

La configuración de los repositorios se encuentra en el fichero ***/etc/apt/sources.list*** que es el que indica al comando **apt-get** donde coger los paquetes para su instalación.

Un ejemplo de este fichero para una Debian de Ubuntu sería:

$cat /etc/apt/sources.list

```
#deb cdrom:[Ubuntu 11.04 _Natty Narwhal_ - Release i386 (20110427.1)]/ natty
main restricted
deb http://es.archive.ubuntu.com natty  main  restricted
deb-src http://es.archive.ubuntu.com/ubuntu/ natty  main  restricted
##Major bug fix updates produced after the final release of the distribution.
deb http://es.archive.ubuntu.com/ubuntu/ natty-updates  main  restricted
deb-src http://es.archive.ubuntu.com/ubuntu/ natty-updates  main  restricted
```

La primera línea, que se encuentra comentada, indica el dispositivo desde donde se instaló nuestro Linux y podría interesar quitar el comentario para actualizar desde el CD/DVD. El resto de líneas indican fuentes desde donde se puede recuperar información y paquetes al sistema local.
Posteriormente habría que usar el comando **apt-get update** para sincronizar la información que figura en su base de datos local con las fuentes especificadas en el fichero. La sincronización debería de realizarse siempre antes de instalar o actualizar un paquete y también después de modificar ***/etc/apt/sources.list*.** También se recomienda poner los recursos más rápidos al inicio del fichero. Se pueden añadir comentarios poniendo al principio de la línea el símbolo **#**.

102.4.3.2.2 apt-get

La sintaxis del comando es:

apt-get [opción/es] orden [paquete/s]

La **orden** podrá ser:

update Actualiza la base de datos, solo la lista de paquetes y versiones disponibles, desde los repositorios. Se suele utilizar esta opción después de modificar el fichero sources.list.

install Instala el paquete que indiquemos a continuación. Si el paquete ya estuviera instalado lo actualiza. Ejecute **apt-get update** antes de actualizar paquetes para asegurarse de que la base de datos local muestre las últimas versiones disponibles. Si el nombre del paquete va seguido de – (menos) indica que en vez de instalarlo hay que desinstalarlo.
Cuando se ordena la instalación de un paquete, el apt-get revisa primero si ya fue descargado, si no lo fue, irá al primer recurso del sources.list a buscar la versión más nueva del programa, y si este tiene dependencias se añadirán a la lista de instalación.

remove Borra el paquete que indiquemos a continuación. Hay que tener en cuenta que borra el paquete indicado pero no los paquetes que tuvieron que instalarse como requisito previo a aquel aunque no vayan a ser necesarios según nuestro árbol de dependencias. Esta orden avisa de esta eventualidad pero no borra esos paquetes. También indica los paquetes no necesarios que pudiera haber de antes. Tampoco borra los archivos de configuración del paquete.

autoremove Eliminará los paquetes que le indiquemos junto con sus dependencias que no vayan a ser necesarias para los paquetes que queden instalados en el sistema. Incluirá también aquellas dependencias innecesarias para el sistema aunque no hallan sido instaladas por el/los que se pretenden eliminar. La opción **remove** junto con **--auto-remove** es equivalente.
Si se usa autoremove sin ningún nombre de paquete, los paquetes que no se estén usando y que se

instalaron como dependencias de otros se eliminarán de su sistema de forma automática.
purge Junto con el nombre de un paquete lo desinstala y borra sus archivos de configuración.
upgrade Actualiza **todos** los paquetes instalados de los que haya nuevas versiones disponibles. También es muy conveniente ejecutar apt-get update antes de su uso.
clean Elimina todos los archivos de paquetes descargados y que estén en la cache. Cuando apt-get instala un programa, guarda una copia del fichero deb en los directorios ***/var/cache/apt/archives*** y***/var/cache/apt/archives/partial***. Con el tiempo esos directorios pueden llegar a ocupar mucho espacio, para limpiar ambos directorios se usa esta opción.
check Actualiza la cache y verifica las dependencias.
Entre las **opciones** tenemos las siguientes:
-f Intenta corregir dependencias rotas.
-d Descarga un paquete pero no lo instala.
-s Simula la ejecución de la orden pero no la realiza. Se usa por ejemplo para ver si un paquete depende de otros paquetes o que implicaría un upgrade.
-y Responde a todo que si, cuidado con esta opción.
-h Muestra la ayuda.
Ejemplos:
#apt-get update (Actualiza la cache local con la de los repositorios de referencia indicados en /etc/apt/sources.list)
#apt-get install vim-gtk (Instala el paquete vim-gtk y todas sus dependencias)
#apt-get install tzdata (Instala el paquete tzdata y todas sus dependencias)
#apt-get install libqt* (Instala todos los paquetes cuyos nombres coinciden con el patrón asi como las dependencias que necesitan)
#apt-get install -s gcl (Instala el paquete gcl de forma simulada, no produce n ingún efecto real en el sistema)
#apt-get install -d gcl (Descarga un paquete sin instalarlo y luego podrá visualizar la información del paquete con **dpkg --info**. Generalmente los archivos descargados están en ***/var/cache/apt/archives/)***
#apt-get remove -s gcl (Elimina el paquete gcl de forma simulada, no produce n ingún efecto real en el sistema)
#apt-get clean (Borra totalmente el repositorio local que contiene los ficheros de los paquetes descargados)

102.4.3.2.3 apt-cache

Busca información sobre paquetes en nuestro sistema, en la caché local. Puede utilizar expresiones regulares.
Su sintaxis es:
apt-cache orden paquete/patron_de_busqueda
Veamos sus opciones más comunes:
search Busca, por su nombre o comentario, un paquete en la base de datos local APT.
show Muestra la descripción del paquete.
Ejemplos:
#apt-cache search torrent
#apt-cache search "linux loader"
#apt-cache show torrent

Configuración de un cortafuegos en un sistema Linux

La herramienta que proporciona Linux como firewall se llama **iptables**, y se encuentra integrada en el propio nucleo del sistema. Por tanto no requiere de la instalación de ningún paquete adicional.

Funcionamiento de iptables:

iptables permite al administrador del sistema definir reglas acerca de qué hacer con los paquetes de red. Las reglas se agrupan en *cadenas*: cada cadena es una lista ordenada de reglas. Las cadenas se agrupan en *tablas*: cada tabla está asociada con un tipo diferente de procesamiento de paquetes.

Cada regla especifica qué paquetes la cumplen y un *destino* que indica qué hacer con el paquete si éste cumple la regla. Cada paquete de red que llega a una computadora o que se envía desde una computadora recorre por lo menos una cadena y cada regla de esa cadena se comprueba con el paquete.

Si la regla cumple con el datagrama, el recorrido se detiene y el destino de la regla dicta lo que se debe hacer con el paquete y no se comprueban las siguientes reglas.

Si el paquete alcanza el fin de una cadena sin haber cumplido ninguna regla se aplica la politica por defecto definida para esa cadena. Solo las cadenas predefinidas tienen políticas.

Tablas

Hay tres tablas ya incorporadas, cada una de las cuales contiene ciertas cadenas predefinidas. Inicialmente, todas las cadenas están vacías.

- filter table (*Tabla de filtros*) — Esta tabla es la responsable del filtrado (es decir, de bloquear o permitir que un paquete continúe su camino). Todos los paquetes pasan a través de la tabla de filtros. Contiene las siguientes cadenas predefinidas y cualquier paquete pasará por una de ellas:
  - *INPUT chain* (Cadena de ENTRADA) — Todos los paquetes destinados a este sistema atraviesan esta cadena (y por esto se la llama algunas veces *LOCAL_INPUT* o *ENTRADA_LOCAL*)
  - *OUTPUT chain* (Cadena de SALIDA) — Todos los paquetes creados por este sistema atraviesan esta cadena (a la que también se la conoce como *LOCAL_OUTPUT* o *SALIDA_LOCAL*)
  - *FORWARD chain* (Cadena de REDIRECCIÓN) — Todos los paquetes que meramente pasan por este sistema para ser encaminados a su destino recorren esta cadena
- nat table (*Tabla de traducción de direcciones de red*) — Esta tabla es la responsable de configurar las reglas de reescritura de direcciones o de puertos de los paquetes. El primer paquete en cualquier conexión pasa a través de esta tabla; los veredictos determinan como van a reescribirse todos los paquetes de esa conexión. Contiene las siguientes cadenas redefinidas:
  - *PREROUTING chain* (Cadena de PRERUTEO) — Los paquetes entrantes pasan a través de esta cadena antes de que se consulte la tabla de ruteo local, principalmente para DNAT (*destination*-NAT o traducción de direcciones de red de destino)
  - *POSTROUTING chain* (Cadena de POSRUTEO) — Los paquetes salientes pasan por esta cadena después de haberse tomado la decisión del ruteo, principalmente para

SNAT (*source*-NAT o traducción de direcciones de red de origen)
    - *OUTPUT chain* (Cadena de SALIDA) — Permite hacer un DNAT limitado en paquetes generados localmente

- mangle table (*Tabla de destrozo*) — Esta tabla es la responsable de ajustar las opciones de los paquetes, como por ejemplo la calidad de servicio. Todos los paquetes pasan por esta tabla. Debido a que está diseñada para efectos avanzados, contiene todas las cadenas predefinidas posibles:
    - *PREROUTING chain* (Cadena de PRERUTEO)
    - *INPUT chain* (Cadena de ENTRADA)
    - *FORWARD chain* (Cadena de REDIRECCIÓN)
    - *OUTPUT chain* (Cadena de SALIDA)
    - *POSTROUTING chain* (Cadena de POSRUTEO)

Además de las cadenas ya incorporadas, el usuario puede crear todas las cadenas definidas por el usuario que quiera dentro de cada tabla, las cuales permiten agrupar las reglas en forma lógica.

Ejemplos de iptables:

```
# Permitir el acceso a nuestro equipo desde la dirección 195.65.34.234
iptables -A INPUT -s 195.65.34.234 -j ACCEPT

# Permitir el acceso a mysql desde la dirección  231.45.134.23
iptables -A INPUT -s 231.45.134.23 -p tcp --dport 3306 -j ACCEPT

# Autorizar el acceso a FTP desde  80.37.45.194
iptables -A INPUT -s 80.37.45.194 -p tcp -dport 20:21 -j ACCEPT

# El puerto 80 de www debe estar abierto, en un servidor web.
iptables -A INPUT -p tcp --dport 80 -j ACCEPT

# Denegar al acceso al FTP, mysql y SSH desde cualquier sitio
iptables -A INPUT -p tcp --dport 20:21 -j DROP
iptables -A INPUT -p tcp --dport 3306 -j DROP
iptables -A INPUT -p tcp --dport 22 -j DROP
```

Comandos iptables

Los comandos de `iptables` son:

- `-A` — Añade la regla `iptables` al final de la cadena especificada. Este es el comando utilizado para simplemente añadir una regla cuando el orden de las reglas en la cadena no importa.

```
iptables -A INPUT -p udp -j DROP
```

- `-D` — Borra una regla de una cadena en particular por número (como el `5` para la quinta regla de una cadena). Puede también teclear la regla entera e `iptables` borrará la regla en la cadena que corresponda.

```
iptables -D INPUT -p udp -j DROP
```

- `-E` — Renombra una cadena definida por el usuario. Esto no afecta a la estructura de la tabla. Tan sólo le evita el problema de borrar la cadena, creándola bajo un nuevo nombre, y

reconfigurando todas las reglas de dicha cadena.

```
iptables -E nombre-de-cadena-viejo nombre-de-cadena-nuevo
```

- `-F` — Libera la cadena seleccionada, que borra cada regla de la cadena. Si no se especifica ninguna cadena, este comando libera cada regla de cada cadena,

  Para tirar todas las cadenas, usamos este comando:

  ```
  iptables -F
  ```

- `-h` — Proporciona una lista de estructuras de comandos útiles, así como una resumen rápido de parámetros de comandos y opciones.
- `-I` — Inserta una regla en una cadena en un punto determinado. Asigne un número a la regla a insertar e `iptables` lo pondrá allí. Si no especifica ningún número, `iptables` posicionará su comando al principio de la lista de reglas.

  Para insertar una regla nueva en el segundo lugar de la cadena de OUTPUT que descarte todo el tráfico TCP dirigido al puerto 80 en cualquier terminal, usamos este comando:

  ```
  iptables -I OUTPUT 2 -p tcp --dport 80 -j DROP
  ```

- `-L` — Lista todas las reglas de la cadena especificada tras el comando. Para ver una lista de todas las cadenas en la tabla `filter` por defecto. La sintaxis siguiente deberá utilizarse para ver todas la lista de todas las reglas de una cadena específica en una tabla en particular:

  ```
  iptables -L OUTPUT
  ```

- `-N` — Crea una nueva cadena con un nombre especificado por el usuario.

  ```
  iptables { -N | --new-chain } cadena
  ```

- `-P` — Configura la política por defecto para una cadena en particular de tal forma que cuando los paquetes atravieses la cadena completa sin cumplir ninguna regla, serán enviados a un objetivo en particular, como puedan ser ACCEPT o DROP.

  ```
  iptables -P INPUT DROP
  ```

- `-R` — Reemplaza una regla en una cadena en particular. Deberá utilizar un número de regla detrás del nombre de la cadena para reemplazar esta cadena. La primera regla de una cadena se refiere a la regla número `1`.

  ```
  iptables -R INPUT 4 -p icmp -j DROP
  ```

- `-X` — Borra una cadena especificada por el usuario. No se permite borrar ninguna de las cadenas predefinidas para cualquier tabla.

  Borrar todas las cadenas creadas por el usuario:

  ```
  iptables -X
  ```

- `-Z` — Pone ceros en los contadores de byte y de paquete en todas las cadenas de una tabla en particular. Para poner a cero los contadores de todas las cadenas:

  ```
  iptables -Z
  ```

Las reglas se aplican en el orden en que se ejecutan dentro de cada cadena. Por tanto, para facilitar la confección del cortafuegos los más útil es crear un script en el que se incluyan todas las reglas, incluyendo en él al principio las ordenes que limpian las reglas de todas la cadenas y establecer las

políticas por defecto de cada cadena predefinida.

El script podría comenzar del siguiente modo:

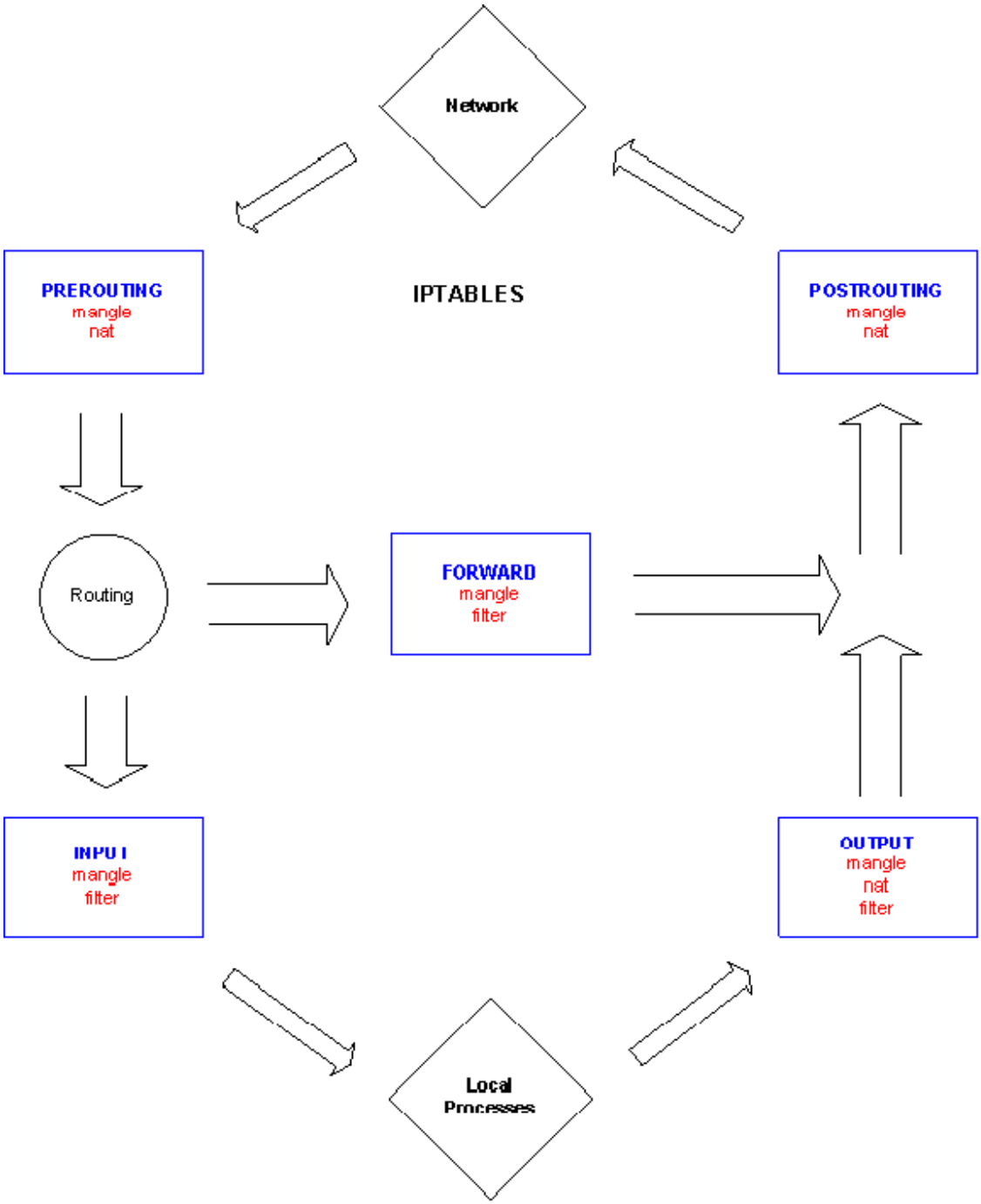
```
## Limpiar las reglas
iptables −F
iptables −X
iptables −Z
iptables −t nat −F

## Establecer politicas por defecto para las cadenas predefinidas
iptables −P INPUT ACCEPT
iptables −P OUTPUT ACCEPT
iptables −P FORWARD ACCEPT
iptables −t nat −P PREROUTING ACCEPT
iptables −t nat −P POSTROUTING ACCEPT

## A partir de aquí pondríamos las reglas de filtrado.
```

De este modo si modificamos alguna regla basta con volver a lanzar el script, con lo que se eliminarán todas las reglas anteriores, volviendo a ejecutarse de nuevo las reglas definidas en el script.

El siguiente esquema muestra la secuencia que sigue un paquete al pasar por las distintas cadenas:



Destinos de reglas

El destino de una regla puede ser el nombre de una cadena definida por el usuario o uno de los destinos ya incorporados `ACCEPT`, `DROP`, . . . (*aceptar*, *descartar*, ...).

Existen los siguientes destinos ya incorporados:

`ACCEPT` (aceptar)

Este destino hace que se acepte el paquete.

`DROP` (descartar)

Este destino hace que netfilter descarte el paquete sin ningún otro tipo de procesamiento.

`RETURN` (retorno)

Hace que el paquete en cuestión deje de circular por la cadena en cuya regla se ejecutó el destino `RETURN`.

Hay muchos destinos de extensión disponibles. Algunos de los más comunes son:

`REJECT` (rechazo)

Este destino tiene el mismo efecto que 'DROP', salvo que envía un paquete de error a quien envió originalmente.

`LOG` (bitácora)

Este destino lleva un *log* o bitácora del paquete. Puede usarse en cualquier cadena en cualquier tabla, y muchas veces se usa para analizar fallos.

Especificaciones de las reglas

La mayoría de las formas de comandos de iptables requieren que se les indiquen una especificación de reglas, que es usada para comparar un subconjunto particular del tráfico de paquetes de red procesados por una cadena. La especificación de regla incluye también un destino que especifica qué hacer con paquetes que son comparados por la regla. Las siguientes opciones se usan (frecuentemente combinadas unas con otras) para crear especificaciones de reglas.

`-j` *`destino`*

`--jump` *`destino`*

Especifica el destino de una regla. El destino es el nombre de una cadena definida por el usuario (creada usando la opción `-N`, uno de los destinos ya incorporados, `ACCEPT`, `DROP`, `QUEUE`, o `RETURN`, o un destino de extensión, como `REJECT`, `LOG`, `DNAT`, o `SNAT`. Si esta opción es omitida en una regla, entonces el comparado de la regla no tendrá efecto en el destino de un paquete, pero los contadores en la regla se incrementarán.

`-i [!]` *`in-interface`*

`--in-interface [!]` *`in-interface`*

Nombre de una interfaz a través de la cual un paquete va a ser recibido (solo para paquetes entrando en las cadenas de INPUT, FORWARD y PREROUTING). Cuando se usa el argumento '!' antes del nombre de la interfaz, el significado se invierte. Si el nombre de la interfaz termina con '+', entonces cualquier interfaz que comience con este nombre será comparada. Si esta opción se omite, se comparará todo nombre de interfaz.

`-o [!]` *`out-interface`*

`--out-interface [!]` *`out-interface`*

Nombre de una interfaz a través de la cual un paquete va a ser enviado (para paquetes entrando en las cadenas de FORWARD, OUTPUT y POSTROUTING). Cuando se usa el argumento '!' antes del nombre de la interfaz, el significado se invierte. Si el nombre de la

interfaz termina con '+', entonces cualquier interfaz que comience con este nombre será comparada. Si esta opción se omite, se comparará todo nombre de interfaz.

`-p [!] protocol`

`--protocol [!] protocol`

Compara paquetes del nombre de protocolo especificado. Si '!' precede el nombre de protocolo, se comparan todos los paquetes que no son el protocolo especificado. Nombres de protocolo válidos son `icmp`, `udp`, `tcp`... Una lista de todos los protocolos válidos puede encontrarse en el archivo /etc/protocols.

`-s [!] origen[/prefijo]`

`--source [!] origen[/prefijo]`

Compara paquetes IP viniendo de la dirección de origen especificada. La dirección de origen puede ser una dirección IP, una dirección IP con un prefijo de red asociado, o un nombre de terminal (*hostname*). Si '!' precede al origen, se comparan todos los paquetes que no vienen del origen especificado.

`-d [!] destino[/prefijo]`

`--destination [!] destino[/prefijo]`

Compara paquetes IP yendo a la dirección de destino especificada. La dirección de destino puede ser una dirección IP, una dirección IP con un prefijo de red asociado, o un nombre de terminal (*hostname*). Si '!' precede al origen, se *matchean* todos los paquetes que no van al destino especificado.

`--destination-port [!] [puerto[[:puerto]]`

`--dport [!] [puerto[[:puerto]]`

Matchea paquetes TCP o UDP (dependiendo del argumento a la opción `-p`) destinados a los puertos o rango de puertos (cuando se usa la forma `puerto:puerto`) especificados. Si '!' precede la especificación de puertos, se *matchean* todos los paquetes TCP o UDP que no están destinados a los puertos o rango de puertos especificados.

`--source-port [!] [puerto[[:puerto]]`

`--sport [!] [puerto[[:puerto]]`

Matchea paquetes TCP o UDP (dependiendo del argumento a la opción `-p`) que vienen de los puertos o rango de puertos (cuando se usa la forma `puerto:puerto`) especificados. Si '!' precede la especificación de puertos, se *matchean* todos los paquetes TCP o UDP que no vienen de los puertos o rango de puertos especificados.

Ejemplos de reglas:

Reglas que limitan los accesos al sistema:

```
# Permitir el acceso FTP  desde la dirección  80.37.45.194
iptables –A INPUT –s 80.37.45.194 –p tcp --dport 20:21 –j ACCEPT

#Abrir el puerto 80 para el servidor web a todo el mundo
iptables –A INPUT –p tcp --dport 80 –j ACCEPT
```

Cerrar los puertos 20 y 21 usados por FTP
```
iptables -A INPUT -p tcp --dport 20:21 -j DROP
```

Cerrar todos los puertos tcp
```
iptables -A INPUT -p tcp -j DROP
```

Ejemplos de reglas para la retransmisión de paquetes:

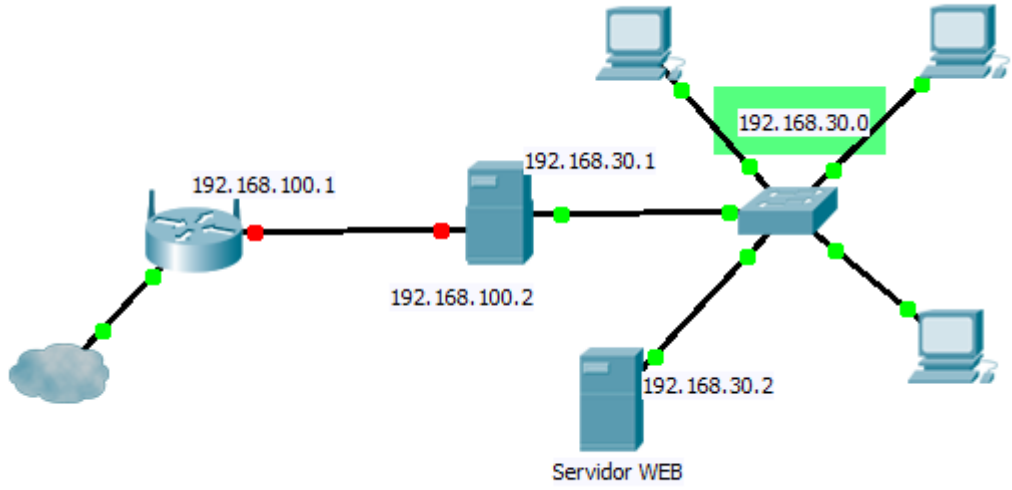
Aceptar los accesos al puerto 80 (servicio web) desde la red 192.168.10.0 que entren desde la interfaz eth1
```
iptables -A FORWARD -s 192.168.10.0/24 -i eth1 -p tcp --dport 80 -j ACCEPT
```

Aceptamos que vayan a puertos https desde la red 192.168.10.0 que entren desde la interfaz eth1
```
iptables -A FORWARD -s 192.168.10.0/24 -i eth1 -p tcp --dport 443 -j ACCEPT
```

Aceptamos que todos consulten el servidor DNS con dirección 192.168.10.4
```
iptables -A FORWARD -d 192.168.10.4  -p tcp --dport 53 -j ACCEPT
iptables -A FORWARD -d 192.168.10.4  -p udp --dport 53 -j ACCEPT
```

Denegar todo lo que procede de la red 192.168.10.0 que entra por eth1 y va dirigido a la red 172.19.0.0
```
iptables -A FORWARD -s 192.168.10.0/24 -i eth1 -d 172.19.0.0/16 -j DROP
```

Ejemplo de script completo para un cortafuegos:
Suponiendo que tenemos un equipo linux que da salida a una red privada a internet:



vamos a crear un script para crear un cortafuegos que permita a los equipos de la red local salir a internet solo para usar los servicios web, dns y correo electronico los demás se deben cortar. Se deben permitir los accesos desde fuera hacia el servidor web, el resto de los puertos conocidos deben ser cortados.

```
## Limpiar las reglas
iptables -F
```

```
iptables –X
iptables –Z
iptables –t nat –F

## Establecer politicas por defecto para las cadenas predefinidas
iptables –P INPUT ACCEPT
iptables –P OUTPUT ACCEPT
iptables –P FORWARD ACCEPT
iptables –t nat –P PREROUTING ACCEPT
iptables –t nat –P POSTROUTING ACCEPT

## Activar el reenvio de paquetes
echo 1 > /proc/sys/net/ipv4/ip_forward

## A partir de aquí pondríamos las reglas de filtrado.
iptables -A FORWARD -s 192.168.30.0/24 -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -s 192.168.30.0/24 -p tcp --dport 25 -j ACCEPT
iptables -A FORWARD -s 192.168.30.0/24 -p tcp --dport 110 -j ACCEPT
iptables -A FORWARD -s 192.168.30.0/24 -p tcp --dport 53 -j ACCEPT
iptables -A FORWARD -s 192.168.30.0/24 -p udp --dport 53 -j ACCEPT
iptables -A FORWARD -p tcp -d 192.168.30.2 --dport 80 -j ACCEPT
iptables -A FORWARD -p tcp -d 192.168.30.2 --dport 1:1024 -j DROP
iptables -A FORWARD -s 192.168.30.2 -p tcp ! --dport 1:1024 -j ACCEPT
iptables -A FORWARD -s 192.168.30.0/24 -j DROP
```

Destinos de reglas para NAT

`REDIRECT`

Este destino se utiliza para redireccionar el puerto con la opción '--to-ports' detro del mismo sistema.

`DNAT`

Este destino hace que la dirección (y opcionalmente el puerto) de destino del paquete sean reescritos para traducción de dirección de red. Mediante la opción '--to-destination' debe indicarse el destino a usar. Esto es válido solamente en las cadenas de SALIDA y PRERUTEO dentro de la tabla de nat. Esta decisión se recuerda para todos los paquetes futuros que pertenecen a la misma conexión y las respuestas tendrán su dirección y puerto de origen cambiados al original (es decir, la inversa de este paquete).

`SNAT`

Este destino hace que la dirección (y opcionalmente el puerto) de origen del paquete sean reescritos para traducción de dirección de red. Mediante la opción '--to-source' debe indicarse el origen a usar. Esto es válido solamente en la cadena de POSRUTEO dentro de la tabla de nat y, como DNAT, se recuerda para todos los paquetes que pertenecen a la misma conexión.

`MASQUERADE`

Esta es una forma especial, restringida de SNAT para direcciones IP dinámicas, como las que proveen la mayoría de los proveedores de servicios de Internet (ISPs) para módems o línea de abonado digital (DSL). En vez de cambiar la regla de SNAT cada vez que la dirección IP cambia, se calcula la dirección IP de origen a la cual hacer NAT fijándose en la

dirección IP de la interfaz de salida cuando un paquete coincide con esta regla. Adicionalmente, recuerda que conexiones usan MASQUERADE y si la dirección de la interfaz cambia (por ejemplo, por reconectarse al ISP), todas las conexiones que hacen NAT a la dirección vieja se olvidan.

Conversión NAT con iptables

La conversión NAT permite modificar las direcciones de origen (SNAT) o de destino (DNAT) de los paquetes.

El uso típico de la conversión NAT es permitir las conexiones a redes públicas desde redes privadas. Su funcionamiento es el siguiente: para acceder a internet desde una red privada es necesario que en el punto de salida a la red pública se sustituya la dirección IP privada por una pública. Todas las conversiones que se realizan se van registrando en una tabla con el fin de realizar la convesión inversa en los paquetes de respuesta.

Para transformar la dirección de origen hay que utilizar el destino SNAT.

Ejemplo que transforma las direcciones de la red **192.168.186.0/24** por la dirección **195.10.20.10**:

```
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.186.0/24 -j SNAT --to-source 195.10.20.10
```

De esta forma en todos los paquetes que salgan por el interfaz eth0 y procedan de la red **192.168.186.0/24** se sustituirá su dirección de origen por **195.10.20.10.** Cada transformación que se haga se almacena en la tabla NAT de ese modo en el paquete de respuesta se volverá a cambiar la dirección, en este caso la de destino para que continúe su camino por la red interna.

Un caso especial para conversiones de la dirección de origen consiste en el uso del destino MASQUERADE, esta opción modifica la dirección de origen con la dirección del interfaz de salida en lugar de hacerlo por una dirección concreta. Esto se suele utilizar cuando la dirección del interfaz de salida puede cambiar por asignarse mediante DHCP. Si la dirección es estática su efecto es el mismo que el del destino SNAT.

El ejemplo anterior con la opción MASQUERADE sería:

```
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.186.0/24 -j MASQUERADE
```

Seguridad de la conversión NAT

La conversión NAT permite una seguridad total en la red interna con respecto a los accesos desde el exterior. Es decir, no permite ningún acceso desde el exterior, ya que sólo pueden pasar aquellos paquetes que son respuesta a una petición desde el interior, pero no admite comunicaciones que se inicien desde fuera de la red interna.

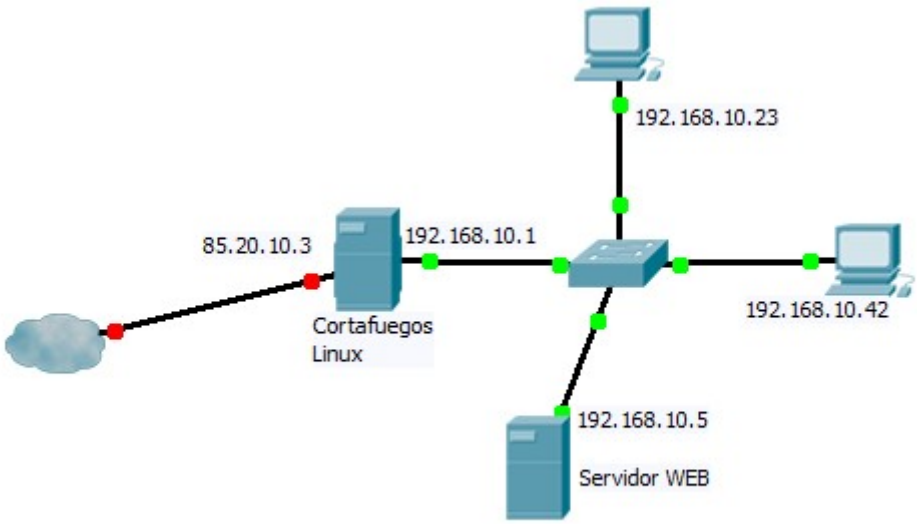
Esto se cumple para conversiones SNAT o MASQUERADE, que transforman las direcciones internas de una red privada en los paquetes que salen al exterior.

Es posible abrir el acceso desde el exterior a algunos equipos de la red interna usando la conversión

DNAT.

Usos de la conversión DNAT

Supangamos que tenemos una red privada que tiene acceso a Internet a través de una máquina Linux que actúa de cortafuegos y conversor NAT. Al mismo tiempo, en la red interna disponemos de un servidor Web que deseamos sea accesible desde el exterior.



Veamos como debemos configurar iptables en el cortafuegos para que permita el acceso a Internet haciendo un enmascaramiento de la red 192.168.10.0 y permitir el acceso desde el exterior al servidor Web 192.168.10.5.

El script de iptables podría ser el siguiente:

```
## Limpiar las reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

## Establecer politicas por defecto para las cadenas predefinidas
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT

## Activar el reenvio de paquetes
echo 1 > /proc/sys/net/ipv4/ip_forward

## Reglas de enmascaramiento
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.10.0/24 -j MASQUERADE
```

```
## Abriendo el acceso al servidor WEB
iptables -t nat -A PREROUTING -i eth0 -p tcp -d 85.20.10.3 --dport 80 -j DNAT
--to 192.168.10.5:80
```

Una vez lanzado este script los equipos de la red privada 192.168.10.0 podrán salir a internet poniendo como puerta de enlace 192.168.10.1 y desde fuera se accederá al servidor web con la url http://85.20.10.3 ya que los paquetes que lleguen a esta dirección y al puerto 80 se redirigen a la dirección interna 192.168.10.5:80.

Especificaciones de las reglas, ampliación:

`--tcp-flags [!] mask comp`

Matchea paquetes TCP que tienen marcadas o desmarcadas ciertas banderas del protocolo TCP. El primer argumento especifica las banderas a examinar en cada paquete TCP, escritas en una lista separada por comas (no se permiten espacios). El segundo argumento es otra lista separada por comas de banderas que deben estar marcadas dentro de las que se debe examinar. Estas banderas son: SYN, ACK, FIN, RST, URG, PSH, ALL, y NONE. Por lo tanto, la opción "`--tcp-flags SYN, ACK, FIN, RST SYN`" solo va a *matchear* paquetes con la bandera SYN marcada y las banderas ACK, FIN y RST desmarcadas.

`[!] --syn`

Matchea paquetes TCP que tienen la bandera SYN marcada y las banderas ACK, FIN y RST desmarcadas. Estos paquetes son los que se usan para iniciar conexiones TCP. Al bloquear tales paquetes en la cadena de INPUT, se previenen conexiones TCP entrantes, pero conexiones TCP salientes no serán afectadas. Esta opción puede combinarse con otras, como `--source`, para bloquear o dejar pasar conexiones TCP entrantes solo de ciertas terminales o redes. Esta opción es equivalente a "`--tcp-flags SYN, RST, ACK SYN`". Si '!' precede a `--syn`, el significado de la opción se invierte.

Protocolo ICMP

Los paquetes que usan el protocolo de control de mensajes de internet (Internet Control Message Protocol, ICMP) pueden ser seleccionados usando la siguiente opción cuando se especifique `-p icmp`:

- `--icmp-type` Selecciona el nombre o el número del tipo ICMP que concuerde con la regla. Se puede obtener una lista de nombres válidos ICMP tecleando el comando

  `iptables -p icmp -h`.

Módulos con opciones de selección adicionales

Las opciones de selección adicionales, que no son específicas de ningún protocolo en particular están también disponibles a través de módulos que se cargan cuando el comando `iptables` los necesite. Para usar una de estas opciones deberá cargar el módulo por su nombre incluyendo `-m <nombre-modulo>` en el comando `iptables` que crea la regla.

El módulo `limit` le permite poner un límite en el número de paquetes que podrán ser seleccionados por una regla en particular. Esto es especialmente beneficioso cuando se usa la regla de logging ya que hace que el flujo de paquetes seleccionados no llene nuestros ficheros log con mensajes repetitivos ni utilice demasiados recursos del sistema.

- `--limit` — Configura el número de coincidencias en un intervalo de tiempo, especificado con un número y un modificador de tiempo ordenados en el formato *<número>/<tiempo>*. Por ejemplom si usamos `--limit 5/hour` sólo dejaremos que una regla sea efectiva cinco veces a la hora,

  Si no se utiliza ningún número ni modificador de tiempo, se asume el siguiente valor por defecto: `3/hour`.

- `--limit-burst` — Configura un límite en el número de paquetes capaces de cumplir una regla en un determinado tiempo. Esta opción deberá ser usada junto con la opción `--limit`, y acepta un número para configurar el intervalo de tiempo (threshold).

  Si no se especifica ningún número, tan sólo cinco paquetes serán capaces inicialmente de cumplir la regla.

El módulo `state`, que utiliza la opción `--state`, puede seleccionar un paquete con los siguientes estados de conexión particulares:

- `ESTABLISHED` El paquete seleccionado se asocia con otros paquetes en una conexión establecida.
- `INVALID` El paquete seleccionado no puede ser asociado a una conexión conocida.
- `NEW` El paquete seleccionado o bien está creando una nueva conexión o bien forma parte de una conexión de dos caminos que antes no había sido vista.
- `RELATED` El paquete seleccionado está iniciando una nueva conexión en algún punto de la conexión existente.

Estos estados de conexión se pueden utilizar en combinación con otros separándolos mediante comas como en `-m state --state INVALID,NEW`.

Para seleccionar una dirección MAC hardware de un dispositivo Ethernet en particularm utilice **el módulo `mac`**, que acepta `--mac-source` con una dirección MAC como opción. Para excluir una dirección MAC de una regla, ponga un punto de exclamación (`!`) delante de la opción `--mac-source`.

Instalación y configuración de servidores «proxy»:

Tipos de Proxy. (caché, transparentes, inverso).

Instalación de un servidor Proxy.

Instalación de un cliente Proxy.

Configuración de ACLs en un servidor Proxy.

Control del ancho de banda mediante Proxy.

Definición y tipos de proxy:

Un servidor proxy es un equipo con acceso a internet, con un programa proxy instalado, este programa permite el acceso a intenet a los equipos de una red local a través del servidor proxy, retransmitiendo éste los paquetes de los servicios http, https, ftp, etc hacia la red pública.

A parte de esta, que es la función principal de un servidor proxy, puede tener otras funciones, así nos podemos encontrar con distintos tipos de servidores proxy según las funciones adicionales que desempeñe:

- Proxy caché: provee la funcionalidad de cache, mediante la cual se almacenan localmente las paginas consultadas por los usuarios de forma que incrementa la rapidez de acceso a la información web y ftp.
- Proxy tranparente: cuando el cliente no detecta que su conexión a internet se realiza a través de un servidor proxy. Esto se consigue redireccionando los paquetes en el enrutador hacia el puerto del servidor proxy. De este modo se consigue que todos los paquetes pasen por el proxy sin que los clientes tengan conocimiento de ello.
- Proxy con ACL: permite limitar los accesos a Internet de los usuarios de la red local, según direcciones de origen, dominios de destino, puertos, días y horas, etc.
- Proxy inverso: se utiliza para acelerar los accesos a un servidor web interno en la red local, o bien, configurar un balanceo de carga hacia varios servidores internos en la red local que gestionen una misma web.

Instalación de un servidor proxy.

Existen muchas aplicaciones proxy para distintas plataformas. Squid es un proxy para Linux con todas las funcionalidades posibles.

Vamos a estudiar la instalación de squid y su configuración básica para crear un proxy caché.

Para instalar squid basta ejecutar el comando:

```
apt-get install squid
```

Squid dispone de un único fichero de configuración:

```
/etc/squid/squid.conf
```

En este fichero las líneas comentadas comienzan por # al principio de la línea.

Las directivas de configuración deben comenzar desde el principio de la línea, no se puede incluir un espacio en blanco al principio.

Opciones básicas de configuración:

Parámetro http_port

Con este parámetro configuramos el puerto de escucha de nuestro servidor squid, por defecto es el puerto 3128, pero también puede ser utilizado el 8080.

```
http_port 3128
```

Si se desea incrementar la seguridad, puede vincularse el servicio a una IP que sólo se pueda acceder desde la red local. Considerando que el servidor utilizado posee una IP 192.168.1.254, puede hacerse lo siguiente:

```
# Default: http_port 3128
http_port 192.168.1.254:3128
http_port 192.168.1.254:8080
```

Parámetro cache_mem

Establece la cantidad de memoria RAM dedicada para almacenar los datos mas solicitados. Esta opción viene comentada por lo tanto se descomenta para darle un valor apropiado.

```
# cache_mem 8 MB
```

por

```
cache_mem 50 MB
```

El valor ya depende del administrador y de la carga que tenga el squid.

Parámetros cache_swap

Dentro del cache_swap, existen dos parámetros: cache_swap_low cache_swap-hight Con estos le indicamos a squid que mantenga los niveles del espacio del área de intercambio o también conocido como swap. Estos parámetros viene siempre desactivados, por tanto los buscaremos para activarlos.

```
#cache_swap_low 90
#cache_swap_high 95
```

por

```
cache_swap_low 90
cache_swap_high 95
```

Con esto decimos al squid que mantenga los niveles del espacio del área de intercambio entre 90% y 95%.

Parámetros maximum_object_size

Utilizamos esta directiva para indicar el tamaño máximo para los objetos a almacenar en la cache.

#maximum_object_size 4096 KB

por

maximum_object_size 10240 MB

Parámetro hierarchy_stoplist

Este parámetro es útil para indicar a squid que páginas que contengan ciertos caracteres no deben almacenarse en cache. También se pueden incluir como sitios de webmail y páginas locales en su red ya que no seria necesario almacenarlas en el cache, esta opción ya viene habilitada solamente tendremos que modificarle algunos datos de la misma.

hierarchy_stoplist cgi-bin ?

por

hierarchy_stoplist cgi-bin ? hotmail gmail yahoo

Parámetro cache_dir

Con este parámetro establecemos el tamaño que deseamos que tenga la cache en el disco, para ello tendremos que habilitar y modificar el siguiente dato.

#cache_dir ufs /var/spool/squid 100 16 256

por

cache_dir ufs /var/spool/squid 700 16 256

Con esto establecemos el tamaño que deseamos que tenga la cache en el disco, se puede incrementar hasta el tamaño que desee el administrador, nosotros establecemos 700MB de cache con 16 directorios subordinados y 256 niveles cada uno.

Configuración del cliente

Los navegadores necesitan ser configurados para acceder a Internet a través de un servidor proxy. Veamos como se configuran algunos de los más usados:

- **Firefox**: Menú *Herramientas/Opciones*, seleccionamos *Avanzado* y la pestaña *Red*, aquí en el apartados *Conexión* pinchamos en botón *Configuración*, en la ventana que se abre seleccionamos *Configuración manual del proxy*: indicamos la dirección IP del servidor proxy y el puerto.
- **Chrome**: Pinchamos en Herramientas/*Opciones*, seleccionamos *Avanzado* y en la sección *Red* pinchamos en el botón *Cambiar la configuración de proxy...*, en la ventaña que se abre seleccionamos la pestaña *Conexiones*, donde pinchamos el botón *Configuración de LAN*, seleccionamos *Usar un servidor proxy para la LAN*, indicando la dirección IP y el puerto del servidor proxy.

- **Internet Explorer**: Menú *Herramientas/Opciones de Internet,* seleccionamos la pestaña *Conexiones*, donde pinchamos el botón *Configuración de LAN*, seleccionamos *Usar un servidor proxy para la LAN*, indicando la dirección IP y el puerto del servidor proxy.

Configuración de ACLs en un servidor Proxy

Reglas acl

Una ACL es una definición de control de acceso, que utiliza squid para identificar un conjunto de paquetes, existen varios tipos de reglas ACL que aparecen en la tabla.

src	time
dts	url_regex
srcdomain	urlpath_regex
dstdomain	port
srcdom_regex	macaddress
dstdom_regex	password

Regla Tipo src

Esta reglas especifica una o varias dirección IP de origen o un segmento de red con su mascara de red. Nomenclatura:

```
acl [Nombre] src [Contenido]
```

Ejemplos:

1) El nombre de la regla es llamada redlocal y a ella se asignan un segmento de red 192.168.1.0 con máscara de 24 bits.

```
acl redlocal src 192.168.1.0/24
```

2) Esta regla es llamada jefes a ella pertenecen algunas IP de nuestro segmento de red.

```
acl jefes src 192.168.1.10 192.168.1.20
```

3) Esta regla, que se llama sistemas, hace referencia a un archivo *permitidos* que se encuentra en */etc/squid*, y contiene las IP del personal que trabaja en el area de sistemas.

```
acl sistemas src "/etc/squid/permitidos"
```

Regla Tipo dts

Especifica una dirección de destino en formato IP y mascara o el nombre de dominio del destino. Nomenclatura:

```
acl [Nombre] dts [Contenido]
```

Ejemplos:

1)En esta regla, llamada webmial, contendrá como destino final las direcciones de webmail más conocidos de internet.

```
acl webmail dst www.gmail.com www.hotmail.com www.yahoo.com
```

2)En esta regla llamada iplocales contendrá algunas de la IP de nuestro segmento de red.

```
acl iplocales dst 192.168.1.109 192.168.1.103
```

Regla Tipo srcdomain.

La regla de tipo srcdomain se usa para hacer referencia a dominios de origen y se determina por la resolución DNS inversa. Para poder usar esta regla es necesario contar con un DNS local. Nomenclatura:

```
acl [Nombre] srcdomain [Contenido]
```

Ejemplo:

La regla repos indica qué máquinas de nuestra red local están agregadas a la misma.

```
acl repos srcdomain repoubu.agora.es repodeb. agora.es repocen. agora.es
```

Regla Tipo dstdomain.

La regla de tipo dstdomain se utiliza para referirse a dominios de destino. Nomenclatura:

```
acl [Nombre] dstdomain [Contenido]
```

Ejemplo:

La regla permitidos indica un grupo de dominios que pueden estar en Internet .

```
acl pemitidos dstdomain .linuxparatodos.net .factor.com.mx .eluniversal.com
.reforma.com
```

Regla Tipo srcdom_regex.

Esta regla se encarga de evaluar expresiones regulares incluidas en el nombre de dominio de origen. Nomenclatura:

```
acl [Nombre] srcdom_regex [Contenido]
```

Ejemplo:

La regla intranet recoge todos los dominios locales con la palabra factor en mayúsculas y minúsculas.

```
acl intranet srcdom_regex -i factor
```

Regla Tipo dstdom_regex

Esta regla se encarga de evaluar expresiones regulares incluidas en el nombre de dominio de destino. Nomenclatura:

```
acl [Nombre] dstdom_regex [Contenido]
```

Ejemplo:

La regla google_todos recoge todos los dominios de destino que contienen la palabra google en mayúsculas o minúsculas.

```
acl google_todos dstdom_regex -i google
```

Regla Tipo time

Esta regla estable un rango de tiempo definido en los días de la semana. Parámetros por días de la semana:

Parámetros	Días
S	Domingo
M	Lunes
T	Martes
W	Miércoles
H	Jueves
F	Viernes
A	Sábado

Para definir el rango horario se utilizan 24:00 hrs indicando hora de origen y de final separadas por un guión. Nomenclatura:

```
acl [Nombre] time [días][horas]
```

Ejemplo:

La regla horario estable que está habilitada los días de Lunes a Viernes de 09:00 a 18:00 hrs.

```
acl horario time MTWHF 09:00-18:00
```

Regla Tipo url_regex

Permite especificar expresiones regulares para comprobar la url, para este tipo de regla se recomienda tener un archivo en cual agregamos todas la palabras que nosotros creamos que deban recoger la regla. Nomenclatura:

```
acl [Nombre] url_regex "Path"
```

Ejemplo de archivo descargas.txt:

```
descarga
download
softonic
megaupload
cine
peliculas
```

Ejemplo:

Esta regla se llama descargas el cual manda a llamar aun archivo que contiene palabras relacionadas a pornografía.

```
acl descargas url_regex "/etc/squid/listas/descargas.txt"
```

Regla Tipo urlpath_regex

Esta regla nos permite la administración de descargas recogiendo por ejemplo una serie de extensiones de archivos, se recomienda tener registrar todas las extensiones en un fichero de texto, haciendo referencia a él en la regla. Nomenclatura:

```
acl [Nombre] urlpath_regex "Path"
```

Ejemplo de archivo extensiones.txt:

```
\.avi$
\.mpg$
\.mpeg$
\.avi$
\.flv$
\.exe$
\.bat$
\.zip$
\.mp3$
```

Ejemplo:

Esta regla llamada extensiones administrará las descargas por medio de la extensiones de los archivos.

```
acl extensiones urlpath_regex "/etc/squid/listas/extensiones.txt"
```

Regla Tipo port

Esta regla nos permite comprobar el puerto de destino y así controlar el tipo de servicio al que se accede. Nomenclatura:

```
acl [Nombre] port [número-puerto]
```

Ejemplo:

Esta regla llamada SSL-port recoge el número de puerto para el servicio web seguro https.

```
acl SSL-port port 443
```

Otra regla llamada No-registrados recoge el rango de puertos no registrados, usados generalmente por la parte del cliente.

```
acl No-registrados port 1024-65535
```

Regla Tipo macaddress

Este tipo de regla nos permite administrar squid por medio de las direcciones MAC. Nomenclatura:

```
acl [Nombre] arp "Mac Address"
```

Ejemplo: Esta regla se llama adminmac en ella proporcionamos las direcciones Mac de las máquinas clientes .

```
acl adminmac arp 09:00:2b:23:45:67 00:1f:3c:5f:fd:b1 00:1e:ec:70:7e:24
```

Regla Tipo password

Este tipo de regla permite controlar el acceso a internet por medio de un usuario y password. Para poder habilitar este método tendremos que seguir lo siguientes pasos de configuración.

1) Creamos el archivo que contendrá las claves. Lo creamos oculto para mayor seguridad.

```
[root@mantis squid]# touch .claves
```

2) Le asignamos permisos de Lectura/Escritura y el usuario encargado del archivo.

```
[root@mantis squid]#  chmod 600 .claves
[root@mantis squid]#  chown squid:squid .claves
```

3)Creación de usuario y password para el acceso a internet.

```
[root@mantis squid]# htpasswd .claves clientes
```

4) Habilitaremos las siguientes opciones dentro del fichero de configuración del servidor squid, busquemos el primer parámetro llamado auth_param basic.

```
#auth_param basic program <uncomment and complete this line>
```

Este parámetro lo modificaremos de la siguiente manera.

```
auth_param basic program /usr/lib/squid/ncsa_auth /etc/squid/.claves
```

Podemos usar esta otra opción:

```
authenticate_program /usr/lib/squid/ncsa_auth /etc/squid/.claves
```

Estamos indicando la aplicación que controla la autentificación y donde se encuentra el archivo que contiene las cuentas de los usuarios.

5) Por ultimo tendremos que habilitar la regla acl encargada de la autenticación de password.

```
acl password proxy_auth REQUIRED
```

Con esto ya tendremos habilitada la regla para la autenticación de los usuarios.

Control de Acceso

Éstas definen si se permite o no el acceso. Se aplican a las *Listas de Control de Acceso,* se puede aplicar a una ACL o a varias en una sola orden. Si se incluyen varias reglas en una sola línea para permitir o denegar, el paquete debe cumplir todas las reglas referidas. Nomenclatura:

```
http_access allow/deny Regla
```

Las reglas pueden aparecer negadas precedidas por el carácter "!", en esos casos los paquetes que cumplen la regla serán los **no recogidos** en la misma.

Ejemplo:

```
acl all src 0.0.0.0/0.0.0.0
acl jefes src 192.168.1.23 192.168.1.25
http_access allow jefes
http_access deny all
```

Ejemplo:

```
acl all src 0.0.0.0/0.0.0.0
acl jefes src 192.168.1.23 192.168.1.25
acl red-local src 192.168.1.0/255.255.255.0
acl laboral time  MTWHF 09:00-15:00
acl no-permitidos url_regex "/etc/squid/listas/paginas.txt"
http_access allow jefes
http_access allow red-local laboral !no-permitidos
http_access deny all
```

Cuando al proxy le llega una petición, se comprueba en orden cada regla hasta que alguna sea aplicable y, cuando se aplique, ya no se miran más reglas. Por tanto, es importante **el orden** en el que se incluyen en el fichero de configuración las reglas para el control de acceso.

Ya que las reglas se comprueban en orden, es importante seguir el criterio de colocar primero las reglas más particulares, dejando para el final las más generales, ya que si una regla más general que engloba a una particular es colocada antes, nunca se llegará a comprobar la regla particular.

Es conveniente mantener un orden dentro del fichero de configuración, ya que todas las directivas deben aparecer en el mismo fichero, por ello, se debe respetar una estructura fácilmente comprensible para permitir una administración lo más fácil posible. Por eso, es conveniente dividir el fichero de configuración en varias secciones:

1. Directivas de configuración general.
2. Listas de Control de Acceso.
3. Órdenes http_access que gestionen el control de acceso.

Servidor proxy transparente

Configurar squid como proxy transparente permite que las conexiones sean enrutadas al proxy sin hacer ninguna configuración en los clientes para que tengan salida a internet. Los clientes se conectarán a Internet pasando por el proxy sin darse cuente de ello. Este tipo de configuración depende de reglas de nuestro firewall.

Parámetro http_port

Solamente tendremos que configurar este parámetro para que sea un proxy transparente. Se le debe indicar la IP del servidor squid, puerto de escucha y la palabra transparent. Cambiamos

```
http_port 3128
```

por

```
http_port 192.168.1.254:3128 transparent
```

No es imprescindible indicar la dirección IP, pero sí conveniente porque majora la seguridad.

Reglas del Firewall

Para poder configurar el proxy transparente, tendremos que configurar reglas de firewall, en nuestro caso usaremos reglas de iptables ya que es la herramienta más utilizada en todas las distribuciones GNU/Linux. Para que funcione de manera transparente debemos de aplicar la siguiente regla en iptables.

```
iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j REDIRECT --to-port
```

```
3128
```

Con esto ya tenemos redireccionadas todas las conexiones http (puerto 80) hacia el proxy (puerto 3128). Si pretendemos redireccionar a través del proxy otras conexiones como ftp, ssh, smtp, pop3, etc; debemos incluir las reglas iptables similares referidas a los puertos correspondientes a estos servicios.

Aclaraciones sobre squid transparente.

Hay aspectos de squid que no funcionan cuando se configura como proxy transparente:

1. Reglas del tipo password. No funcionan con el proxy transparente ya que para que funcione esta regla se precisa que haya un diálogo entre el proxy y el navegador que no es posible en este caso.
2. Conexiones usando el puerto seguro 443 (https). No se puede redireccionar el puerto seguro 443 (https) al proxy transparente ya que éste no puede interpretar su contenido y, en estos casos, corta siempre la conexión. Si se desea usar este puerto con el proxy habrá que configurar el navegador para conectarse directamente a través del proxy.

Combinando squid e iptables.

Cuando configuramos squid transparente redireccionando las conexiones a Internet por medio de iptables, debemos tener en cuenta que no es necesario activar el bit de redireccionamiento:

echo 1 > /proc/sys/net/ipv4/ip_forward

ya que los paquetes son reconstruidos por squid y enviados como si tuvieran su origen en el servidor proxy, poniendo como IP de origen su dirección pública. Por tanto, tampoco es necesario activar la conversión NAT, ya que es el propio squid quien se encarga de cambiar la dirección de origen y de mantener la tabla con las conexiones establecidas entre los clientes de la red local e Internet.

Por otra parte, se pueden combinar las dos opciones para dar acceso a Internet a la red local, es decir, permitir que determinadas conexiones se realicen a través del proxy y otras mediante conversión NAT aplicada directamente por iptables. Por ejemplo; se pueden redireccionar al proxy los servicios http, smtp y pop3, mientras que otros servicios como ssh y https se retransmiten directamente a través de iptables sin redireccionar al proxy.

El siguiente script para iptables nos permitiría hacer esto:

```
## Limpiar las reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

## Establecer politicas por defecto para las cadenas predefinidas
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT

## Activar el reenvio de paquetes
echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
## Reglas de enmascaramiento
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.10.0/24 -j MASQUERADE

## Redireccionamos al proxy el servicio WEB
iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j REDIRECT --to-port
3128


## Redireccionamos al proxy el servicio SMTP
iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 25 -j REDIRECT --to-port
3128

## Redireccionamos al proxy el servicio POP3
iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 110 -j REDIRECT --to-port
3128
```

Cadenas de iptables que atraviesan los paquetes redireccionados a squid.

Este es un aspecto importante a tener en cuenta cuando se combina squid con iptables. Hay que tener claro qué cadenas de iptables deben pasar los paquetes que se redireccionan a squid:

- FORWARD: los paquetes redireccionados a squid **no pasan** por esta cadena, ya que son redirigidos hacia el mismo sistema, y por esta cadena no pasan los paquetes que tienen su origen o van destinados al propio sistema.
- OUTPUT: los paquetes que son reenviados por squid **pasan** por esta cadena antes de ser enviados a la red ya que son paquetes generados dentro del sistema por el propio squid.

Proteger el ancho de banda

Con squid podemos establecer límites al ancho de banda a utilizar, bien, en conjunto, individual para cada usuario, distintos anchos de banda según los usuarios o según los destinos.

Parámetros de configuración:

Cada una de las reglas de regulación está definida por un número entero que vamos a utilizar para indentificarlo en los distintos parámetros.

delay_pools
Esta parámetro se utiliza para definir cuantas reglas de regulación vamos a definir.

delay_class
Este parámetro toma dos argumentos el primero el identificador de la regla y en segundo lugar el tipo (clase) de la regla.

El primero argumento es un número entero para identificar la regla.
El segundo argumento puede ser 1,2 ó 3 para indicar uno de estos tipos de reglas.

delay_parameters
establece los valores de la regla. Los argumentos de este parámetro son parejas de valores velocidad/tamaño, donde velociad es un número entero que indica una velocidad en bytes por segundo (B/s), y tamaño indica el número de bytes de reserva que se transmiten antes de aplicar la velocidad de transferencia.

Es decir cada pareja especifica el número de bytes de margen que se permiten antes de que se haga efectiva la restricción de velocidad.

Tipos de reglas
Hay tres clases de regulaciones, con características diferentes.

Clase 1
La clase uno consiste en un canal individual compartido por todos los usuarios. Es la clase más simple, todas las tasas de descarga van juntas y lo único que tenemos que especificar es la velocidad, en bytes por segundo, y el número de bytes a partir de los cuales tiene que retardar la descarga.

Ejemplo: Supongamos que queremos que la velocidad de descarga de los ficheros .wmv que ocupen más de 1Mb sea de 8k/s.

```
delay_pools 1
delay_class 1 1
delay_parameters 1 8192/1048576
acl ficheros_wmv url_regex \.wmv$
delay_access 1 allow ficheros_wmv
```

Cuando en esa red, algún equipo haya bajado más de 1Mb correspondiente a ficheros .wmv entonces la descarga total de ficheros .wmv se hará a una velocidad de 8k/s. Si quisiéramos establecer este límite para los hosts de una red deberíamos elegir la clase 2.

En el anterior ejemplo hemos supuesto que hay un único canal de regulación en el parámetro delay_pools. Ahora vamos a ver un ejemplo con dos canales de clase 1. Queremos que el tráfico de la red local no tenga límite de transferencia, mientras que para el resto de conexiones vamos a utilizar un total de 256Kbits/s.

```
delay_pools 2
delay_class 1 1
delay_parameters 1 -1/-1
acl red_local src 192.168.0.0/24
delay_access 1 allow red_local
delay_class 2 1
delay_parameters 2 32768/1024
acl resto src 0.0.0.0/0.0.0.0
delay_access 2 allow resto
```

La tasa de transferencia viene especificada en bytes por segundo, por ejemplo convirtiendo a otras unidades uno de los parámetros de delay_parameters:

32768 B/s = 32 KB/s = 32 x 8 Kb/s (bits) = 256 Kb/s = 256 Kbits/s

El valor 1024 es un número de bytes lo suficientemente pequeño para que se alcance pronto y se estabilice el ancho de banda.

El valor -1 indica ilimitado.

Las regulaciones de clase 1 están diseñadas para evitar que la acl correspondiente consuma todo el ancho de banda, es decir definen la suma máxima de los anchos de banda de todos los clientes, pero no evita que un cliente pueda consumir ese ancho de banda a costa de otro; todos os clientes comparten un máximo ancho de banda.

Clase 2
La clase dos que consiste en un canal común con 256 canales individuales. Es decir tenemos un canal global y dentro de él otros 256 canales que se aplican a las máquinas de una red de clase C.

Ahora vamos a ver otro ejemplo con dos canales, en este caso ambos de de clase 2. Ahora también que el tráfico de la red local no tenga límite de transferencia, ni global ni por IP, mientras que para el resto de conexiones a internet vamos a utilizar un total de 256Kbits/s y a cada máquina de la red le asigna un máximo de 64kbits/s.

delay_pools 2
delay_class 1 2
delay_parameters 1 -1/-1 -1/-1
acl red_local src 192.168.0.0/24
delay_access 1 allow red_local
delay_class 2 2
delay_parameters 2 32768/1024 8192/1024
acl resto src 0.0.0.0/0.0.0.0
delay_access 2 allow resto

Este ejemplo es muy parecido al anterior de clase 1, sólo tenemos que agregarle una pareja de valores al parámetro "delay_parameters". La primera pareja de este parámetro especifica el ancho de banda global, mientras que la segunda pareja especifica el ancho de banda por host.

Hay que tener en cuenta que los clientes están limitados por el tamaño del canal más pequeño, por lo que no tiene sentido que el canal común tenga un tamaño menor que los canales individuales.

Clase 3
La clase tres que define un canal común que contiene 256 canales de red, 65536 canales individuales. Muy parecida a la clase dos pero par redes de clase B.

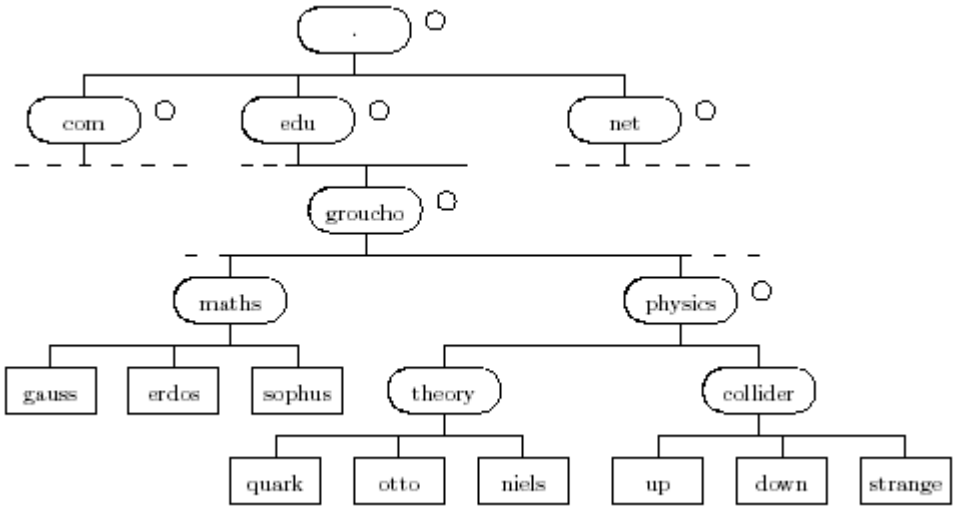
Hay que tener en cuenta que los clientes están limitados por el tamaño del canal más pequeño, por lo que no tiene sentido que el canal común tenga un tamaño menor que los canales individuales.

SERVICIO DE RESOLUCIÓN DE NOMBRES DNS

Cómo funciona el DNS

El DNS organiza los nombres de máquina (hostname) en una jerarquía de dominios. Un *dominio* es una colección de nodos relacionados de alguna forma—porque estén en la misma red, tal como los nodos de una universidad—. Por ejemplo, las universidades americanas se agrupan en el dominio edu. Cada universidad tiene allí un *subdominio*. Un nodo, por ejemplo erdos, tendrá un nombre completo (conocido como totalmente cualificado) tal como erdos.maths.groucho.edu. Este nombre totalmente cualificado también se conoce por las siglas FQDN.

En la figura vemos una parte del espacio de nombres. La raíz del árbol, que se identifica con un punto sencillo, es lo que se denomina *dominio raíz* y es el origen de todos los dominios. Para indicar que un nombre es FQDN, a veces se termina su escritura en un punto. Este punto significa que el último componente del nombre es el dominio raíz.



El sistema DNS hace más cosas. Permite delegar la *autoridad* de un subdominio a sus administradores. Por ejemplo, los responsables del Centro de Cálculo Groucho pueden crear un subdominio para cada departamento, y delegar su control a éstos. Así, cada departamento puede definir libremente todos los nodos que quiera dentro de su subdominio e incluso crear nuevos subdominios y delegarlos.

Para esto, el espacio de nombres se divide en *zonas*, cada una asignada a un dominio. Hay que ver la diferencia entre *zona* y *dominio*: por ejemplo, el dominio groucho.edu incluye todas las máquinas y subdominios de éste. Mientras que la zona groucho.edu solo incluye las máquinas del dominio, no los subdominios delegados. Es decir, los nodos del subdominio physics.groucho.edu pertenecen a una zona diferente. En la Figura, el inicio de la zona se marca con un pequeño círculo a la derecha del nombre de dominio.

Búsquedas con DNS

Cuando nuestra aplicación quiera buscar la información de erdos, contactará con un servidor de nombres local, quien lleva a cabo una secuencia de peticiones. En primer lugar pregunta al servidor de nombres raíz, preguntando por erdos.maths.groucho.edu. El servidor raíz reconoce que el nombre no pertenece a ninguna de sus zonas *de autoridad* pero sí sabe qué hacer con la zona *edu*. Esto es, devuelve a nuestro servidor más información sobre los servidores de nombres que pueden servir la zona edu. Ahora nuestro servidor preguntará por este nombre a uno de esos servidores. Ellos nos enviarán a uno que tenga información autorizada del dominio groucho.edu. Ahora nuestro servidor interrogará a éste y finalmente obtendrá la dirección de erdos.

Para acelerar futuras peticiones de nombres, el servidor almacena la información obtenida en la búsqueda anterior en su *cache* local. Así, la próxima vez que busquemos algún nodo de groucho.edu, ya no habrá que ir a los servidores raíz o los de la zona edu. El servidor de nombres no almacenará para siempre la información en la cache; la limpiará cada cierto tiempo. El tiempo de vida se llama *TTL*.

Tipos de servidores de nombres

Los servidores de nombres que mantienen oficialmente la información de una zona se conocen como *autorizados* de la zona, y a veces se conocen como *servidores principales o maestros*. Cualquier petición de nodos de esa zona irá a parar a uno de estos servidores principales.

Los servidores principales deben estar bien sincronizados. Es decir, uno de ellos será llamado *primario*, que carga su información de un fichero, y hacer a los demás *secundarios*, que obtienen su información pidiéndosela periódicamente al primario.

El objetivo de tener varios servidores principales es distribuir la carga y dar cierta tolerancia a fallos. Cuando uno de los servidores principales falla, todas las peticiones acabarán en los demás. Por supuesto, este esquema no nos protege de fallos del servidor que produzcan errores en todas las peticiones DNS, como podrían ser errores del software.

También podemos instalar un servidor de nombres que no es maestro de ninguna zona. Esto es útil, para dar servicio de nombres a una red local aprovechando sus características de ahorro de ancho de banda gracias a su cache. Estos servidores se conocen como de *solo-cache*.

Resolución inversa

La operación más habitual con el DNS es obtener la dirección IP correspondiente a un nombre de nodo. Sin embargo, a veces queremos hacer la operación opuesta: encontrar el nombre a partir de la dirección IP. Esto se conoce como *resolcuión inversa*, y la usan diversas aplicaciones para comprobación de identidad del cliente. Cuando se utiliza el fichero hosts, la resolución se realiza mediante una búsqueda simple en el fichero. Con el DNS, una búsqueda exhaustiva en el espacio de nombres carece de sentido. En su lugar, existe un dominio especial, el in-addr.arpa, que contiene las

direcciones IP de todos los sistemas en una notación de puntos invertida.

En cada zona de autoridad habrá que definir un dominio para la resolución inversa, así por ejemplo el dominio 8.76.149.in-addr.arpa, recogerá información de las direcciones 149.76.8.X.

Resolución de nombres en Linux.

La red TCP/IP puede utilizar diferentes métodos para convertir nombres en direcciones IP. El mecanismo más simple consiste en almacenar los nombres en una tabla de máquinas en el fichero /etc/hosts. Esto es únicamente interesante en el caso de pequeñas redes de área local que sólo requieran la administración de una persona, y que no tengan tráfico IP con el mundo exterior.

Alternativamente, puede utilizarse BIND el *servicio de nombres Internet de Berkeley* o "Berkeley Internet Name Domain", para traducir nombres de máquinas a direcciones IP (cosa que también se conoce como *resolución*). Configurar BIND puede ser una laboriosa tarea pero, una vez hecho, los cambios en la topología de la red serán mucho más fáciles de hacer. En Linux, como en muchos otros sistemas Unix, el servicio de nombres se realiza mediante un programa llamado **named**. Al iniciarse, carga un conjunto de ficheros maestros en su *cache* y espera peticiones de procesos locales o remotos. Existen distintas maneras de preparar BIND, y no es necesario ejecutar un servidor de nombres en cada máquina: generalmente, uno para toda la red es suficiente.

La biblioteca de resolución

Cuando hablamos del *sistema de resolución*, no nos referiremos a una aplicación en particular, sino a la *biblioteca de resolución*: un conjunto de funciones que pueden encontrarse en las bibliotecas estándar del lenguaje C. Las rutinas principales son `gethostbyname(2)` y `gethostbyaddr(2)`, que buscan la dirección IP de una máquina a partir del nombre y viceversa. Es posible configurarlas para que simplemente miren en el fichero hosts local (o remoto, si se usa NIS).

Las funciones del sistema de resolución leen ficheros de configuración cuando son llamadas. Desde estos ficheros, determinan qué bases de datos hay que interrogar, en qué orden y otros detalles relevantes. En la antigua biblioteca libc de Linux, se utilizaba el fichero /etc/host.conf como fichero maestro, pero en la versión 2 de las bibliotecas, la glibc, se utiliza el fichero /etc/nsswitch.conf. Vamos a describir ambas formas, puesto que son muy usuales.

Ejemplo de fichero host.conf

```
# /etc/host.conf
# Tenemos servidor de nombres, pero no NIS
order bind hosts
# Permitir multiples direcciones
multi on
# Contra los nombres falsos
nospoof on
```

Ejemplo de fichero nsswitch.conf

```
# /etc/nsswitch.conf
#
# Ejemplo de configuracion del nsswitch de GNU.
# En el paquete 'libc6-doc' se documentan estos ficheros.
hosts: dns files
networks: files
```

Configuración del fichero resolv.conf

Cuando se configura la librería de resolución para utilizar los servicios de BIND, tiene que indicarse también qué servidores utilizar. El fichero resolv.conf contiene una lista de servidores, que si está vacía hará considerar al sistema que el servidor está en su máquina.

Ejemplo de fichero resolv.conf:

```
# /etc/resolv.conf
# Nuestro dominio
domain vbrew.com
#
# Nuestro servidor principal va a ser vlager:
nameserver 172.16.1.1
```

Configuración de BIND9

named es el servidor DNS en casi todas las máquinas Unix.

Se instala con:

```
apt-get install bind9
```

named suele iniciarse al arrancar la máquina, y ejecutarse hasta que se apaga. Las versiones anteriores de BIND hasta la 8 obtienen la información que necesitan de un fichero llamado /etc/named.boot. Las nuevas versiones usan el fichero /etc/bind/named.conf. Además, hay que configurar los ficheros de zona. En las versiones más actuales de BIND9 la configuración de named.conf se divide en varios ficheros y en el fichero named.conf solo aparece la referencia a ellos con la orden include.

Para ejecutar **named**, solo tiene que teclear:

```
# /etc/init.d/bind9 start
```

Durante la instalación normalmente se crea un script que ejecuta **named** en el proceso de arranque de la máquina.

El programa **named** se iniciará y leerá el fichero **named.conf** y los ficheros de zona que se especifiquen en él.

El fichero named.conf de BIND 9

Veamos un ejemplo de este fichero para comprender su sintaxis y las opciones principales. No las estudiaremos todas, pues son muchas, sólo las imprescindibles para el correcto funcionamiento de named.

```
// /etc/named.boot para vlager.vbrew.com
options {
       directory "/var/named";
};

zone "." {
       type hint;
       file "named.ca";
};

zone "vbrew.com" {
       type master;
       file "named.hosts";
};

zone "0.0.127.in-addr.arpa" {
       type master;
       file "named.local";
};

zone "16.172.in-addr.arpa" {
       type master;
       file "named.rev";
};

// servidor esclavo
zone "vin.com" {
       type slave;
       masters { 192.168.3.15 };
};

// servidor de otro dominio (forward)
```

```
zone "com" {
       type forward;
       forwarders { 192.168.3.1; 140.82.34.2 };
};
```

Si observamos el ejemplo, veremos que la información se agrupa en bloques en estilo C, encerrado entre llaves (signos { y }).

Los comentarios del fichero se escriben en notación similar a C++, es decir, dos barras (signo //).

El bloque **options** recoge las opciones globales de configuración. La sentencia **directory** va ahora dentro de ella, e indica el directorio por defecto en el que se buscan los ficheros de bases de datos.

Cada zona comienza con la palabra zone, seguido del nombre del dominio. Dentro del bloque se especifica:

type:

indica el tipo para ese dominio, que puede ser:

- hint, para el cache, recogido en la zona del dominio raiz.
- master, para maestro o primario.
- slave, para secundario o esclavo.
- forward, para indicar direcciones de otros servidores a explorar para el dominio especificado.

file:

Indica el fichero de base de datos donde se almacena la información de esa zona.

masters:

Indica la dirección del servidor DNS primario.

forwarders:

Indica las direcciones de los servidores DNS a explorar para ese dominio.

Ficheros de base de datos DNS

Los ficheros incluidos con **named**, como **named.hosts**, siempre tienen un dominio asociado a ellos

llamado *origen*. Este es el nombre de dominio especificado con los comandos zone con type master. En un fichero maestro, se pueden especificar nombres de máquinas y dominios relativos a este dominio. Un nombre dado en un fichero de configuración se considera absoluto si termina con un punto. En caso contrario se considera relativo al origen. Al origen en sí mismo nos podemos referir con «@».

Todos los datos en un fichero principal se dividen en *registros de recursos* o RRs. Son la unidad de información del DNS. Cada RR tiene un tipo. Los registros de tipo A, por ejemplo, asocian un nombre a una dirección IP. Los registros de tipo CNAME asocian un alias de una máquina con su nombre oficial. Como ejemplo, obsérvese Ejemplo 6-11, que muestra el fichero named.hosts para nuestro sistema.

La representación de los RRs en los ficheros utiliza el siguiente formato:

[*domain*] [*ttl*] [*class*] *type rdata*

Los campos se separan por espacios o tabulaciones. Una entrada puede continuarse en varias líneas si se abre un paréntesis antes del primer fin de línea y el último campo es seguido de un cierre de paréntesis. Cualquier cosa entre un punto y coma y el siguiente salto de línea será un comentario.

domain

Aquí va el nombre del dominio que se aplica al RR actual. Si no se da nombre de dominio, se asume el mismo que se puso para el RR anterior.

ttl

Con el fin de forzar al sistema DNS a descartar información después de cierto tiempo, cada RR lleva asociado un *tiempo de vida* o *ttl*. El campo *ttl* especifica, en segundos, el tiempo de validez de la información desde que se obtiene del servidor. Es un número decimal de hasta ocho dígitos.

Si no se especifica ningún valor, tomará uno por defecto del campo *minimum* del registro SOA precedente.

class

Aquí se indica la clase de dirección: IN para direcciones IP, HS para objetos de la clase Hesiod.

Trabajando con redes TCP/IP debe usarse siempre la clase IN.

Si no se especifica ningún valor, se toma el valor del RR anterior.

type

Describe el tipo de RR. Los tipos habituales son A, SOA, PTR, MX, CNAME y NS. En las siguientes secciones comentaremos estos tipos de RRs.

rdata

Contiene los datos asociados al RR. El formato depende del tipo, y se describirán más adelante. A continuación se presenta una lista incompleta de RRs que se utilizan en los ficheros de DNS. Hay algunos más que no vamos a comentar. Son experimentales, y de escaso uso.

Tipos de registros de recurso:

SOA

Describe una zona de autoridad (SOA significa «Start of Authority», es decir, «Comienzo de Autoridad»). Señala que los registros siguientes contienen información «autorizada» para el dominio.

Cada fichero incluido en la opción type master debe tener un registro SOA para esta zona. Los datos asociados contienen los siguientes campos:

origin

Nombre canónico del servidor de nombres primario para este dominio. Se suele dar como nombre absoluto.

contact

Dirección de correo electrónico de la persona responsable de mantener el dominio, reemplazando el carácter «@» por un punto. Por ejemplo, si el responsable de nuestra red fuese janet, este campo contendrá: janet.vbrew.com.

serial

Este es el número de versión del fichero de zona, expresado con un número decimal. Cuando se cambien datos del fichero, deberá incrementarse este número. Se suele expresar como número de versión en el día actual, es decir, en el formato AAAAMMDDnn siendo AAAA el año, MM el mes, DD el día y nn el número de revisión de ese día (01 si no hay más de una). Por ejemplo, 2001072201 para el 22 de julio de 2001.

El número de versión es utilizado por los servidores secundarios para saber cuándo la información de una zona ha cambiado. Para mantenerse actualizados, los servidores secundarios piden cada cierto tiempo el registro SOA del primario, y comparan el número de versión con el que tienen en la *cache*. Si ha cambiado, el servidor secundario pedirá de nuevo la información de zona al primario.

refresh

Especifica el intervalo, en segundos, que esperan los servidores secundarios entre peticiones de registros SOA a los primarios. De nuevo, se trata de un número decimal de hasta ocho dígitos. Normalmente, la topología de la red no cambia mucho, con lo que este número será como poco de un día para grandes redes, y de mucho más tiempo para redes pequeñas.

retry

Este número determina los intervalos de tiempo entre reintentos de comunicación con servidores primarios cuando una petición de una zona falla. No debe ser pequeño ya que un fallo temporal del servidor primario hará que el secundario cargue inútilmente la red. Buenas elecciones son una hora o como poco media hora.

expire

Especifica el tiempo, en segundos, que tardará el servidor en descartar los datos de zona si no ha podido contactar con el servidor primario. Normalmente se pondrá un valor grande, de por lo menos una semana (604800 segundos), aunque si se incrementa a un mes o más será incluso más razonable.

minimum

Valor por defecto para el valor del *ttl* en los registros de recursos que no lo especifiquen. Sirve para indicar a otros servidores de nombres que descarten el RR tras cierto tiempo. No tiene efecto, sin embargo, sobre el tiempo en el que un servidor secundario intenta actualizar la información de zona.

El valor de *minimum* debe ser grande, en especial para redes locales con topologías poco cambiantes. Una buena elección puede ser de una semana o un mes. En el caso de que haya registros RR que cambien con frecuencia, siempre podrá asignarle valores particulares de *ttl*.

A

Asocia direcciones IP con nombres. El campo de datos contiene la dirección separando los octetos por puntos, como es habitual.

Para cada máquina sólo puede haber un registro A, que se considera nombre oficial o *canónico*. Cualquier otro nombre será un alias y debe ser incluido con registros CNAME.

NS

Apunta a un servidor de nombres maestro de una zona subordinada. El campo de datos contiene el nombre del servidor. Para traducir ese nombre debe proporcionarse un registro A adicional, que se

conoce como *glue*, al proporcionar la dirección IP del servidor.

Hay que incluir registros NS en dos casos: primero, cuando delegamos la autoridad a una zona subordinada. Segundo, en la base de datos del servidor principal de cualquier zona. Los servidores NS especificados en el fichero de zona deben coincidir exactamente con los que especifica la zona padre que delega.

El registro NS especifica el nombre del servidor de nombres primario y los secundarios para una zona. Estos nombres deben poderse resolver para poderse usar. A veces los servidores pertenecen al mismo dominio que sirven, lo que ocasiona un problema de *el huevo o la gallina*: no podemos obtener el servidor de nombres hasta que accedamos al dominio, pero para acceder el dominio hay que conocer la IP del servidor de nombres... Para resolver este problema se crean registros A direectamente en la zona padre, para resolver esas direcciones. Estos son los que ya comentamos antes, los *registros glue* (podríamos traducirlos como registros-pegamento), puesto que unen o *pegan* la zona hija a la zona padre.

CNAME

Asocia un alias con su *nombre canónico*. El nombre canónico se determina con un registro A. Los alias son indicados mediante registros CNAME.

PTR

Se usa para asociar nombres del dominio in-addr.arpa con sus nombres normales. Se usa para obtener nombres a partir de direcciones IP (traducción inversa). El nombre de la máquina debe ser el canónico.

MX

Especifica el *servidor de correo* para un dominio. La sintáxis del registro MX es:

`[domain] [ttl] [class] MX preference host`

host nombra el servidor de correo para el dominio *domain*. Cada servidor tiene asociado un valor de *preference* (preferencia). Cuando un agente de transferencia de mensajes quiere entregar correo al dominio, intentará conectarse a esos servidores hasta conseguir entregar el mensaje; empezando por el que tenga menor valor de preferencia.

HINFO

Este registro da información sobre el hardware y el software de la máquina. Su sintaxis es:

`[domain] [ttl] [class] HINFO hardware software`

El campo *hardware* identifica el hardware usado en este nodo. Para indicarlo, se siguen ciertas convenciones, especificadas en el RFC 1700. Si el campo contiene blancos, debe encerrarse entre comillas dobles. El campo *software* indica el sistema operativo que corre el nodo, que también está normalizado.

Por ejemplo, un registro `HINFO` para describir un sistema Intel corriendo Linux podría ser:

```
tao 36500 IN HINFO IBM-PC LINUX2.2
```

y para el caso de que se tratara de un sistema basado en Motorola 68000:

```
cevad 36500 IN HINFO ATARI-104ST LINUX2.0
jedd 36500 IN HINFO AMIGA-3000 LINUX2.0
```

Cómo hacer los ficheros maestros

Ejemplo 6-10, Ejemplo 6-11, Ejemplo 6-12, y Ejemplo 6-13 muestran ficheros de ejemplo para un servidor de nombres de la cervecera virtual, localizada en vlager. Debido a la naturaleza de la red propuesta (una simple LAN), el ejemplo es muy simple también.

El fichero de cache named.ca mostrado en Ejemplo 6-10 contiene ejemplos de registros de servidores raíz. Un fichero típico de cache contiene como una docena de servidores de esta clase. Se puede obtener una lista de los servidores raíz usando la utilidad **nslookup** mostrada en la siguiente sección.

Ejemplo 6-10. El fichero named.ca

```
;
; /var/named/named.ca Fichero de cache para la cervecera.
; Al no estar en Internet no necesitamos servidores
; raiz. Si no fuera asi, descomentense.
;
;. 3600000 IN NS A.ROOT-SERVERS.NET.
;A.ROOT-SERVERS.NET. 3600000 A 198.41.0.4
;. 3600000 NS B.ROOT-SERVERS.NET.
;B.ROOT-SERVERS.NET. 3600000 A 128.9.0.107
;. 3600000 NS C.ROOT-SERVERS.NET.
;C.ROOT-SERVERS.NET. 3600000 A 192.33.4.12
;. 3600000 NS D.ROOT-SERVERS.NET.
;D.ROOT-SERVERS.NET. 3600000 A 128.8.10.90
;. 3600000 NS E.ROOT-SERVERS.NET.
;E.ROOT-SERVERS.NET. 3600000 A 192.203.230.10
;. 3600000 NS F.ROOT-SERVERS.NET.
;F.ROOT-SERVERS.NET. 3600000 A 192.5.5.241
```

```
;. 3600000 NS G.ROOT-SERVERS.NET.
;G.ROOT-SERVERS.NET. 3600000 A 192.112.36.4
;. 3600000 NS H.ROOT-SERVERS.NET.
;H.ROOT-SERVERS.NET. 3600000 A 128.63.2.53
;. 3600000 NS I.ROOT-SERVERS.NET.
;I.ROOT-SERVERS.NET. 3600000 A 192.36.148.17
;. 3600000 NS J.ROOT-SERVERS.NET.
;J.ROOT-SERVERS.NET. 3600000 A 198.41.0.10
;. 3600000 NS K.ROOT-SERVERS.NET.
;K.ROOT-SERVERS.NET. 3600000 A 193.0.14.129
;. 3600000 NS L.ROOT-SERVERS.NET.
;L.ROOT-SERVERS.NET. 3600000 A 198.32.64.12
;. 3600000 NS M.ROOT-SERVERS.NET.
;M.ROOT-SERVERS.NET. 3600000 A 202.12.27.33
;
```

Ejemplo 6-11. Fichero named.hosts

```
;
; /var/named/named.hosts Nodos de la cervecera
; El origen es vbrew.com
;
@      IN SOA vlager.vbrew.com. janet.vbrew.com. (
             2000012601 ; serie
             86400 ; refresco: uno al dia
             3600 ; reintento: una hora
             3600000 ; caducidad: 42 dias
             604800 ; minimo: 1 semana
             )
       IN NS vlager.vbrew.com.
;
; Correo local se entrega a vlager
       IN MX 10 vlager
;
; direccion 'loopback'
localhost.   IN A 127.0.0.1
;
; Delegar la autoridad de zona para el subdominio bourbon
bourbon             IN NS serv.bourbon.vbrew.com.
serv.bourbon IN A 172.16.3.1

; La ethernet de la cervecera virtual
```

```
vlager              IN A 172.16.1.1
vlager-if1   IN CNAME vlager
; vlager es tambien servidor de noticias
news                IN CNAME vlager
vstout              IN A 172.16.1.2
vale                IN A 172.16.1.3
;
; Ethernet de la vinatera virtual
vlager-if2   IN A 172.16.2.1
vbardolino   IN A 172.16.2.2
vchianti            IN A 172.16.2.3
vbeaujolais IN A 172.16.2.4
```

Ejemplo 6-12. Fichero named.local

```
;
; /var/named/named.local Resolucion inversa de 127.0.0
; El origen es 0.0.127.in-addr.arpa.
;
@      IN SOA vlager.vbrew.com. joe.vbrew.com. (
             1 ; serie
             360000 ; refresco: 100 horas
             3600 ; reintento: una hora
             3600000 ; caducidad: 42 dias
             360000 ; minimo: 100 horas
             )
       IN NS vlager.vbrew.com.
1      IN PTR localhost.
```

Ejemplo 6-13. Fichero named.rev

```
;
; /var/named/named.rev Resolucion inversa de nuestras IPs
; El origen es 16.172.in-addr.arpa.
;
@      IN SOA vlager.vbrew.com. joe.vbrew.com. (
             16 ; serie
             86400 ; refresco: una vez diaria
             3600 ; reintento: una hora
             3600000 ; caducidad: 42 dias
             604800 ; minimo: 1 semana
             )
```

```
       IN NS vlager.vbrew.com.
; cervecera
1.1    IN PTR vlager.vbrew.com.
2.1    IN PTR vstout.vbrew.com.
3.1    IN PTR vale.vbrew.com.
; vinatera
1.2    IN PTR vlager-if2.vbrew.com.
2.2    IN PTR vbardolino.vbrew.com.
3.2    IN PTR vchianti.vbrew.com.
4.2    IN PTR vbeaujolais.vbrew.com.
```

Cómo verificar la configuración

nslookup es una estupenda utilidad para comprobar el funcionamiento de un servidor de nombres. Se puede usar interactivamente o pasándole la pregunta por la línea de comandos. En este último caso podemos invocar el comando así:

```
$ nslookup nombre-de-host
```

nslookup envía sus peticiones al servidor citado en resolv.conf. Si este fichero tiene más de un servidor, **nslookup** eligirá uno al azar.

El modo interactivo es mucho más interesante. No solo sirve para buscar la IP de un nodo, sino que también podemos interrogar acerca de cualquier tipo de registro DNS y transferirnos toda la información de una zona si queremos.

Si se invoca sin argumentos, **nslookup** muestra el nombre del servidor elegido y entra en modo interactivo. En el prompt > podemos escribir cualquier nombre de dominio. Al principio preguntará solo por registros A, es decir, obtención de la IP asociada.

Podemos elegir un tipo de registro diferente con el comando:

```
> set type=tipo
```

donde *tipo* es uno de los tipos de RR descritos antes, o ANY.

Veamos una posible sesión de **nslookup**:

```
$ nslookup
Default Server: tao.linux.org.au
Address: 203.41.101.121
> metalab.unc.edu
Server: tao.linux.org.au
Address: 203.41.101.121
```

```
Name: metalab.unc.edu
Address: 152.2.254.81
>
```

La salida muestra el servidor DNS interrogado y el resultado obtenido.

Si preguntamos por algo que no tiene IP asociada pero sí otros registros de otra clase, el programa nos devolverá una advertencia del tipo `No type A records found`. Sin embargo, podemos usar el citado comando **set type** para buscar registros de otras clases. Por ejemplo, el registro SOA de un dominio puede ser pedido así:

```
> unc.edu
Server: tao.linux.org.au
Address: 203.41.101.121
*** No address (A) records available for unc.edu
> set type=SOA
> unc.edu
Server: tao.linux.org.au
Address: 203.41.101.121
unc.edu
       origin = ns.unc.edu
       mail addr = host-reg.ns.unc.edu
       serial = 1998111011
       refresh = 14400 (4H)
       retry = 3600 (1H)
       expire = 1209600 (2W)
       minimum ttl = 86400 (1D)
unc.edu name server = ns2.unc.edu
unc.edu name server = ncnoc.ncren.net
unc.edu name server = ns.unc.edu
ns2.unc.edu internet address = 152.2.253.100
ncnoc.ncren.net internet address = 192.101.21.1
ncnoc.ncren.net internet address = 128.109.193.1
ns.unc.edu internet address = 152.2.21.1
```

Con el tipo ANY obtendremos todos los registros existentes asociados al nombre dado.

Una aplicación práctica de **nslookup**, para depurar un servidor, es obtener la lista de servidores raíz. Para ello no hay más que pedir los NS del registro raíz (.):

```
> set type=NS
> .
```

```
Server: tao.linux.org.au
Address: 203.41.101.121
Non-authoritative answer:
(root) name server = A.ROOT-SERVERS.NET
(root) name server = H.ROOT-SERVERS.NET
(root) name server = B.ROOT-SERVERS.NET
(root) name server = C.ROOT-SERVERS.NET
(root) name server = D.ROOT-SERVERS.NET
(root) name server = E.ROOT-SERVERS.NET
(root) name server = I.ROOT-SERVERS.NET
(root) name server = F.ROOT-SERVERS.NET
(root) name server = G.ROOT-SERVERS.NET
(root) name server = J.ROOT-SERVERS.NET
(root) name server = K.ROOT-SERVERS.NET
(root) name server = L.ROOT-SERVERS.NET
(root) name server = M.ROOT-SERVERS.NET
Authoritative answers can be found from:
A.ROOT-SERVERS.NET internet address = 198.41.0.4
H.ROOT-SERVERS.NET internet address = 128.63.2.53
B.ROOT-SERVERS.NET internet address = 128.9.0.107
C.ROOT-SERVERS.NET internet address = 192.33.4.12
D.ROOT-SERVERS.NET internet address = 128.8.10.90
E.ROOT-SERVERS.NET internet address = 192.203.230.10
I.ROOT-SERVERS.NET internet address = 192.36.148.17
F.ROOT-SERVERS.NET internet address = 192.5.5.241
G.ROOT-SERVERS.NET internet address = 192.112.36.4
J.ROOT-SERVERS.NET internet address = 198.41.0.10
K.ROOT-SERVERS.NET internet address = 193.0.14.129
L.ROOT-SERVERS.NET internet address = 198.32.64.12
M.ROOT-SERVERS.NET internet address = 202.12.27.33
```

Para ver el conjunto completo de comandos, podemos usar **help** dentro de **nslookup**.

Declaraciones de adicionales:

Declaración `acl`

La sentencia `acl` (o sentencia de control de acceso) define grupos de hosts a los que se les puede permitir o negar el acceso al servidor de nombres.

Una declaración `acl` tiene la siguiente forma:

```
acl <acl-name> {
```

```
    <match-element>;

    [<match-element>; ...]

};
```

En esta declaración, sustituya *<acl-name>* con el nombre de la lista de control de acceso y reemplace *<match-element>* con una lista de direcciones IP separada por puntos y comas. La mayoría de las veces, una dirección IP individual o notación de red IP (tal como `10.0.1.0/24`) es usada para identificar las direcciones IP dentro de la declaración `acl`.

La siguiente lista de control de acceso ya están definidas como palabras claves para simplificar la configuración:

- `any` — Hace coincidir todas las direcciones IP.
- `localhost` — Hace coincidir cualquier dirección IP que se use el sistema local.
- `localnets` — Hace coincidir cualquier dirección IP en cualquier red en la que el sistema local está conectado.
- `none` — No concuerda ninguna dirección IP.

Cuando lo utilice con otras pautas (tales como declaraciones `options`), las declaraciones `acl` pueden ser muy útiles al asegurar el uso correcto de su servidor de nombres BIND.

El ejemplo siguiente define dos listas de control de acceso y utiliza una declaración `options` para definir cómo son tratadas en el servidor de nombres:

```
 acl black-hats {

    10.0.2.0/24;

    192.168.0.0/24;

 };

acl red-hats {

    10.0.1.0/24;

 };

options {

    blackhole { black-hats; };

    allow-query { red-hats; };

    allow-recursion { red-hats; };

 }
```

```
```

Este ejemplo contiene dos listas de control de acceso, `black-hats` y `red-hats`. Los hosts en la lista `black-hats` se les niega el acceso al servidor de nombres, mientras que a los hosts en la lista `red-hats` se les dá acceso normal.

Declaración `include`

La declaración `include` permite incluir archivos en un `named.conf`. De esta forma los datos de configuración confidenciales (tales como `llaves`) se pueden colocar en un archivo separado con permisos restringidos.

Una declaración `include` tiene la forma siguiente:

```
include  "<file-name>"
```

En esta declaración, *`<file-name>`* es reemplazado con una ruta absoluta a un archivo.

Declaración `options`

La declaracion `options` define opciones de configuración de servidor globales y configura otras declaraciones por defecto. Puede ser usado para especificar la ubicación del directorio de trabajo `named`, los tipos de consulta permitidos y mucho más.

La declaración `options` toma la forma siguiente:

```
options {

        <option>;

        [<option>; ...]

};
```

En esta declaración, las directivas *`<option>`* son reemplazadas con una opción válida.

Las siguientes son opciones usadas a menudo:

- `allow-query` — Especifica qué hosts tienen permitido consultar este servidor de nombres. Por defecto, todos los hosts tienen derecho a consultar. Una lista de control de acceso, o una colección de direcciones IP o redes se puede usar aquí para sólo permitir a hosts particulares hacer consultas al servidor de nombres.
- `allow-recursion` — Parecida a la opción `allow-query`, salvo que se aplica a las peticiones recursivas. Por defecto, todos los hosts están autorizados a presentar peticiones recursivas en un servidor de nombres.
- `blackhole` — Especifica que hosts no tienen permitido consultar al servidor de nombres.
- `directory` — Especifica el directorio de trabajo `named` si es diferente del valor predeterminado `/var/named`.
- `forward` — Especifica el comportamiento de reenvío de una directiva `forwarders`.

  Se aceptan las siguientes opciones:

  - `first` — Indica que los servidores de nombres especificados en la directiva `forwarders` sean consultados antes de que `named` intente resolver el nombre el mismo.
  - `only` — Especifica que `named` no intente la resolución de nombres él mismo en el evento de que fallen las consultas a los servidores de nombres especificados en la directriz `forwarders`.

- `forwarders` — Especifica una lista de direcciones IP válidas para los servidores de nombres donde las peticiones se pueden reenviar para ser resueltas.
- `listen-on` — Especifica la interfaz de red en la cual `named` escucha por solicitudes. Por defecto, todas las interfaces son usadas.

  Usando esta directiva en un servidor DNS que también actúa como una gateway, BIND se puede configurar para sólo contestar solicitudes que se originan desde algunas de las redes.

  Una directiva `listen-on` se parece al ejemplo siguiente:

```
options {

   listen-on { 10.0.1.1; };

};
```

  En este ejemplo, solamente son aceptadas las peticiones que llegan desde la interfaz de red sirviendo a la red privada (`10.0.1.1`).

- `notify` — Controla si `named` notifica a los servidores esclavos cuando una zona es actualizada. Acepta las opciones siguientes:
  - `yes` — Notifica a los servidores esclavos.
  - `no` — No notifica a los servidores esclavos.
  - `explicit` — Solamente notifica a los servidores esclavos especificados en una lista de `also-notify` dentro de la declaración de una zona.
- `pid-file` — Especifica la ubicación del archivo del proceso ID creado por `named`.
- `root-delegation-only` — Activa la implementación de las propiedades de delegación en dominios de nivel superior (TLDs) y zonas raíz con una lista opcional de exclusión. La *delegación* es el proceso de separar una zona sencilla en múltiples zonas. Para poder crear una zona delegada, se utilizan elementos conocidos como *registros NS*. Los registros de servidor de nombres (registros de delegación) anuncian los servidores de nombres autorizados para una zona particular.

  El siguiente ejemplo de `root-delegation-only` especifica una lista excluyente de los TDLs desde los que se esperan respuestas no delegadas:

```
options {

   root-delegation-only exclude { "ad"; "ar"; "biz"; "cr"; "cu"; "de"; "dm"; "id;

                              "lu"; "lv"; "md"; "ms"; "museum"; "name"; "no"; "pa";
```

```
                                "pf"; "se"; "sr"; "to"; "tw"; "us"; "uy"; };

};
```

- `statistics-file` — Permite especificar la localización alternativa de los archivos de estadísticas. Por defecto, las estadísticas de `named` son guardadas al archivo `/var/named/named.stats`.

Declaraciones adicionales en zone:

Las opciones más comunes para la declaración `zone` incluyen lo siguiente:

- `allow-query` — Especifica los clientes que se autorizan para pedir información sobre una zona. Por defecto, todas las peticiones de información son autorizadas.
- `allow-transfer` — Especifica los servidores esclavos que están autorizados para pedir una transferencia de información de la zona. Por defecto, todas las peticiones se autorizan.
- `allow-update` — Especifica los hosts que están autorizados para actualizar dinámicamente la información en sus zonas. Por defecto, no se autoriza la actualización de la información dinámicamente.

  Tenga cuidado cuando autorice a los hosts para actualizar la información de su zona. No habilite esta opción si no tiene confianza en el host que vaya a usar. Es mejor que el administrador actualice manualmente los registros de zona y que vuelva a cargar el servicio `named`.
- `file` — Especifica el nombre del archivo en el directorio de trabajo `named` que contiene los datos de configuración de zona.
- `masters` — Especifica las direcciones IP desde las cuales solicitar información autorizada. Solamente se usa si la zona está definida como `type slave`.
- `notify` — Controla si `named` notifica a los servidores esclavos cuando una zona es actualizada. Esta directiva sólo acepta las opciones siguientes:
  - `yes` — Notifica a los servidores esclavos.
  - `no` — No notifica a los servidores esclavos.
  - `explicit` — Solamente notifica a los servidores esclavos especificados en una lista de `also-notify` dentro de la declaración de una zona.
- `type` — Define el tipo de zona.

  Abajo se muestra una lista de las opciones válidas:
  - `delegation-only` — Refuerza el estado de delegación de las zonas de infraestructura tales como COM, NET u ORG. Cualquier respuesta recibida sin una delegación explícita o implícita es tratada como `NXDOMAIN`. Esta opción solamente es aplicable en TLDs o en archivos raíz de zona en implementaciones recursivas o de caché.
  - `forward` — Dice al servidor de nombres que lleve a cabo todas las peticiones de información de la zona en cuestión hacia otros servidores de nombres.

- `hint` — Tipo especial de zona que se usa para orientar hacia los servidores de nombres root que sirven para resolver peticiones de una zona que no se conoce. No se requiere mayor configuración que la establecida por defecto con una zona `hint`.
- `master` — Designa el servidor de nombres actual como el que tiene la autoridad para esa zona. Una zona se puede configurar como tipo `master` si los archivos de configuración de la zona residen en el sistema.
- `slave` — Designa el servidor de nombres como un servidor esclavo para esa zona. También especifica la dirección IP del servidor de nombres maestro para la zona.

`zone-statistics` — Configura `named` para mantener estadísticas concerniente a esta zona, escribiéndola a su ubicación por defecto (`/var/named/named.stats`) o al archivo listado en la opción `statistics-file` en la declaración `server`.

Declaración `view`

- `view "`*`<view-name>`*`"` — Crea vistas especiales dependiendo de en qué red esté el host que contacta el servidor de nombres. Esto permite a determinados hosts recibir una respuesta que se refiere a una zona particular mientras que otros hosts reciben información completamente diferente. Alternativamente, algunas zonas puede que sólo estén disponibles para ciertos hosts de confianza mientras que otros hosts menos autorizados, sólo pueden hacer peticiones a otras zonas.

  Se pueden usar múltiples visualizaciones, siempre y cuando sus nombres sean únicos. La opción `match-clients` especifica las direcciones IP que aplican a una vista particular. Cualquier declaración de `options` puede también ser usada dentro de una vista, ignorando las opciones globales ya configuradas por `named`. La mayoría de las sentencias `view` contienen múltiples declaraciones `zone` que aplican a la lista `match-clients`. El orden en que las sentencias `view` son listadas es importante, pues la primera sentencia `view` que coincida con una dirección IP de cliente particular es usada.

Ejemplo:

```
view "internal" {
       match-clients { 192.168.0.0/16; 127.0.0.0/8; };
       recursion yes;
       zone "mi-dominio.cl"  {
               type master;
               file "int.mi-dominio.cl";
               allow-transfer { any; };
               allow-update { none; };
       };
};
view "external" {
       match-clients { any; };
       recursion no;
       zone "mi-dominio.cl"  {
               type master;
               file "ext.mi-dominio.cl";
              allow-transfer { none; };
              allow-update { none; };
       };
};
```

Ejemplos de declaraciones de zonas:

Para un servidor primario:

```
zone "example.com" IN {
  type master;
  file "example.com.zone";
  allow-update { none; };
};
```

Para un servidor secundario:

```
zone "example.com" {
  type slave;
  file "example.com.zone";
  masters { 192.168.0.1; };
};
```

SERVICIO DE CONFIGURACIÓN AUTOMÁTICA. DHCP.

Justificación del servicio DHCP:

Cuando el número de equipos que componen una red local aumenta, la configuración del protocolo TCP/IP para todos los equipos de la red es una labor muy tediosa si ha de hacerse manualmente de uno en uno por parte del administrador, además puede provocar errores frecuentes como repetición de direcciones IP, etc.

Para facilitar la labor al administrador de una red se desarrolló el protocolo de configuración automática DHCP. Este protocolo permite suministrar una configuración válida de forma automática por parte de un servidor DHCP a los clientes de su red local que lo soliciten. Hay que señalar que no sólo proporciona una dirección IP sino la configuración completa: dirección IP, máscara, puerta de enlace, servidores DNS, etc.

Debemos darnos cuenta que el proceso de concesión sólo es posible para aquellos equipos que estén dentro de la misma red local, ya que el proceso de concesión se inicia mediante mensajes ethernet de difusión que como es sabido no pueden atravesar un router, por lo tanto no saldrán de la red local.

¿Como funciona?

La concesión de una dirección IP dinámicamente ocurre más o menos así:

- Etapa de descubrimiento:

Cuando un host no posee un número IP determinado (o sea, necesita un IP de un servidor DHCP), manda un mensaje de difusión llamado DHCPDISCOVER. Este mensaje es enviado dentro de la capa física de la red. Este mensaje incluye además algunos parámetros adicionales, como IPs sugeridos o tiempo de duración del número IP anterior que tuvo (si lo hubiera).

- Etapa de Ofrecimiento:

El mensaje llega a un servidor DHCP (los clientes que no posean el servicio DHCP ignoran este mensaje). El servidor responde de la misma manera física, pero con un mensaje llamado DHCPOFFER.

El cliente recibe UNO o MÁS mensajes DHCPOFFER de uno o mas servidores. El cliente entonces elige (por tiempo de respuesta, por IP, etc...es bastante oscuro el proceso de elección). Al elegir, el cliente envía un mensaje DHCPREQUEST al servidor que ha elegido para su IP (server identifier), junto con otras opciones. Si el cliente no recibe mensajes DHCPOFFER, expira la petición y reenvía un nuevo mensaje DHCPDISCOVER.

- Etapa de Encuentro:

El servidor recibe el broadcast con el mensaje DHCPREQUEST del cliente. El servidor responde

con un mensaje DHCPACK que contiene los parámetros para el cliente (el numero IP, máscara, etc). Aquí viene la etapa de "leasing" de IP. Si el servidor no puede satisfacer el mensaje DHCPREQUEST, el servidor igualmente debe responder con un DHCPNAK. El servidor marca los números IPs no disponibles.

- Etapa de Préstamo:

El cliente recibe el mensaje DHCPACK y revisa si la configuración esta OK. Si el cliente detecta un error, arroja un mensaje DHCPDECLINE y reinicia el proceso. Si en vez de recibir un DHCPACK, el cliente recibe un mensaje DHCPNAK, el cliente reinicia el proceso.

- Etapa de Devolución:

El cliente enviá un mensaje DHCPRELEASE al servidor cuando libera su IP.

Configurando Clientes

La configuración de cliente es la más fácil de todas.

En Linux Debian basta con modificar el fichero /etc/network/interfaces poniendo en la entrada iface eth0 el valor dhcp en lugar de static, y suprimir las opciones address, netmask, etc.

Por ejemplo:

```
# /etc/network/interfaces -- configuration file for ifup(8), ifdown(8)
# The loopback interface
iface lo inet loopback


iface eth0 inet dhcp
```

Configurando un servidor LINUX DHCP:

El primer paso es instalar el paquete dhcp del servidor, la última versión es dhcp3-server.

Ejecutamos el comando:

```
apt-get install dhcp3-server
```

o en las últimas versiones:

```
apt-get install isc-dhcp-server
```

Una vez instalado el paquete dispondremos en el sistema del demonio *dhcpd*, que se ejecuta automáticamente al arrancar el servidor por el correspondiente script. Este programa es el encargado de servir las direcciones IP a los clientes que las solicitan.

Hay que tener en cuenta que recién instalado el paquete al arrancar el servicio este falla porque no tiene ninguna configuración que se adapte a su dirección IP.

Para arrancar el servicio hay que usar el comando:

```
/etc/init.d/dhcp3-server start
```

`/etc/init.d/isc-dhcp-server start` (en la última versión)

Al modificar la configuración hay que reiniciar el servicio para que coja la nueva configuración con el comando:

```
/etc/init.d/dhcp3-server restart
```

`/etc/init.d/isc-dhcp-server restart` (en la última versión)

Nos interesan dos ficheros:

- El `dhcpd.leases` que contiene una base de datos con las direcciones concedidas, de este modo sabe cuales tiene libres y cuales están ocupadas y por quién. Su ubicación varía según las versiones, algunas de ellas son:

  `/var/lib/dhcp/dhcpd.leases`

  `/var/state/dhcp/dhcpd.leases`

- El otro fichero que nos interesa es `/etc/dhcp/dhcpd.conf`, en él se guarda la configuración según la cual actúa *dhcpd*. Por tanto su contenido es lo que más nos interesa. Veamos algunas de sus opciones de configuración:

```
# /etc/dhcp/dhcpd.conf
#----------opciones globales------------
#opciones mas comunes
      default-lease-time <tiempo>; tiempo de préstamo de IP por defecto (en segundos)
```

```
      max-lease-time <tiempo>;  cuanto tiempo estará prestado cada IP (en segundos)
      option subnet-mask <mask>;  indica la mascara de red
      option routers <ip1>,<ip2>;  indica los routers de la red
      option domain-name-servers <ip1>,<ip2>;  indica los servidores DNS
      option domain-name "<dominio>";  indica el dominio por defecto
      server-name "<nombre-host>";  indica el nombre del servidor DHCP
#otras opciones
      option smtp-server <ip>,<ip>;  indica servidores SMTP
      option pop3-server <ip>,<ip>;  indica servidores POP3
      option nntp-server <ip>,<ip>;  indica servidores de noticias
      option netbios-name-servers <ip>,<ip>;  indica servidores WINS
#----------opciones globales------------

#----------opciones locales------------
      subnet <network> netmask <mask>;  indica la subred y la mascara para los clientes
      {
            range <ip1> <ip2>; indica el rango de IPs a entregar
            range <ip1> <ip2>; ...
            ....
      }
#----------opciones locales------------
```

Es posible mezclar opciones globales y locales.

Veamos un ejemplo:

```
subnet 192.168.1.0 netmask 255.255.255.0 {
      option routers 192.168.1.1;
      option subnet-mask 255.255.255.0;
      option domain-name "midominio";
      range 192.168.1.2 192.168.1.50; # 48 computadores
      default-lease-time 21600; # 1/4 de un dia
      max-lease-time 43200; # 1/2 dia
      option domain-name-server 192.168.1.1;
      option netbios-name-servers 192.168.1.1;
}
```

Otras opciones de configuración no vistas en el ejemplo son aquellas que se utilizan para **conceder direcciones fijas** a determinados equipos. Por ejemplo para conceder la dirección 192.168.1.69 siempre a un equipo llamado *estacion1* pondríamos:

```
subnet .... {
      ....
      host estacion1 {
            hardware ethernet 12:34:56:78:AB:CD; # direccion MAC
            fixed-address 192.168.1.69; # direccion IP,
      }
      ...
}
```

Actualización de DNS dinámico.

En el caso de que, además, tengamos montado un servidor DNS, podemos desear que las máquinas de alguna de las zonas controladas por nuestro servidor DNS sea actualizada dinámicamente con las IP's que nuestro DHCP asigne. Esto es posible con bind9 y dhcp3; con versiones anteriores debemos descargar los paquetes correspondiente (dhcp-dns).

Supongamos que ya tenemos montada la zona “agora.es.” en nuestro DNS y deseamos que las máquinas de esa zona que obtengan IP's de nuestro DHCP se encuentren disponibles en el DNS.

Primero debemos copiar el fichero /etc/bind/rncd.key en el directorio /etc/dhcp.

Después, modificamos el fichero dhcpd.conf, añadiendo las siguientes líneas:

```
include "/etc/dhcp/rndc.key";

allow unknown-clients;
ddns-update-style interim;
ddns-domainname "agora.es.";
use-host-decl-names on;
zone agora.es. {
        primary 192.168.188.10;
        key rndc-key;
}
```

Posteriormente debemos modificar la configuración de la zona “agora.es.” en el servidor bind9, indicando en dicha zona que permitimos las modificaciones quedando la zona del siguiente modo en el fichero named.conf.local:

```
include “/etc/bind/rndc.key”;
zone "agora.es" {
      type master;
      file "/etc/bind/db.agora";
      allow-update { 192.168.188.0/24; key “rndc-key”; };
};
```

Por último, antes de reiniciar los servicios debemos modificar los permisos sobre el directorio /etc/bind, ya que el usuario que ejecuta *named* es *bind*, dicho usuario tiene permiso de lectura y

ejecución sobre este directorio, pero debe crear ficheros en él, por tanto debemos añadirle el permiso de escritura para poder crear y borrar ficheros. Si observamos los permisos con el comando:

```
ls -l /etc

…
drwxr-sr-x 2 root bind 4096 oct 9 12:45 bind
…
```

Lo que nos indica que el directorio /etc/bind tiene como propietario al usuario root y como grupo bind, el grupo tiene permiso de lectura y ejecución por tanto debemos añadirle el de escritura con el comado:

```
chmod g+w /etc/bind
```

Una vez terminadas las modificaciones podemos reiniciar los dos servicios:

```
/etc/init.d/isc-dhcp-server restart
/etc/init.d/bind9 restart
```

Con esto todos las máquinas que obtengan direcciones dinámicas en la subred donde se encuentren estas líneas, se encontrarán en la zona “agora.es.”. Si el nombre de la máquina cliente (se obtiene con el comando *hostname*) es aula16, su nombre DNS completo será aula16.agora.es y se podrá llegar a ella a través del DNS.

En el caso de que estemos asignando direcciones fijas (fixed-address) a un determinado host, en su declaración debemos añadir

```
update-static-leases on;
```

Veamos el siguiente ejemplo:

```
#Éstas opciones son generales pues no pertenecen a ninguna declaración
#de subred.

include "/etc/dhcp/rndc.key";

allow unknown-clients;
ddns-update-style interim;
ddns-domainname "agora.es.";
use-host-decl-names on;
zone agora.es. {
        primary 192.168.188.10;
        key rndc-key;
}

option domain-name-servers 192.168.188.10;
default-lease-time 600;
max-lease-time 7200;
subnet 192.168.188.0 netmask 255.255.255.0 {
      #el hostname de este host es, en realidad MICASA, pero funciona
```

```
      #igualmente.
      #pues, al final, obtiene como nombre MICASA.agora.es
      host MICA{
            hardware ethernet 00:c0:26:10:18:c6;
            fixed-address 172.23.240.15;
            #Activar esta opción es la única manera de actualizar el DNS
            #con direcciones fijas para un host (fixed-address)
            update-static-leases on;
      }
      #opciones generales
      option routers 192.168.188.2;
      range 192.168.188.20 192.168.188.100;
}
```

SERVIDORES WEB

Características generales de un servidor Web

Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando o cediendo una respuesta en cualquier lenguaje o Aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado o interpretado por un navegador web. Para la transmisión de todos estos datos suele utilizarse algún protocolo. Generalmente se utiliza el protocolo HTTP (protocolo de transferencia de hipertexto) para estas comunicaciones, perteneciente a la capa de aplicación del modelo OSI.

Servidores Web Linux

Apache

El servidor web apache es el más implantado entre los distintos servidores que ofertan servicios web en Internet. Para instalar estos servicios basta con instalar el paquete apache2 (apache versión 2):

apt-get install apache2

Una vez instalado este paquete ya podemos acceder al servidor web utilizando cualquier navegador, para ello bastará con indicar la ip del servidor web en la barra de direcciones de nuestro navegador o bien indicar el nombre del servidor si hemos instalado previamente el servidor DNS.

Existe un script para arrancar y parar el servidor igual que otros servicios Debian en la carpeta /etc/init.d

```
/etc/init.d/apache2 restart
/etc/init.d/apache2 start
/etc/init.d/apache2 stop
```

El archivo de configuración principal es /etc/apache2/apache2.conf, lo encontramos en la carpeta /etc/apache2 junto con otros archivos de configuración.

En el directorio /etc/apache2/sites-availables tenemos varios ficheros con la configuración de varios sitios, si queremos que se carguen en la configuración debemos crear un enlace en el directorio sites-enabled hacia ellos, por defecto aparece uno al sitio *default* que contiene la configuración del sitio por defecto.

La carpeta raíz del servidor apache es /var/www, los documentos que se encuentren dentro de la carpeta raíz del servidor web, son accesibles por la web. Al acceder a nuestro servidor web desde el navegador de un cliente, nos mostrará por defecto la página index.html localizada en /var/www.

Configuración de Apache

Secciones de configuración

Hay que distinguir entre **Sistema de ficheros** y **espacio web**. El sistema de ficheros es la visión de sus discos desde el punto de vista del sistema operativo. Por el contrario, el espacio web es lo que presenta el servidor web y que visualiza el cliente. El espacio web no tiene que tener correspondencia directa con el sistema de ficheros.

Por defecto en el espacio web la raiz "/" corresponde en el sistema de ficheros con "/var/www".

Secciones relacionadas con el sistema de ficheros

Las secciones `<Directory>` y `<Files>`, junto con sus contrapartes que usan expresiones regulares, aplican sus directivas a áreas del sistema de ficheros. Las directivas incluidas en una sección `<Directory>` se aplican al directorio del sistema de ficheros especificado y a sus subdirectorios. El mismo resultado puede obtenerse usando ficheros .htaccess. Por ejemplo, en la siguiente configuración, se activarán los índices de directorio para el directorio `/var/web/dir1` y sus subdirectorios.

```
<Directory /var/web/dir1>
Options +Indexes
</Directory>
```

Las directivas incluidas en una sección `<Files>` se aplicarán a cualquier fichero cuyo nombre se especifique, sin tener en cuenta en que directorio se encuentra. Por ejemplo, las siguientes directivas de configuración, cuando se colocan en la sección principal del fichero de configuración, deniegan el acceso a cualquier fichero llamado `private.html` sin tener en cuenta de donde se encuentre.

```
<Files private.html>
Order allow,deny
Deny from all
</Files>
```

Para referirse a archivos que se encuentren en un determinado lugar del sistema de ficheros, se pueden combinar las secciones `<Files>` y `<Directory>`. Por ejemplo, la siguiente configuración denegará el acceso a `/var/web/dir1/private.html`, `/var/web/dir1/subdir2/private.html`, `/var/web/dir1/subdir3/private.html`, y cualquier otra aparición de `private.html` que se encuentre en `/var/web/dir1/` o cualquiera de sus subdirectorios.

```
<Directory /var/web/dir1>
<Files private.html>
Order allow,deny
Deny from all
</Files>
</Directory>
```

Secciones relacionadas con el espacio web

La sección `<Location>` y su contraparte que usa expresiones regulares, cambian la configuración para el contenido del espacio web. Por ejemplo, la siguiente configuración evita que se acceda a cualquier URL que empiece por /private. En concreto, se aplicará a peticiones que vayan dirigidas a `http://yoursite.example.com/private` `http://yoursite.example.com/private123` y a `http://yoursite.example.com/private/dir/file.html` así como también a cualquier otra petición que comience por `/private`.

```
<Location /private>
Order Allow,Deny
```

```
Deny from all
</Location>
```

Caracteres comodín y expresiones regulares

Las secciones `<Directory>`, `<Files>`, y `<Location>` pueden usar caracteres comodín del tipo `fnmatch` de la librería estándar de C. El carácter "*" equivale a cualquier secuencia de caracteres, "?" equivale a cualquier carácter individual, y "[*seq*]" equivale a cualquier carácter en *seq*. Ningún carácter comodín equivale a"/", que debe siempre especificarse explícitamente.

Abajo se muestra un ejemplo en el que una sección de configuración que usa caracteres comodín en lugar de una expresión regular modifica la configuración de todos los directorios de usuario:

```
<Directory /home/*/public_html>
Options Indexes
</Directory>
```

Directivas más importantes

Listen

El comando **Listen** establece los puertos en los que secure Web server acepta las peticiones entrantes. secure Web server está configurado para el escuchar en el puerto 80 para comunicaciones no seguras y (en máquinas virtuales que define el servidor seguro) en el puerto 443 para comunicaciones seguras.

ServerName

El comando **ServerName** puede usarse para establecer el nombre de la máquina del servidor diferente al nombre real de máquina como por ejemplo, usar www.your_domain.com aunque el nombre real del servidor sea foo.your_domain.com. Nótese que **ServerName** debe ser un nombre "Domain Name Service" (DNS) válido que se tenga derecho a usar (no basta con inventar uno).

DocumentRoot

DocumentRoot es el directorio que contiene la mayoría de los ficheros HTML que se entregarán en respuesta a peticiones. El directorio predeterminado **DocumentRoot** para servidores seguros y no seguros es /var/www. Por ejemplo, el servidor puede recibir una petición para el siguiente documento:

```
http://your_domain/foo.html
```

El servidor buscará el fichero en el siguiente directorio por defecto:

```
/var/www/foo.html
```

Options

La directiva **Options** controla características del servidor que están disponibles en un directorio en particular. Por ejemplo, en los parámetros restrictivos especificados para el directorio raíz, el comando **Options** sólo permite **FollowSymLinks**. No hay características permitidas, salvo que el servidor pueda seguir enlaces simbólicos en el directorio raíz.

Por defecto, en el directorio **DocumentRoot**, **Options** está configurado para incluir los comandos **Indexes**, **Includes** y **FollowSymLinks**. **Indexes** permite al servidor generar un listado de un directorio si no se especifica el **DirectoryIndex** (index.html, etc.). **Includes** implica que se permiten inclusiones en el servidor y el comando **FollowSymLinks** permite al servidor seguir enlaces simbólicos en ese directorio.

AllowOverride

AllowOverride establece qué directivas **Options** puede obviar un fichero .htaccess. Por defecto, tanto el directorio raíz como **DocumentRoot** están configurados para no permitir la prevalencia de .htaccess.

Order

Order simplemente controla el orden en que **allow** y **deny** se evaluan. El servidor está configurado para evaluar **Allow** antes que **deny** para el directorio **DocumentRoot**.

Allow

Allow especifica qué peticionario puede acceder un directorio dado. El peticionario puede ser **all**, un nombre de dominio, una dirección IP, una dirección IP parcial, un par de máscaras de red, etc. El directorio **DocumentRoot** está configurado para permitir peticiones de **all** (cualquiera).

Deny

Deny funciona como **allow**, pero especifica a quién se niega el acceso. **DocumentRoot** no está configurado para **rechazar** peticiones de nadie.

DirectoryIndex

DirectoryIndex es la página por defecto que entrega el servidor cuando hay una petición de índice de un directorio especificado con una barra (/) al final del nombre del directorio.

Por ejemplo, cuando un usuario pide la página http://*your_domain*/*this_directory*/, recibe la página **DirectoryIndex** si existe, o un listado generado por el servidor. El valor por defecto para **DirectoryIndex** es index.html, index.htm index.shtml e index.cgi. El servidor intentará encontrar cualquiera de estos cuatro, y entregará el primero que encuentre. Si no encuentra ninguno y si **Options Indexes** se encuentra en el directorio, el servidor generará un listado, en formato HTML, de los subdirectorios y ficheros del directorio.

Alias

El comando **Alias** permite que haya directorios fuera del **DocumentRoot** a los que puede acceder el servidor. Cualquier URL que termine en un alias será automáticamente traducido por el recorrido del alias. Por defecto, ya existe un alias configurado.

Por ejemplo:

```
Alias /doc "/usr/share/doc"
```

El servidor puede acceder al directorio doc pero el directorio no está en **DocumentRoot**. doc, un alias, está en /usr/share/doc, y no en /var/www/doc.

Redirect

Cuando se cambia una página de sitio, el comando **Redirect** se puede usar para pasar del viejo URL al nuevo URL. El formato es como sigue:

```
Redirect /path/foo.html

http://new_domain/path/foo.html
```

Así que si se recibe una petición HTTP para un página que solía estar en http://*your_domain*/*path*/foo.html, el servidor devolverá el nuevo URL (http://*new_domain*/*path*/foo.html) al cliente, que tratará de coger el documento desde el nuevo URL.

UserDir

UserDir es el nombre del subdirectorio dentro del directorio de cada usuario dónde estarán los ficheros HTML que serán servidos por el servidor de Web.

Por defecto, el subdirectorio es public_html. Por ejemplo, el servidor podría recibir la siguiente petición:

```
http://your_domain/~username/foo.html
```

El servidor buscaría el fichero:

```
/home/username/public_html/foo.html
```

En el ejemplo, /home/username es el directorio del usuario (nótese que la ruta predeterminada a los directorios de los usuarios puede variar entre sistemas).

Hay que asegurarse que los permisos de los directorios de usuario sean correctos. El valor de los permisos deben ser de 0711. Los bits de lectura (r) y ejecución (x) deben estar activados en el directorio del usuario public_html (0755 valdrá). El valor de los permisos con que se servirán los ficheros desde public_html debe ser 0644 por lo menos.

Servidores virtuales:

Se puede configurar un hosting virtual basado en IP, nombres o puertos.

El hosting virtual basado en IPs usa la dirección IP de la conexión para determinar qué host virtual es el que tiene que servir. Por lo tanto, necesitará tener diferentes direcciones IP para cada host. Si usa hosting virtual basado en nombres, el servidor atiende al nombre de host que especifica el cliente en las cabeceras de HTTP. Usando esta técnica, una sola dirección IP puede ser compartida por muchos sitios web diferentes.

El hosting virtual basado en nombres es normalmente más sencillo, porque solo necesita configurar su servidor de DNS para que localice la dirección IP correcta y entonces configurar Apache para que reconozca los diferentes nombres de host. Auque hay algunos casos en los que es desaconsejable utilizar host virtuales basados en nombres, como son:

- Algunos clientes antiguos no son compatibles con el hosting virtual basado en nombres. Para que el hosting virtual basado en nombres funcione, el cliente debe enviar la cabecera de Host HTTP. Esto es necesario para HTTP/1.1
- El hosting virtual basado en nombres no se puede usar junto con SSL por la naturaleza del protocolo SSL.
- Algunos sistemas operativos y algunos elementos de red tienen implementadas técnicas de gestión de ancho de banda que no pueden diferenciar entre hosts a no ser que no estén en diferentes direcciones IP.

Creando un host virtual por nombre.

Si está añadiendo hosts virtuales a un servidor web ya existente, debe crear también un bloque `<VirtualHost>` para el host que ya tenga funcionando. Los valores de las directivas `ServerName` y `DocumentRoot` desde este nuevo host virtual deben tener los mismos valores que los de las directivas `ServerName DocumentRoot` globales. Ponga este host virtual como el primero en el archivo de configuración para que sea el que actúe como host por defecto.

Por ejemplo, suponga que está sirviendo el dominio `www.domain.com` y quiere añadir el host virtual `www.otherdomain.com`, que apunta a la misma dirección IP. Entonces, lo único que tiene que hacer es añadir lo siguiente al fichero de configuración:

```
NameVirtualHost *:80

<VirtualHost *:80>
ServerName www.domain.com
ServerAlias www.domain.es *.domain.net
DocumentRoot /var/www/domain
</VirtualHost>

<VirtualHost *:80>
ServerName www.otherdomain.com
DocumentRoot /var/www/otherdomain
</VirtualHost>
```

También puede que quiera que se acceda a un determinado sitio web usando diferentes nombres.

```
ServerAlias www.domain.es *.domain.net
```

Host virtual por IP

Para crear host virtuales por IP debemos incluir la ip en las directivas `NameVirtualHost y en el bloque <VirtualHost>` un ejemplo podría ser:

```
NameVirtualHost 192.168.5.10:80
NameVirtualHost 213.15.82.13:80

<VirtualHost 192.168.5.10:80>
ServerName www.domain.com
DocumentRoot /var/www/domain
</VirtualHost>

<VirtualHost 213.15.82.13:80>
ServerName www.otherdomain.com
DocumentRoot /var/www/otherdomain
</VirtualHost>
```

```
Podemos combinarlo con los nombres de dominio:
```

```
NameVirtualHost 192.168.5.10:80
NameVirtualHost 213.15.82.13:80

<VirtualHost 192.168.5.10:80>
ServerName www.domain.com
DocumentRoot /var/www/domain
</VirtualHost>

<VirtualHost 213.15.82.13:80>
ServerName www.otherdomain.com
DocumentRoot /var/www/otherdomain
</VirtualHost>

<VirtualHost 213.15.82.13:80>
ServerName www.midomain.com
DocumentRoot /var/www/midomain
</VirtualHost>
```

Host virtual por puerto

Para crear host virtuales por puerto debemos incluir un puerto distinto en las directivas `NameVirtualHost y en el bloque <VirtualHost>` . Debemos tener en cuenta que si utilizamos puertos distintos al 80 o al 443 debemos incluirlos en la directiva Listen, no basta con que aparezcan en las dos anteriores, un ejemplo podría ser:

```
Listen 80
Listen 8080

NameVirtualHost 172.20.30.40:80
NameVirtualHost 172.20.30.40:8080

<VirtualHost 172.20.30.40:80>
ServerName www.example1.com
DocumentRoot /www/domain-80
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
ServerName www.example1.com
DocumentRoot /www/domain-8080
</VirtualHost>
```

Debemos tener en cuenta que al usar puertos distintos al 80 hay que especificarlo en el navegador del cliente añadiendo dos puntos y el puerto a la URL, por ejemplo:

```
http://www.example1.com:8080
```

Se podrían mezclar las tres posibilidades:

```
Listen 80
Listen 8080

NameVirtualHost *:80
NameVirtualHost 172.20.30.40:8080
```

```
NameVirtualHost 213.15.82.13:8080

<VirtualHost *:80>
ServerName www.example1.com
DocumentRoot /www/domain1-80
</VirtualHost>

<VirtualHost *:80>
ServerName www.example2.com
DocumentRoot /www/domain2-80
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
ServerName www.example1.com
DocumentRoot /www/domain1-8080
</VirtualHost>

<VirtualHost 213.15.82.13:8080>
ServerName www.example1.com
DocumentRoot /www/domain2-8080
</VirtualHost>
```

Autentificación de usuario

Muchas veces tendremos la necesidad de mantener información sensible en nuestro sitio web por lo que la seguridad se vuelve un tema crucial. Dicha información sólo deberá ser vista por la gente que se supone debe verla. Como cada persona tendrá distintos privilegios sobre las cosas que puede o no ver, el administrador tendrá que implementar una forma segura de autentificación y autorización en el servidor. Podemos configurar sitios que deben ser accedidos identificándose mediante nombre de usuario y contraseña.

Creación de usuarios y claves:

El archivo de texto que guarda las claves puede almacenarse en cualquier directorio y tener cualquier nombre. En nuestro ejemplo usaremos *.passwd* que consiste en una lista de nombres de usuarios con sus respectivas claves (passwords) encriptadas por lo que este archivo NO debe ser creado o editado a mano. Este archivo deberá residir fuera de nuestra raíz de documentos, es decir fuera del árbol que contiene todos los archivos de nuestro sitio. Se puede usar el directorio /etc/apache2.

Para crear el archivo es necesario utilizar el programa htpasswd con la opción -c, para crear un nuevo archivo.

```
# cd /etc/apache2

# htpasswd -c .passwd Jorge

New Password:

Re-type new password:

Adding password for user Jorge
```

El primer argumento (después de -c) indica el nombre del archivo que se creará, el segundo indica el nombre

de usuario.

Si el archivo .passwd ya existiera debemos omitir la opción -c pues de no hacerlo borraremos a los usuarios que teníamos anteriormente

```
# cd /etc/apache2

# htpasswd .passwd Pepe

New Password:

Re-type new password:

Adding password for user Pepe
```

En este momento el archivo .passwd se ve así:

```
Jorge:YhW3VOGJDEXQU

Pepe:.BFIqlGrmtAiY
```

Configuración de grupos:

Continuando con el ejemplo, hagamos que tanto Jorge como Pepe pertenezcan al grupo webteam. Para esto crearemos un archivo llamado .groups y lo colocaremos en el mismo directorio en el que pusimos el archivo .passwd. El archivo .groups se ve de la siguiente manera:

```
webteam: Jorge Pepe
```

En este archivo colocaremos los nombres de nuestros grupos seguidos de los nombres de usuario de ese grupo separados por espacios. Un usuario puede pertenecer a varios grupos.

Modificaciones en el fichero de configuración:

Las directivas que utilizaremos son

- AuthType
- AuthName
- AuthUserFile
- AuthGroupFile
- Require

Si solo Jorge tendrá acceso al área privada, deberemos de agregar la siguiente sección al archivo httpd.conf:

```
<DIRECTORY /usr/local/apache/htdocs/admin>
AuthType Basic
AuthName "XPP: Admin Area"
AuthUserFile /etc/apache2/.passwd
Require user Jorge
```

```
</DIRECTORY>
```

Otro caso, si Jorge y Pepe tendrán acceso:

```
<DIRECTORY /usr/local/apache/htdocs/admin>
AuthType Basic
AuthName "XPP: Admin Area"
AuthUserFile /etc/apache2/.passwd
Require user Jorge Pepe
</DIRECTORY>
```

Y por último, si todos los miembros del grupo webteam tendrán acceso:

```
<DIRECTORY /usr/local/apache/htdocs/admin>
AuthType Basic
AuthName "XPP: Admin Area"
AuthUserFile /etc/apache2/.passwd
AuthGroupFile /etc/apache2/.groups
Require group webteam
</DIRECTORY>
```

AuthType

Permite seleccionar el tipo de autentificación para el directorio;

AuthName

Permite especificar el nombre de la autentificación, este nombre aparecerá en el diálogo del login provisto por el cliente (navegador).

AuthUserFile

Indica en donde se encuentra el archivo de usuarios.

AuthGroupFile

Indica en donde se encuentra el archivo de grupos.

Require

Indica aquellos usuarios o grupos que podrán acceder al directorio.

Autenficación con bases de datos.

El Servidor Apache soporta el módulo de autenticación, mod_authn_dbm, que usa las bases de datos DBM. El uso de bases de datos para la autenticación es más eficiente que un fichero de texto cuando el número de usuarios a incluir es muy alto.

Lo primero es cargar el módulo con el comando:

```
a2enmod authn_dbm
```

En el fichero de configuración debemos incluir las siguientes opciones, use la estructura siguiente:

<Location /private/>

AuthType Basic

AuthName "My Private Files"

AuthBasicProvider dbm

AuthDBMUserFile /etc/apache2/authdb

AuthDBMType DB

require valid-user

</Location>

Las directivas nueva a usar son:

AuthBasicProvider dbm

para indicar que usamos dbm.

AuthDBMUserFile /etc/apache2/authdb

para indicar el fichero que contiene la base de datos.

AuthDBMType DB

con esto indicamos el tipo de base de datos, en este caso DB.

Observe que la directiva AuthDBMUserFile también puede ser usada en archivos .htaccess.

Para crear la base de datos se usa el programa htdbm en Servidor Apache HTTP 2.0. El programa htdbm puede operar en una variedad de formatos de bases de datos; la opción -T se puede usar en la línea de comandos para especificar el formato a utilizar.

La tabla siguiente muestra los comandos más usuales para manipular la base de datos.

Acción	**comando htdbm (2.0)**
Añade un usuario a la base de datos (usando la contraseña dada)	`htdbm -b -TDB authdb username password`
Añade un usuario a la base de datos (le pide la contraseña)	`htdbm -TDB authdb username`
Eliminar el usuario de la base de datos	`htdbm -x -TDB authdb username`
Listar usuarios en la base de datos	`htdbm -l -TDB authdb`

Verificar una contraseña	`htdbm -v -TDB authdb username`

Las opciones -m y -s trabajan con htdbm, permitiendo el uso de los algortimos MD5 o SHA1 para las contraseñas hashing, respectivamente.

Cuando cree una nueva base de datos con htdbm, use la opción -c.

INSTALACIÓN DE JOOMLA 2.5

1-En tu terminal :

```
1. apt-get install apache
```

Listo, ya tienes instalado apache 2 en tu máquina.

Por lo regular después de la instalación el servidor web será iniciado automáticamente, pero si necesitas

2-Iniciarlo de manera manual teclea esto en tu terminal:

```
1. /etc/init.d/apache2 start
```

3-Si por alguna razón necesitas detener el servicio, escribe en tu terminal:

```
1. /etc/init.d/apache2 stop
```

El directorio donde se almacenan tus documentos web es: /var/www

Si todo resultó bien, debes ver una página web ordinaria al escribir http://localhost en la barra de direcciones de tu navegador.

AHORA A LA CONQUISTA DE PHP

1-Escribe en tu terminal:

```
1.  apt-get install php5 libapache2-mod-php5 php5-cli php5-mysql
```

2-Reinicia apache con:

```
1. /etc/init.d/apache2 restart
```

3-Para probar que se haya instalado correctamente vamos a crear un pequeño script en php, escribe en tu terminal:

```
1.  nano /var/www/correcto.php
```

PONEMOS ESTE CONTENIDO Y GUARDAMOS:

```
1. <?php
2.
3. phpinfo();
4.
5. ?>
```

Vamos a esta dirección: http://localhost/correcto.php — debes ver una página con información sobre tu instalación de php.

Terminamos con php.

AHORA VAMOS CON MYSQL

1-De nuevo, escribe en tu terminal:

1. apt-get install mysql-server

2-Probemos si todo funciona correctamente:

1. mysql -uroot -p (PRESIONAR TECLA ENTER)
2. poner (contraseña_que_pusiste)

Donde (contraseña_que_pusiste) es la contraseña que ingresaste durante la instalación de mysql.

Si deseas cambiar la contraseña de la cuenta root, ejecuta el siguiente comando después de ingresar a mysql:

1. SET PASSWORD FOR 'root'@'localhost' = PASSWORD('nueva_contraseña');

COMENZANDO LA INSTALACION DE JOOMLA 2.5

1-Descargarmos el paquete de joomla desde la pagina

2-Creamos una carpeta en el siguiente directorio:

1. mkdir /**var**/www/portal

3-Descomprimimos el paquete en el directorio anteriormente creado.

1. tar -xzvf Joomla_2.5.6-Spanish-Pack_Completo.tar.gz -C **var**/www/portal

Si está en zip usamos unzip teniendo copiado el fichero en la carpeta destino:

unzip Joomla_2.5.6-Spanish-Pack_Completo.zip

4-Para evitar problemas futuros, siempre que sea posible, conviene dar plenos permisos a la carpeta portal y todos sus componentes (es necesario disponer de permisos de lectura, escritura y ejecución en algunas carpetas a la hora de ampliar joomla, por ejemplo, si queremos instalar nuevas funcionalidades como componentes, plantillas o plugins).

para hacerlo, basta con poner en la terminal lo siguiente.

PARA HACERLO, BASTA CON PONER COMO PROPIETARIO DE TODOS LOS FICHEROS Y CARPETAS DE JOOMLA AL USUARIO www-data EN UN TERMINAL PONEMOS:

1. chown -R www-data:www-data /**var**/www/portal

5-Ahora cargamos joomla para empezar con la instalación en nuestro server local

http://localhost/portal

NOTA: Si hacemos la instalación desde una máquina remota debemos sustituir “localhost” por la dirección IP o el nombre de dominio de nuestro servidor. SUSTITUIR (PORTAL) POR EL NOMBRE CONCRETO DE LA CARPETA QUE HAS CREADO EN TU SERVER.

6-Comienzo de la instalación de joomla

7-Comprobacion de pre-instalación, joomla! comprueba las versiones de php y mysql instaladas y sus configuraciones para ver que todo está correcto para su instalación. en caso de que algo no esté correcto, habrá que hacer modificaciones en la configuración del servidor.

8-Licencia, tipo de licencia de joomla

9-Base de datos,tenemos que poner los datos de configuración de la base de datos (en donde se va a instalar el contenido del sitio web). el hospedaje o servidor es localhost, usuario es root y contraseña es la que se configuro en mysql anteriormente, nombre de la base de datos y prefijo de las tablas que seran usadas por joomla.

10-Configuración de ftp, como estamos instalando joomla en local no necesitas cambiar nada, déjalo todo tal cual está. haz clic en el botón siguiente para seguir con el proceso.

11-Configuración, aquí debes configurar algunos detalles sobre el sitio como su nombre, tu dirección de correo electrónico, establecer tu contraseña de administrador e incluso la posibilidad de migrar datos desde instalaciones previas de joomla.

12-Finalización de la instalación de joomla, esta última pantalla te advierte que debes borrar la carpeta instalación del servidor, por motivos de seguridad, para comenzar a utilizar joomla. hasta que no la borres y elimines todo su contenido no puedes comenzar a usarlo. pulsa el botón eliminar la carpeta de instalación para que joomla se encargue también de esto.

13-Luego de haber eliminado la carpeta de instalación le damos click sobre el botón que dice administrador, el botón sitio te mostrara un sitio ya pre-definido que crea joomla; y en el botón administración te conduce a la zona privada de tu sitio web desde donde podrás administrar su contenido. puedes acceder en cualquier momento a la administración del sistema si escribes en el campo dirección de tu navegador

http://localhost/portal/administrator

Siempre se mostrará una pantalla de acceso al sistema en el que tendrás que escribir como nombre de usuario admin y como contraseña la que pusiste en la instalación de joomla 2.5 en el paso 11.

Al final verán el panel de administración de joomla así

Joomla 2.5 ya está instalado! a partir de ahora tienes que configurarlo, adaptarlo a las necesidades de tu sitios, puedes descargar plugins, módulos,etc... todo lo necesario para que tu sitio se vea llamativo.

Servidor FTP en Linux

Existen múltiples servidores FTP para Linux. Vamos a centrarnos en el servidor **vsftpd** (*Very Secure FTP Daemon*), que es una servidor FTP rápido, seguro y fácil de configurar. Está disponible en la mayoría de los repositorios de las principales distribuciones de Linux.

Instalación:

Para instalar el paquete **vsftpd** debemos ejecutar con el usuario root:

apt-get install vsftpd

Tras la instalación se habrán creado:

- El fichero de configuración **/etc/vsftpd.conf**
- El usuario **ftp** que se incluye en el grupo **ftp.**
- El directorio **/srv/ftp**. El propietario de este directorio es el usuario **root** y su grupo es **ftp.** Este directorio será el directorio personal del usuario ftp, y será el utilizado por los usuarios anónimos.

  Cuando un usuario anónimo se conecta se le aplica **chroot** sobre el directorio **/srv/ftp**, esto quiere decir, que verá este directorio como si fuese el raiz "/", por tanto no tendrá acceso a los ficheros o directorios por encima de éste.

Configuración:

La configuración está contenida por completo en el fichero /etc/vsftpd. Es una configuración bastante sencilla, en la que cada directiva tiene el formato:

<*directiva*>=<*valor*>

sin espacios delante o detrás del signo "=".

Las directivas pueden tomar valores de tres tipos:

- Booleanas: sus valores pueden ser YES o NO.
- Numéricas.
- Cadenas.

Las directivas que no se configuran toman un valor por defecto.

Los comentarios se aplican a nivel de línea utilizando el carácter "#" al principio de la línea.

Otros ficheros que influyen en el comportamiento del servidor FTP son:

- /etc/ftpusers. Contiene una lista de usuarios que no tienen acceso al servidor FTP. Es usuado por muchos servidores FTP.
- /etc/vsftpd.user_list. Contiene una lista de usuarios a los que según la configuración se les

permite o no acceso al servidor. No se crea por defecto.

- /etc/vsftpd.chroot_list. Contiene una lista de usuarios locales a los que se les aplica chroot, dependiendo de la configuración. No se crea por defecto.

Directivas

Ejecución del servidor

El servidor vsftpd puede ejecutarse como proceso independiente. Para ello, hay que activar la directiva siguiente:

listen=YES

Gestión de usuarios

Las siguientes líneas establecen los tipos de usuario a los que se permite el acceso:

anonymous_enable=YES

local_enable=YES

La primera línea indica que cualquier usuario se puede conectar (YES) al servidor dando el nombre de anonymous o ftp.
La segunda indica que los usuarios locales pueden conectarse al servidor (YES). En el caso de que se desee impedirlo, bastará con cambiar el valor (NO) o comentar la línea con el signo "#" .
Cuando un usuario local se conecta, hay dos opciones:

1. El directorio raíz será el del sistema completo. Este usuario podrá descargarse archivos de *todo* el sistema en función de los permisos asignados.
2. Se conecta como usuario del sistema pero se le cambia su directorio raíz a su directorio home. Para ello tan sólo habrá que activar la directiva siguiente:

chroot_local_user=YES

En general, habrá que tener activada esta segunda opción para limitar el ámbito de actuación, bajo el servicio FTP, de los usuarios locales.
A partir de la versión 2.3.5 del vsftpd se ha introducido un cambio con respecto al enjaulado de los usuarios, no permitiendo el acceso a los usuarios que se enjaulen en un directorio sobre el que tengan permiso de escritura, como medida de seguridad. Esto resulta paradógico y no exento de controversia, ya que la solución pasa por quitar el permiso de escritura al usuario sobre su directorio personal, creando un subdirectorio sobre el que sí tenga permiso de escritura. Pero no siempre es factible quitar a un usuario el permiso de escritura sobre su directorio personal, con lo que la solución sería no enjaular a los usuarios, que es menos seguro.
En la versión 3 se ha subsanado este problema mediante la inclusión de un nueva directiva que al activarla permite enjaular a los usuarios aunque tengan permiso de escritura sobre el directorio. Esta directiva es:

allow_writeable_chroot=YES

Si interesa en el sistema que determinados usuarios locales no queden recluidos en su directorio home (es decir, no queden «enjaulados») cuando se conectan al servidor FTP, pueden incluirse las directivas siguientes donde el fichero /etc/vsftpd.chroot._list contiene los nombres de los usuarios que no quedarán enjaulados:

chroot_list_enable=YES

chroot_list_file=/etc/vsftpd.chroot_list

Los usuarios locales por defecto se conectan a su directorio home, de modo que si hacen chroot el directorio raiz será su directorio personal propio de cada usuario. Es posible indicar un directorio

raiz gobal para todos los usuarios locales con la directiva:

local_root=/srv/local

De este modo el directorio al que acceden todos los usuarios locales será /srv/local, si queremos que los usuarios puedan subir ficheros a este directorio se les debería dar permiso de escritura sobre él.

Carga de archivos para usuarios locales

Si se permite que los usuarios locales suban o carguen archivos al servidor FTP, habrá que habilitar la siguiente directiva:

write_enable=YES

Con esta directiva se pueden establecer los permisos con los que quedará el archivo subido al servidor FTP:

local_umask=022

De este modo, se indica que los permisos de los archivos serán 644, es decir, de lectura y escritura para el dueño y solo de lectura para el grupo y los demás usuarios (*r w - r - - r - -*).

Por último, puede habilitarse la carga de archivos en formato ASCII en el servidor. Bastará con la directiva siguiente:

ascii_upload_enable=YES

Carga de archivos para usuarios anónimos

Por defecto la carga de archivos para los usuarios anónimos está desactivada. Para activarla hay que incluir la directiva:

anon_upload_enable=YES

Es conveniente crear un directorio (*incoming,* por ejemplo) con permisos suficientes para que puedan escribir en él los usuarios de modo que sólo puedan subir ficheros a ese directorio.

mkdir /srv/ftp/incoming
chown root:ftp /srv/ftp/incoming
chmod 775 /srv/ftp/incoming

Por defecto, los archivos cargados lo harán con permisos 600. Si se quiere que un usuario anónimo pueda descargarlos, habrá que cambiarlos a 644. Para ello, deberemos incluir la directiva:

anon_umask=022

En el caso de que se quiera permitir que los usuarios anónimos modifiquen los archivos pertenecientes a usuarios anónimos y que puedan crear directorios, habrá que activar estas otras directivas:

anon_other_write_enable=YES
anon_mkdir_write_enable=YES

Visualización de mensajes

Se puede establecer un mensaje de bienvenida cuando se conecta el usuario al servidor FTP con esta directiva:

ftpd_banner=”Bienvenido al Servidor FTP del aulaS4.com”

Tiempos y números de conexión

Existen una serie de directivas relacionadas con los tiempos de conexión.

A la hora de establecer la conexión (por defecto, 60 segundos) de un usuario remoto en modo pasivo, se emplea la directiva siguiente:

accept_timeout=60

En cambio, para indicar el tiempo máximo que el servidor espera cuando una transferencia no progresa (por lo general, 300 segundos), habrá que utilizar esta otra:

data_connection_timeout=300

En cuanto al tiempo máximo concedido a un usuario remoto que no está activo (es decir, que no

está ejecutando órdenes ftp), se establece con la directiva siguiente.

idle_session_timeout=300

Al igual que en el caso anterior, se asignan por defecto 300 segundos. Pasado este tiempo, se corta la conexión.

Especificar el número máximo de clientes simultáneos que tienen permitido conectarse al servidor.

max_clients=5

El valor por defecto es 0, con lo cual no se pone límite.

Ratios de transmisión

Especificar la cantidad máxima de datos transmitidos por usuarios anónimos en bytes por segundo.

anon_max_rate=1024

El valor por defecto es 0, que no pone límite.

Para indicar un ratio máximo de transmisión para los usuarios locales en bytes por segundo se usa:

local_max_rate=10240

El valor por defecto es 0, que no pone límite.

Modos de conexión

Para permitir las conexiones en modo activo.

port_enable=YES

El valor predeterminado es YES. Si se pone a NO no se permitirán las conexiones activas.

Para permitir conexiones pasivas se usa:

pasv_enable=YES

El valor predeterminado es YES. Si se pone a NO no se permitirán las conexiones pasivas.

En el caso de conexiones pasivas, se puede indicar un conjunto de puertos por encima del 1024 que serán usados por el servidor para las conexiones de datos.

pasv_min_port=30000

pasv_max_port=30100

Con esto se usarian los puertos del 30000 al 30100. El valor mínimo que se puede usar es 1024 y el máximo 65535.

Cuando el servidor se encuentra detrás de un enrutador NAT, los clientes al iniciar la conexión de datos deben indicar como dirección de destino la dirección pública del enrutador NAT. Con esta directiva se le indica al servidor cual es la dirección pública que debe proporcionar a los clientes:

pasv_address=213.14.85.13

Esta directiva no tiene valor predeterminado.