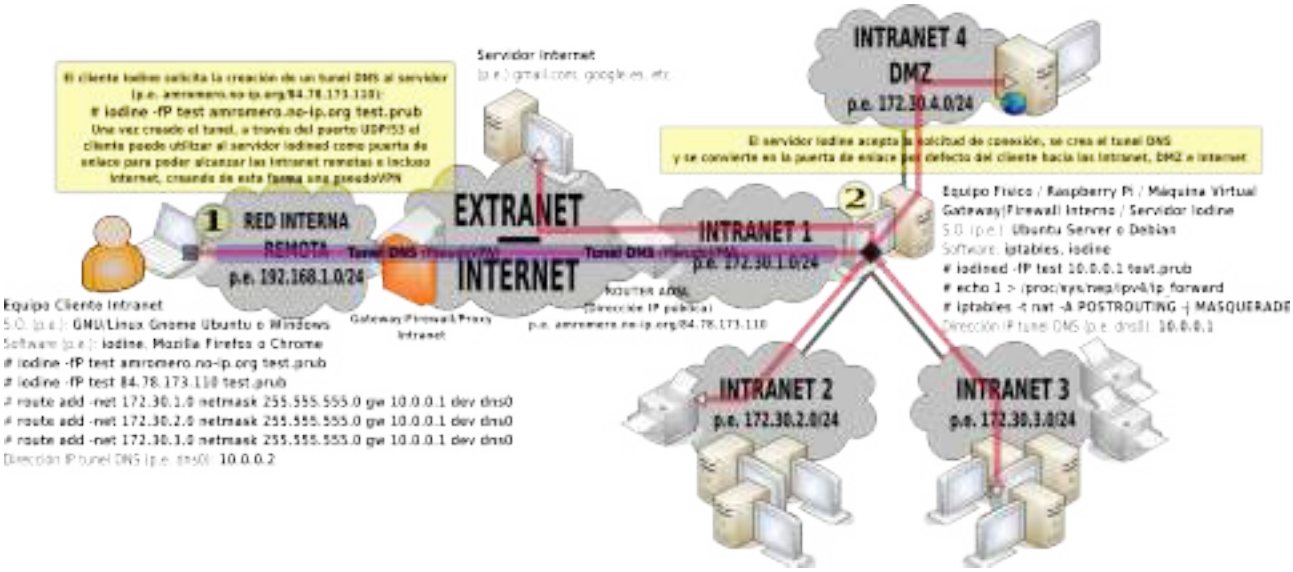
PRÁCTICAS MÓDULO DE

SEGURIDAD INFORMÁTICA Y ALTA DISPONIBILIDAD

VERSIÓN 14.5



IES TIEMPOS MODERNOS
Zaragoza – ESPAÑA
Arturo Martín Romero

Hoy en día la seguridad informática y la necesidad de garantizar una alta disponibilidad de la información ha pasado a ser una faceta importantísima para la mayoría de las empresas, independientemente del sector al que pertenezcan, debido a la gran dependencia que éstas tienen de multitud de herramientas informáticas: software de gestión, contabilidad, etc.

A través de este manual de prácticas no se pretende dar una solución directa a los problemas de seguridad que pueda tener una determinada empresa, ni tampoco se van a presentar "recetas" que puedan protegernos de ciertas amenazas. Esta no es su pretensión. La finalidad que se persigue no va ser otra más que la de presentar diferentes posibilidades de mejora de nuestra infraestructura informática que podrían prevenir la aparición de posibles problemas de seguridad a posteriori.

Al mismo tiempo, a través de los innumerables ejercicios prácticos resueltos que se proponen realizar, este manual de prácticas trata de complementar a los innumerables libros de seguridad con carácter teórico que existen, y que en muchísimas ocasiones a muchos de los que nos dedicamos a este mundo de la informática nos suelen dejar insatisfechos a consecuencia de la gran cantidad de teoría al respecto que presentan, sin apenas entrar en la parte práctica.

Espero que este manual pueda despertar en alguno de los que estáis leyendo esta introducción un mínimo interés por el área de la informática relativa a la implementación de soluciones técnicas encaminadas a mejorar la seguridad y la disponibilidad de la información. Estoy seguro que todo aquel que implemente los ejercicios prácticos que aquí se presentan le ayudará a ejercitar su sagacidad, la cual es vital tanto para disfrutar de su implementación, como para encontrar puntos de mejora, que seguro que los hay.

Para cualquier cuestión, constructiva o destructiva, que pueda mejorar el presente manual, por favor no dudéis en poneros en contacto conmigo.

Dedicado a todos aquellos que creen firmemente
en el software y hardware libre como elemento vital
de mejora de nuestra sociedad,

Arturo Martín Romero
amartinromero@gmail.com

Seguridad Informática y Alta Disponibilidad

Seguridad Perimetral / Firewall / Proxy HTTP / Proxy SOCKS
Listas de Control de Acceso (ACLs) / Cuotas
Criptografía / Claves Simétricas y Asimétricas
Servicios SSH / HTTPS / SFTP / VPNs
Copias de Seguridad Remotas Cifradas
Cluster Hadoop (HDFS) / RAID / LVM
Redes Privadas Virtuales (VPN) / Encapsulamiento Túnel DNS
Seguridad en Redes Wireless / Servicio de autenticación RADIUS

Índice de Prácticas

Con la finalidad facilitar la implementación de algunas de las prácticas que se presentan en este libro/manual se ha creado una lista de reproducción en youtube (*se irán añadiendo o modificando los vídeos constantemente*):

http://www.youtube.com/playlist?list=PLoKwZscaHXlDVqbpY32SgVYuWfS8F-p2X

¡¡Observación!! Para la realización de los videotutoriales explicativos se ha hecho uso de los siguientes programas bajo entorno Gnome Shell Ubuntu:

(1) SimpleScreenRecorder. Tras probar varios de ellos, de entre todos los programas pensados exclusivamente para grabar el escritorio de trabajo, se ha seleccionado este por sus características y buenos resultados. No esta disponible en los repositorios por defecto, por lo que es necesario agregar un nuevo repositorio:

```
[usuario@linux]$ sudo add-apt-repository ppa:maarten-baert/simplescreenrecorder
[usuario@linux]$ sudo apt-get update
[usuario@linux]$ sudo apt-get install simplescreenrecorder ubuntu-restricted-extras
[usuario@linux]$ simplescreenrecorder
```

(2) Mplayer. Me permite mostrar en los videotutoriales una pequeña ventana (*sin barras de título, menús, etc.*) con la imagen capturada por la Webcam. De esta forma, al mismo tiempo que se ve como realizar la práctica, se me muestra a mí mientras lo explico. Esta disponible en los repositorios por defecto.

```
[usuario@linux]$ sudo apt-get install mplayer
[usuario@linux]$ mplayer -noborder -quiet -ontop -fps 15 -geometry 200x150+1110+560 noaudio tv://
```

(3) OpenShot. Me permite editar los vídeos resultantes. Es decir, dispone de varias pistas o capas, en las cuales pudemos ir añadiendo vídeos, música o imágenes que irán componiendo el vídeo resultante. Esta disponible en los repositorios por defecto.

```
[usuario@linux]$ sudo apt-get install openshot
[usuario@linux]$ openshot
```

Práctica Nº1.- Conceptos Básicos. Tablas de enrutamiento. Sistemas de Virtualización

Antes de empezar a tratar asuntos relativos a la seguridad de nuestros sistemas informáticos, es aconsejable profundizar en determinados aspectos que seguramente los alumnos conocerán en mayor o menor medida, y que serán ampliamente utilizados a lo largo de las distintas prácticas que aquí se presentan. En concreto, se repasarán los siguientes apartados:

1. Direccionamiento IP
2. Tablas de enrutamiento.
3. Sistemas de Virtualización.

1.1.- Direccionamiento IP

En este primer apartado repasaremos la importancia que tiene para un equipo tener configuradas al menos una dirección IP, una puerta de enlace y una dirección IP de un servidor DNS.

Las direcciones IP permiten que todo equipo de una red se encuentre identificado y localizado. En el caso de que un equipo no disponga de una dirección IP, estará quedará incomunicado, y no podrá comunicarse con el resto de equipos de la red lógica a la que pertenezca. No obstante, una vez asignada una dirección IP a un equipo, este únicamente podrá comunicarse con aquellos equipos que estén conectados a él a través de un hub o switch, y pertenezcan a la misma red lógica. En el caso de que un equipo desee conmunicarse con equipos externos a su red lógica será necesario un equipo intermediario, denominado ***gateway/puerta de enlace/pasarela/router/encaminador*** en le mundo informático. Por último, será necesario indicar al equipo al menos una dirección IP de algún servidor DNS (*Domain Name Server, Servidor de Nombres de Dominio*) para de esta forma poder hacer uso de nombres de dominio (*p.e. "equipo1.midominio.es", "www.google.com", etc.*) en lugar de direcciones IP a la hora de hacer referencia a un equipo.

¡¡Importante!! Un equipo necesita que le configuren tres direcciones IP para que pueda navegar a través de Internet: **1)** una dirección IP dentro del rango de direcciones IP posibles de la red o subred a la que pertenece, **2)** la dirección IP de la máquina dentro de su red o subred que hace la funciones de puerta de enlace hacía el exterior, y **3)** la dirección IP de la máquina dentro de la Intranet o Extranet encargada de resolver nombres de dominio al equipo que se lo solicite, permitiéndole de esta forma poder hacer uso de nombres de dominio en lugar de direcciones IP.

En el caso de querer profundizar más en este recordatorio relativo al direccionamiento IP, se aconseja hojear el **apéndice A**, localizado al final del presente manual de prácticas, donde se pueden encontrar varios ejercicios prácticos resueltos relativos a este asunto.

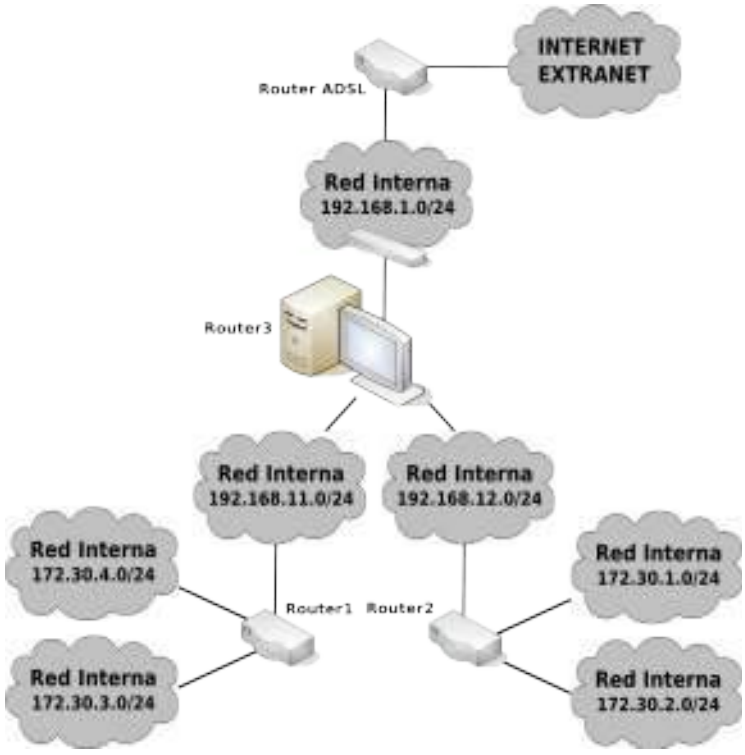
1.2.- Tablas de Enrutamiento

Una tabla de enrutamiento no es más que un conjunto de reglas que permite informar a un equipo de que tiene que hacer, por donde tiene que enviar los paquetes TCP/IP, en el caso de que desee comunicarse con otro equipo, independientemente de que éste se encuentre dentro de su red de área local o fuera.

Para repasar este asunto, a continuación se presentan unos cuantos ejercicios prácticos, que seguro resultarán muy útiles.

Ej. Práctico 1.2.1: Tabla de Enrutamiento a través de Esquema de Red I

Observa el siguiente esquema de red, e indica **cuales serían las tablas de enrutamiento** que nos mostrarían en pantalla los **routers nº1 y nº2** si tenemos en cuenta que tienen asignada la dirección IP más alta posible asignable dentro de la red lógica en la que se encuentran. La asignación de eth0, eth1, etc. podrán asignarse libremente mientras ésta sea coherente.



Solución Ej. Pr. 1.2.1.I.- Tablas de Enrutamiento y Esquema de Red

Tal como se va a mostrar a continuación, las tablas de enrutamiento del router Nº1 y Nº2 admiten dos posibles configuraciones, una redundante y otra simplificada. En dicha tabla, se describen en primer lugar las redes lógicas que son alcanzables por el router sin necesidad de una puerta de enlace (*el router tiene una dirección lógica dentro de la red, lo que le permite comunicarse con el resto de equipos de su misma red lógica a través de un hub o switch*), y a

continuación el resto de las redes lógica alcanzables a través de un gateway/pasarela/puerta de enlace. Recordar que en la tabla de enrutamiento tan sólo tiene que mostrarse el primer salto, o gateway, que le permitirá llegar hacía su destino, independientemente que entre ambos haya que atravesar una cantidad de routers indeterminado (*en la comunicación entre dos equipos los routers intermedios tan sólo deben conocer quien es el siguiente router al que tienen que pasarle los paquetes TCP/IP para que acaben llegando a su destino*).

Por ejemplo, para el **router Nº1**, si observamos detenidamente el esquema de red anterior, su tabla de enrutamiento sin simplificar sería la siguiente:

Redes Destino	Máscara	Gateway	Interfaz Red
172.30.4.0	255.255.255.0	0.0.0.0	Eth0
172.30.3.0	255.255.255.0	0.0.0.0	Eth1
192.168.11.0	255.255.255.0	0.0.0.0	Eth2
192.168.12.0	255.255.255.0	192.168.11.253	Eth2
172.30.1.0	255.255.255.0	192.168.11.253	Eth2
172.30.2.0	255.255.255.0	192.168.11.253	Eth2
192.168.1.0	255.255.255.0	192.168.11.253	Eth2
0.0.0.0	0.0.0.0	192.168.11.253	Eth2

Y si lo razonamos, de manera simplificada, su tabla de enrutamiento sería la siguiente (*siempre pueden simplificarse aquellos destinos que sean alcanzables por el mismo gateway que da salida hacía Internet, la red por defecto "0.0.0.0"*):

Redes Destino	Máscara	Gateway	Interfaz Red
172.30.4.0	255.255.255.0	0.0.0.0	Eth0
172.30.3.0	255.255.255.0	0.0.0.0	Eth1
192.168.11.0	255.255.255.0	0.0.0.0	Eth2
0.0.0.0	0.0.0.0	192.168.11.253	Eth2

Ej. Práctico 1.2.2: Configuración de Red de un Router en GNU/Linux

Siguiendo con el ejercicio anterior, si el **router nº3 fuera un equipo GNU/Linux** indica la lista de comandos *(script de configuración)* necesarios para configurar tanto sus interfaces de red como las rutas necesarias para poder acceder a todas las redes que aparecen en el esquema, teniendo en cuenta que **la dirección de la Intranet del router ADSL es 192.168.1.1**, y que el router nº3 va a recibir **las direcciones IP libres más altas que queden disponibles dentro de cada una de las redes en las que se encuentra**. Configura igualmente sus **servidores DNS**, sabiendo que la direcciones de estos son **195.55.130.247 y 8.8.8.8**, además de aquello que sea necesario para que actúe como un gateway o enrutador hacia el exterior para los equipos de las redes internas *(iptables)*.

Solución Ej. Pr. 1.2.2.I.- Tablas de Enrutamiento y Scripts de Configuración de Red

Si observamos el esquema de red, advertiremos que la tabla de enrutamiento del router nº3 sería la siguiente:

Redes Destino	Máscara	Gateway	Interfaz Red
192.168.1.0	255.255.255.0	0.0.0.0	Eth2
192.168.11.0	255.255.255.0	0.0.0.0	Eth0
192.168.12.0	255.255.255.0	0.0.0.0	Eth1
172.30.3.0	255.255.255.0	192.168.11.254	Eth0
172.30.4.0	255.255.255.0	192.168.11.254	Eth0
172.30.1.0	255.255.255.0	192.168.12.254	Eth1
172.30.2.0	255.255.255.0	192.168.12.254	Eth1
0.0.0.0	0.0.0.0	192.168.1.1	Eth2

A continuación mostraremos dos maneras de configurar un equipo GNU/Linux como router o gateway: (1) mediante la ayuda de scripts del sistema o (2) creando nuestro propio script de configuración:

(1) En el caso de querer hacer uso de los scripts ya proporcionados por el sistema haríamos los siguiente:

a) Editaríamos el archivo **/etc/network/interfaces** para asignar las direcciones IP a las interfaces de red del equipo, indicar quien es su gateway por defecto, quienes son sus servidores DNS y para añadir las reglas de enrutamiento estático que le informen de como alcanzar todas las redes del entorno:

```
[root@linux]# nano /etc/network/interfaces
# Contenido del archivo interfaces:
auto lo
iface lo inet loopback

auto eth0 eth1 eth2
iface eth0 inet static
address 192.168.11.253
netmask 255.255.255.0
up route add -net 172.30.3.0 netmask 255.255.255.0 gw 192.168.11.254
up route add -net 172.30.4.0 netmask 255.255.255.0 gw 192.168.11.254
down route add -net 172.30.3.0 netmask 255.255.255.0 gw 192.168.11.254
down route add -net 172.30.4.0 netmask 255.255.255.0 gw 192.168.11.254

iface eth1 inet static
```

```
address 192.168.12.253
netmask 255.255.255.0
up route add -net 172.30.1.0 netmask 255.255.255.0 gw 192.168.12.254
up route add -net 172.30.2.0 netmask 255.255.255.0 gw 192.168.12.254
down route add -net 172.30.1.0 netmask 255.255.255.0 gw 192.168.12.254
down route add -net 172.30.2.0 netmask 255.255.255.0 gw 192.168.12.254

iface eth2 inet static
address 192.168.1.254
netmask 255.255.255.0
gateway 192.168.1.1
dns-nameservers 8.8.8.8 195.55.130.247
```

Para que surta efecto la configuración anterior podemos reactivar las interfaces de red o directamente reiniciar el equipo (*init 0*):

```
[root@linux]# ifdown -a
[root@linux]# ifup -a
```

b) Tras la configuración IP tan sólo deberemos activar el ip_forwarding para permitir el reenvío de paquetes entre las diferentes interfaces de red, además de activar el enmascaramiento de los paquetes que le atraviesen con la finalidad de que los paquetes de retorno puedan llegar a su destino (*cambia la dirección IP de origen del equipo de la intranet por la dirección IP de la interfaz de salida del router*). Una posibilidad sería añadir los comandos necesarios al script **/etc/init.d/rc.local**, el cual se ejecuta al iniciarse la máquina:

```
[root@linux]# echo 'echo "1" > /proc/sys/net/ipv4/ip_forward' >> /etc/init.d/rc.local
[root@linux]# echo 'iptables -t nat -A POSTROUTING -j MASQUERADE' >> /etc/init.d/rc.local
```

(2) Como segunda opción podríamos crear nuestro propio script de configuración. Para ello deberíamos tener en cuenta que antes de su creación y ejecución, deberíamos inhabilitar los servicios disponibles en GNU/Linux encargados de gestionar la red: a) "**network-manager**" si nos encontramos en entorno gráfico o b) "**networking**" si usamos un entorno de consola y ficheros de configuración de red tales como "**/etc/network/interfaces**", para evitar conflictos. Para la gestión de estos servicios de red, habilitación e inhabilitación, podemos instalar la herramienta clásica de gestión de servicios en GNU/Linux "**chkconfig**" (*las distribuciones Ubuntu en sus últimas versiones no la integran*), o la herramienta software que la sustituye "**update-rc.d**". En concreto, para inhabilitarlos haríamos lo siguiente:

```
[usuario@linux]$ sudo update-rc.d network-manager disable
[usuario@linux]$ sudo update-rc.d networking disable
```

O lo que sería equivalente:

```
[usuario@linux]$ sudo su
[root@linux]# update-rc.d network-manager disable
```

```
[root@linux]# update-rc.d networking disable
```

En el caso de disponer de "**chkconfig**":

```
[usuario@linux]$ sudo chkconfig network-manager off
[usuario@linux]$ sudo chkconfig networking off
```

En relación al script, quedaría de la siguiente forma:

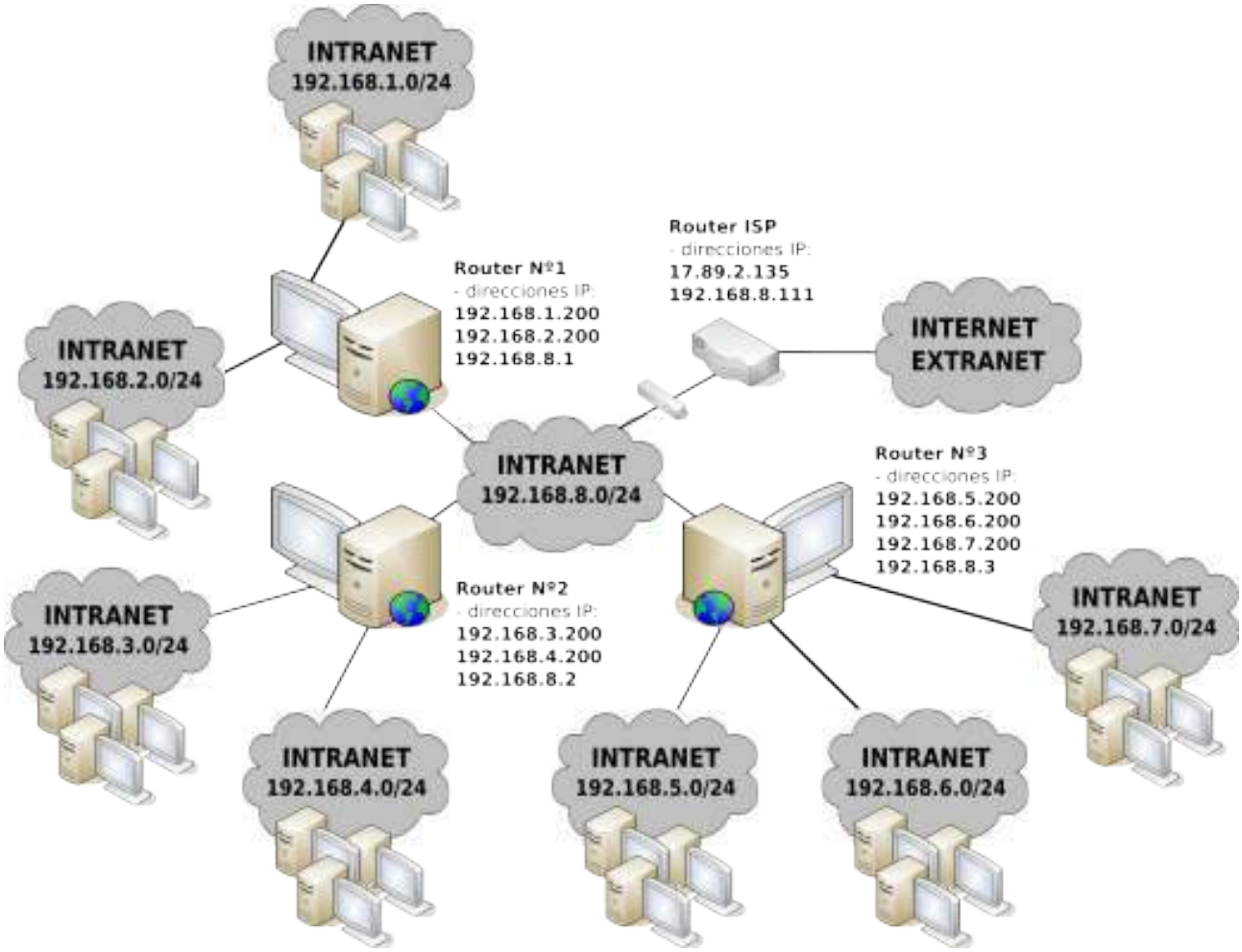
```
[root@linux]# nano /etc/init.d/conf-red
#!/bin/bash
# Comenzamos configurando las interfaces de red
ifconfig eth0 192.168.11.253 netmask 255.255.255.0
ifconfig eth1 192.168.12.253 netmask 255.255.255.0
ifconfig eth2 192.168.1.254 netmask 255.255.255.0
# Configuramos las reglas de enrutamiento que requieren de gateway
route add -net 172.30.3.0 netmask 255.255.255.0 gw 192.168.11.254 dev eth0
route add -net 172.30.4.0 netmask 255.255.255.0 gw 192.168.11.254 dev eth0
route add -net 172.30.1.0 netmask 255.255.255.0 gw 192.168.12.254 dev eth1
route add -net 172.30.2.0 netmask 255.255.255.0 gw 192.168.12.254 dev eth1
route add default gw 192.168.1.1 dev eth2
# Terminamos permitiendo reenviar y enmascarar paquetes TCP/IP para que haga de gateway
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -j MASQUERADE
```

Por útlimo, con la finalidad de que el script se ejecute cada vez que se inicia la máquina (runlevel 2) ejecutaremos los siguientes comandos:

```
[root@linux]# chmod +x /etc/init.d/conf-red
[root@linux]# ln -s /etc/init.d/conf-red /etc/rc2.d/S99conf-red
```

Ej. Práctico 1.2.3: Tabla de enrutamiento a través de Esquema de Red II

Al igual que en los ejercicios anteriores, **indica como sería la tabla de enrutamiento** del equipo "**ROUTER/GATEWAY/FIREWALL 1**" que aparece en el esquema de red que se muestra en la siguiente figura, teniendo en cuenta que debe poder alcanzar cualquier equipo de las Intranets e Internet. De igual forma, **indica los scripts o comandos que serían necesarios ejecutar** para una correcta configuración (*ifconfig, route e iptables*).



Solución Ej. Pr. 1.2.3.I.- Tablas de Enrutamiento y Scripts de Configuración de Red

Si observamos el esquema de red anterior advertiremos que la tabla de enrutamiento que mostraría el router Nº1 sería la siguiente (*route -n*):

Redes Destino	Máscara	Gateway	Interfaz Red
192.168.1.0	255.255.255.0	0.0.0.0	eth0
192.168.2.0	255.255.255.0	0.0.0.0	eth1
192.168.3.0	255.255.255.0	192.168.8.2	eth2
192.168.4.0	255.255.255.0	192.168.8.2	eth2
192.168.5.0	255.255.255.0	192.168.8.3	eth2
192.168.6.0	255.255.255.0	192.168.8.3	eth2
192.168.7.0	255.255.255.0	192.168.8.3	eth2
192.168.8.0	255.255.255.0	0.0.0.0	eth2
0.0.0.0	0.0.0.0	192.168.8.111	eth2

Para su configuración se podrá hacer uso de **(1)** scripts de configuración disponibles en el sistema o **(2)** crear nuestro propio script de configuración:

(1) En el caso de querer hacer uso de los scripts ya proporcionados por el sistema haríamos los siguiente:

a) Editaríamos el archivo **/etc/network/interfaces** para asignar las direcciones IP a las interfaces de red del equipo, indicar quien es su gateway por defecto, quienes son sus servidores DNS y para añadir las reglas de enrutamiento estático que le informen de como alcanzar todas las redes del entorno:

```
[root@linux]# nano /etc/network/interfaces
# Contenido del archivo interfaces:
auto lo eth0 eth1 eth2
iface lo inet loopback

iface eth0 inet static
address  192.168.1.200
netmask 255.255.255.0

iface eth1 inet static
address 192.168.2.200
netmask 255.255.255.0

iface eth2 inet static
address 192.168.8.1
netmask 255.255.255.0
gateway 192.168.8.111
dns-nameservers 8.8.8.8
post-up route add -net 192.168.3.0 netmask 255.255.255.0 gw 192.168.8.2
post-up route add -net 192.168.4.0 netmask 255.255.255.0 gw 192.168.8.2
post-up route add -net 192.168.5.0 netmask 255.255.255.0 gw 192.168.8.3
post-up route add -net 192.168.6.0 netmask 255.255.255.0 gw 192.168.8.3
post-up route add -net 192.168.7.0 netmask 255.255.255.0 gw 192.168.8.3
```

Para que surta efecto la configuración anterior deberemos reactivar las interfaces de red o directamente reiniciar el equipo (*init 0*):

```
[root@linux]# ifdown -a
[root@linux]# ifup -a
```

b) Tras la configuración IP tan sólo deberemos activar el ip_forwarding para permitir el reenvío de paquetes entre las diferentes interfaces de red, además de activar el enmascaramiento de los paquetes que le atraviesen con la finalidad de que los paquetes de retorno puedan llegar a su destino (*cambia la dirección IP de origen del equipo de la intranet por la dirección IP de la interfaz de salida del router*). Una posibilidad sería añadir los comandos necesarios al script

/etc/init.d/rc.local, el cual se ejecuta al iniciarse la máquina:

```
[root@linux]# echo 'echo "1" > /proc/sys/net/ipv4/ip_forward' >> /etc/init.d/rc.local
[root@linux]# echo 'iptables -t nat -A POSTROUTING -j MASQUERADE' >> /etc/init.d/rc.local
```

(2) En el caso de querer crear nuestro propio script de configuración de red, podríamos hacer uso del siguiente:

```
[root@linux]# nano /etc/init.d/conf-red
#!/bin/bash
# Comenzamos configurando las interfaces de red
ifconfig eth0 192.168.1.200 netmask 255.255.255.0
ifconfig eth1 192.168.2.200 netmask 255.255.255.0
ifconfig eth2 192.168.8.1 netmask 255.255.255.0
# Configuramos las reglas de enrutamiento que requieren de gateway
route add -net 192.168.3.0 netmask 255.255.255.0 gw 192.168.8.2 dev eth2
route add -net 192.168.4.0 netmask 255.255.255.0 gw 192.168.8.2 dev eth2
route add -net 192.168.5.0 netmask 255.255.255.0 gw 192.168.8.3 dev eth2
route add -net 192.168.6.0 netmask 255.255.255.0 gw 192.168.8.3 dev eth2
route add -net 192.168.7.0 netmask 255.255.255.0 gw 192.168.8.3 dev eth2
route add default gw 192.168.8.111 dev eth2
# Terminamos permitiendo reenviar y enmascarar paquetes TCP/IP para que haga de gateway
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -j MASQUERADE
```

```
[root@linux]# chmod +x /etc/init.d/conf-red
[root@linux]# ln -s /etc/init.d/conf-red /etc/rc2.d/S99conf-red
```

Ej. Práctico 1.2.4: Esquema de Red a partir de la Tabla de Enrutamiento I

Dibuja el esquema de red del equipo/router cuya tabla de enrutamiento se muestra a continuación, indicando la totalidad de las redes. En el caso en que detectes algún error, o regla de enrutamiento que no sea posible indica cual es, y como lo solucionarías para que la tabla de enrutamiento fuera coherente:

```
[root@linux]# route -n
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Iface
192.168.110.0	**0.0.0.0**	**255.255.255.0**	**U**	**eth0**
192.168.120.0	**0.0.0.0**	**255.255.255.0**	**U**	**eth0**
192.168.130.0	**0.0.0.0**	**255.255.255.0**	**U**	**eth1**
192.168.140.0	**0.0.0.0**	**255.255.0.0**	**U**	**eth1**
172.30.0.0	**0.0.0.0**	**255.255.0.0**	**U**	**eth1**
172.40.0.0	**0.0.0.0**	**255.255.0.0**	**U**	**eth2**
12.1.1.0	**192.168.110.1**	**255.0.0.0**	**UG**	**eth0**
12.1.2.0	**192.168.120.1**	**255.255.255.0**	**UG**	**eth1**

12.1.3.0	**192.168.140.1**	**255.255.255.0**	**UG**	**eth1**
12.1.4.0	**172.40.1.100**	**255.255.255.0**	**UG**	**eth2**
0.0.0.0	**172.40.1.150**	**0.0.0.0**	**U**	**eth2**

Solución Ej. Pr. 1.2.4.I.- Esquemas de Red y Tablas de enrutamiento

En primer lugar indicaremos cuales son las reglas que no tienes ningún sentido, y como podrían corregirse. En concreto, dos de las reglas de enrutamiento son imposibles, ya que la máscara que aparece no concuerda con su correspondiente red de destino. Además en otra de las reglas, la interfaz de red no puede ser la indicada, "**eth1**" ya que a través de dicha interfaz no puede comunicarse con el gateway "**192.168.120.1**", ya que la red "**192.168.120.0**" es alcanzable desde la "**eth0**" según la tabla de enrutamiento anterior:

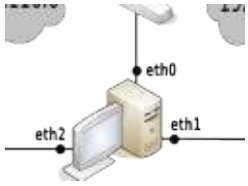
192.168.140.0	**0.0.0.0**	**255.255.0.0**	**U**	**eth1**
12.1.1.0	**192.168.110.1**	**255.0.0.0**	**UG**	**eth0**
12.1.2.0	**192.168.120.1**	**255.255.255.0**	**UG**	**eth1**

Por tanto, una posible solución a los errores anteriores serían:

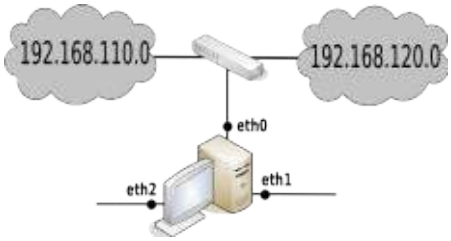
192.168.140.0	**0.0.0.0**	**255.255.255.0**	**U**	**eth1**
12.1.1.0	**192.168.110.1**	**255.255.255.0**	**UG**	**eth0**
12.1.2.0	**192.168.120.1**	**255.255.255.0**	**UG**	**eth0**

En cuanto al esquema de red, para su obtención seguiremos los siguientes pasos:

1) Empezaremos pintándonos el el router en cuestión con sus interfaces de red:

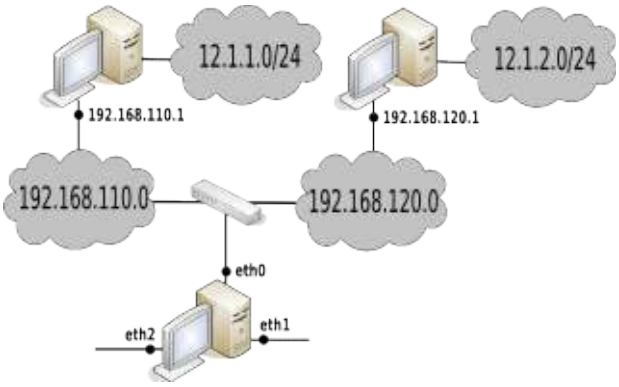


2) A continuación representaremos las redes a las que tiene acceso el router desde su primera interfaz de red eth0 sin necesidad de gateway:

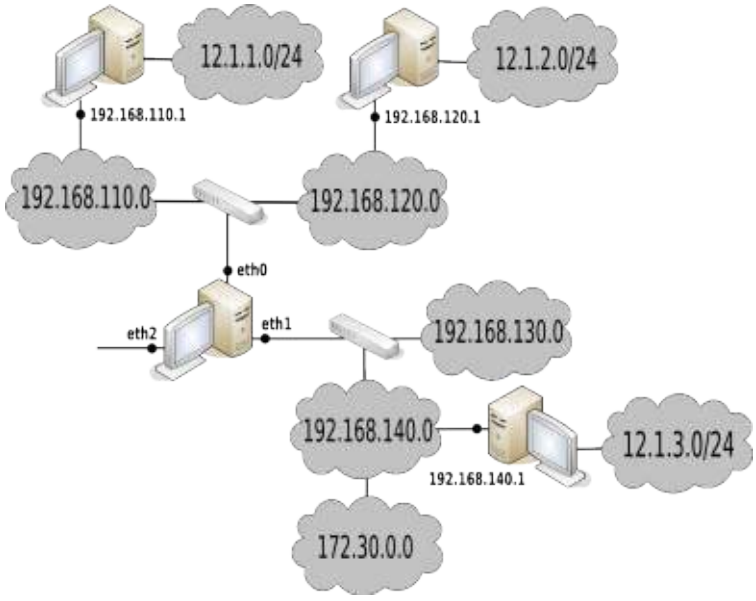


3) Seguiremos representado los gateway que en las redes anteriores, "192.168.110.0" y "192.168.120.0", permiten que el router del cual se muestra su tabla de enrutamiento pueda alcanzar otras redes lógicas del entorno. Si observamos la tabla de enrutamiento anterior, observaremos que

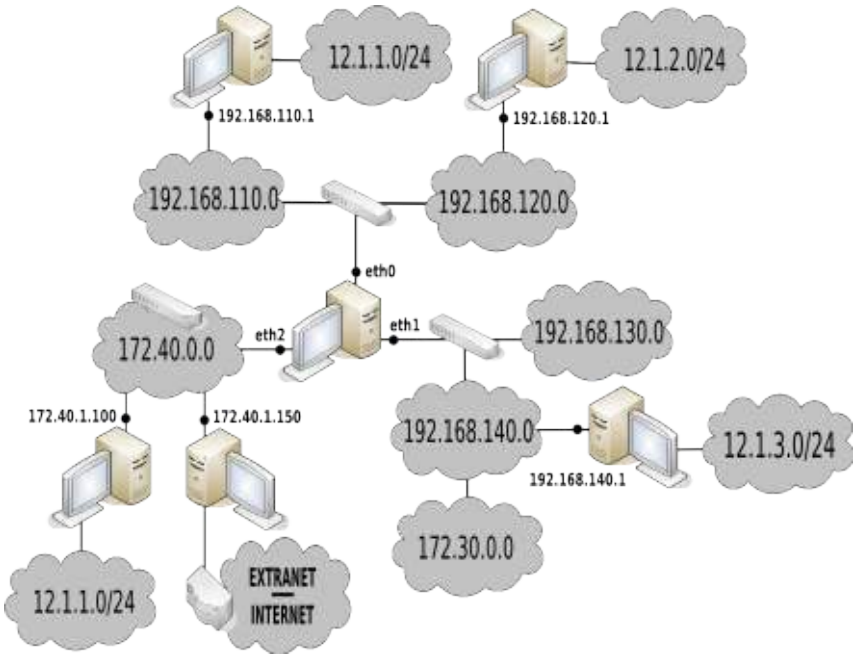
hay dos gateway, el "192.168.110.1" y el "192.168.120.1" (*ya hemos comentado que era una de las erratas realizadas a propósito*), y a través de estos se alcanzan las subredes lógicas "12.1.1.0/24" y "12.1.2.0/24":



4) Continuaremos haciendo lo mismo que hemos hecho con eth0, pero ahora con la interfaz **eth1**: comenzaremos representando las redes lógicas que se alcanzan de manera directa desde dicha interfaz, para a continuación representar los gateway que hay en dichas redes que permiten alcanzar otras redes. En este caso, tras corregir la errata, se advierte que sólo hay un único gateway, el "192.168.140.1" que permite alcanzar la subred "12.1.3.0/24":



5) Y terminaremos haciendo lo mismo de desde eth2, terminando de esta forma un posible esquema de red del router del cual se ha proporcionado su tabla de enrutamiento:



Ej. Práctico 1.2.5: Esquema de Red a partir de la Tabla de Enrutamiento II

Siguiendo los mismos pasos de resolución que en el ejercicio práctico anterior, **dibuja el esquema de red** del equipo/router cuya tabla de enrutamiento se muestra a continuación, indicando la totalidad de las redes. En el caso en que detectes algún error, o regla de enrutamiento que no sea posible indica cual es, y como lo solucionarías para que la tabla de enrutamiento fuera coherente:

```
[root@linux]# route -n
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Iface
192.168.1.0	**0.0.0.0**	**255.255.255.0**	**U**	**eth0**
192.168.2.0	**0.0.0.0**	**255.255.255.0**	**U**	**eth0**
192.168.3.0	**0.0.0.0**	**255.255.255.0**	**U**	**eth1**
12.1.0.0	**0.0.0.0**	**255.255.0.0**	**U**	**eth2**
12.2.0.0	**192.168.1.254**	**255.255.0.0**	**UG**	**eth0**
172.100.0.0	**192.168.1.254**	**255.255.0.0**	**UG**	**eth0**
172.200.0.0	**192.168.3.254**	**255.255.0.0**	**UG**	**eth1**
192.168.6.0	**12.1.1.122**	**255.255.255.0**	**UG**	**eth2**
192.168.7.0	**12.1.1.122**	**255.255.255.0**	**UG**	**eth2**
0.0.0.0	**12.1.1.123**	**0.0.0.0**	**UG**	**eth2**

Ej. Práctico 1.2.6: Esquema de Red a partir de la Tabla de Enrutamiento III

Dibuja el esquema de red del equipo/router cuya tabla de enrutamiento se muestra a continuación, indicando la totalidad de las redes. En el caso en que detectes algún error, o regla de enrutamiento que no sea posible indica cual es, y como lo solucionarías para que la tabla de

enrutamiento fuera coherente:

```
[root@linux]# route -n
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	Iface
192.168.10.0	**0.0.0.0**	**255.255.255.0**	**U**	**eth0**
192.168.20.0	**0.0.0.0**	**255.255.255.0**	**U**	**eth0**
192.168.30.0	**0.0.0.0**	**255.255.255.0**	**U**	**eth0**
12.1.0.0	**0.0.0.0**	**255.255.0.0**	**U**	**eth1**
12.2.0.0	**0.0.0.0**	**255.255.0.0**	**U**	**eth1**
172.100.0.0	**0.0.0.0**	**255.255.0.0**	**U**	**eth2**
172.200.0.0	**0.0.0.0**	**255.255.0.0**	**U**	**eth2**
192.168.60.0	**0.0.0.0**	**255.255.255.0**	**U**	**eth3**
192.168.70.0	**0.0.0.0**	**255.255.255.0**	**U**	**eth3**
172.30.0.0	**192.168.10.200**	**255.255.0.0**	**UG**	**eth0**
172.40.0.0	**192.168.20.200**	**255.255.0.0**	**UG**	**eth0**
172.50.0.0	**192.168.20.201**	**255.255.0.0**	**UG**	**eth0**
172.60.0.0	**192.168.30.100**	**255.255.0.0**	**UG**	**eth0**
172.70.0.0	**12.1.1.100**	**255.255.0.0**	**UG**	**eth1**
172.80.0.0	**12.2.1.100**	**255.255.0.0**	**UG**	**eth1**
12.200.200.0	**12.2.1.200**	**255.255.0.0**	**UG**	**eth1**
12.222.222.0	**172.200.10.10**	**255.255.255.0**	**UG**	**eth2**
192.168.40.0	**192.168.60.200**	**255.255.255.0**	**UG**	**eth3**
192.168.50.0	**192.168.70.200**	**255.255.255.0**	**UG**	**eth3**
192.168.90.0	**192.168.50.200**	**255.255.255.0**	**UG**	**eth3**
0.0.0.0	**172.200.20.22**	**0.0.0.0**	**UG**	**eth2**

Solución Ej. Pr. 1.2.6.I.- Esquemas de Red y Tablas de enrutamiento

En primer lugar indicaremos cuales son las reglas erróneas que no tienen sentido. Comenzaremos con la penúltima regla en la cual se indica que para alcanzar la red "**192.168.90.0**" se hace uso del gateway "**192.168.50.200**", lo cual es imposible, ya que eso implicaría dos saltos, es decir, ese gateway pertenece a la red "**192.168.50.0**" alcanzable a través del gateway "**192.168.70.200**":

192.168.50.0	**192.168.70.200**	**255.255.255.0**	**UG**	**eth3**
192.168.90.0	**192.168.50.200**	**255.255.255.0**	**UG**	**eth3**

Por tanto, la solución es que a través del gateway "**192.168.70.200**" se alcanzan ambas redes:

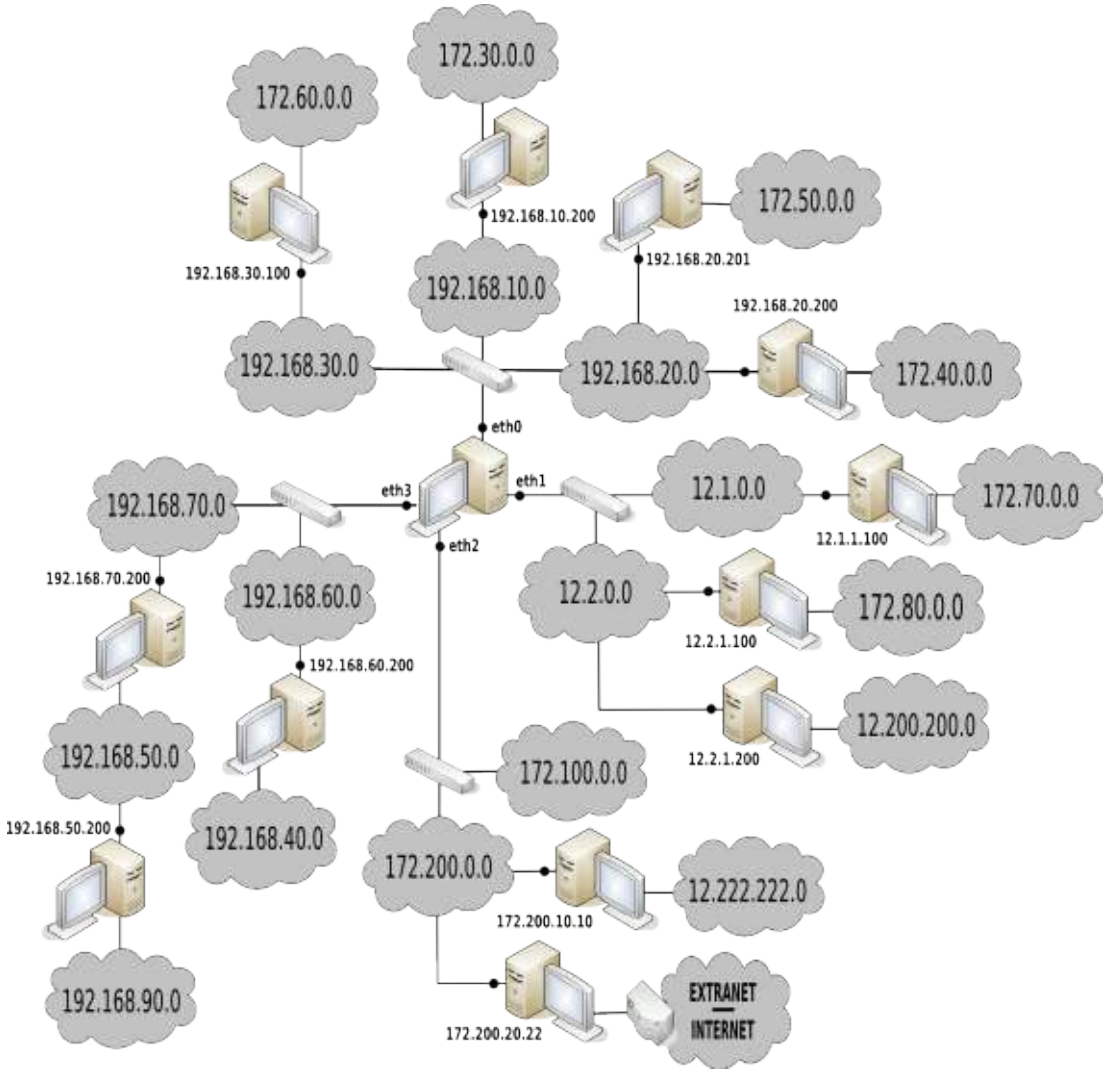
192.168.50.0	**192.168.70.200**	**255.255.255.0**	**UG**	**eth3**
192.168.90.0	**192.168.70.200**	**255.255.255.0**	**UG**	**eth3**

La otra regla que sería errónea sería la siguiente, donde la máscara no concuerda con la

dirección de subred (*red de clase A dividida en subredes, donde se ha hecho uso de las tres primeras cantidades para identificar a la subred*):

12.200.200.0	**12.2.1.200**	**255.255.0.0**	**UG**	**eth1**
12.200.200.0	**12.2.1.200**	**255.255.255.0**	**UG**	**eth1**

En cuanto al esquema de red resultante, siguiendo las pautas explicadas en el ejercicio práctico nº4, sería el siguiente:



Ej. Práctico 1.2.7: Esquema de Red a partir de la Tabla de Enrutamiento IV

Dibuja el esquema de red del equipo/router cuya tabla de enrutamiento se muestra a continuación, indicando la totalidad de las redes. En el caso en que detectes algún error, o alguna regla de enrutamiento que no sea posible indica cual es, y como lo solucionarías para que la tabla de enrutamiento fuera coherente:

```
[root@linux]# route -n
```

Kernel IP routing table

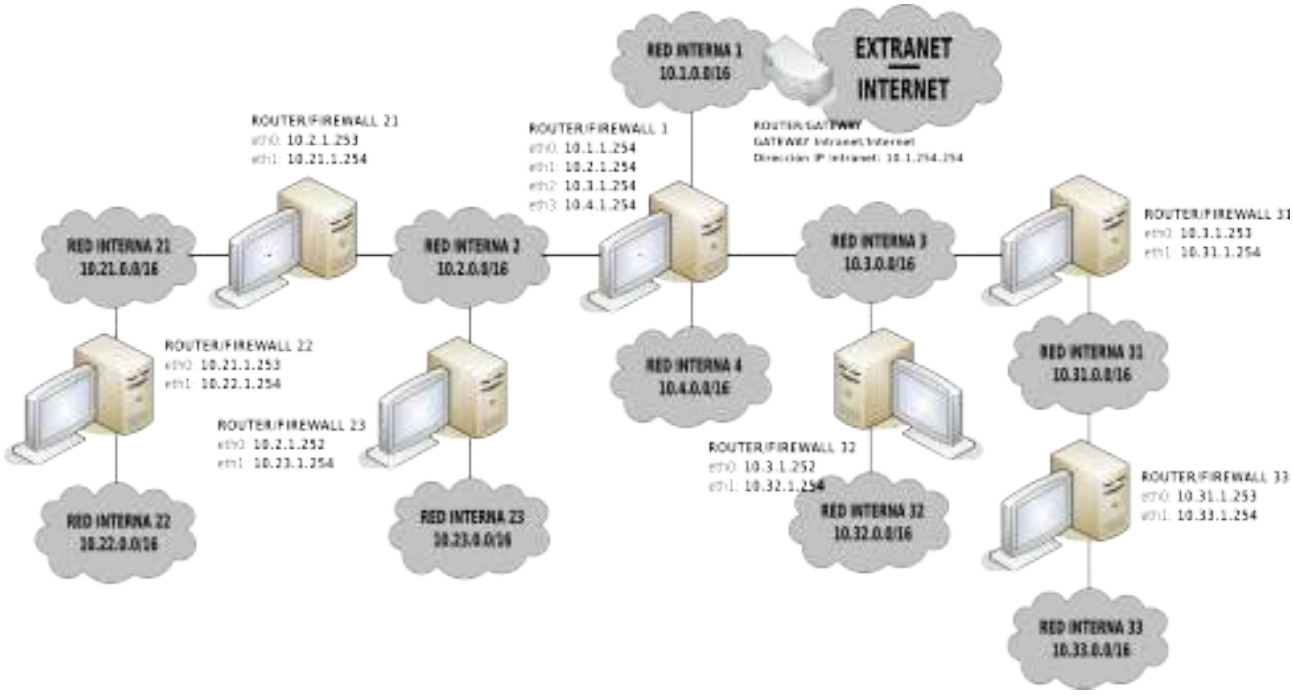
Destination	Gateway	Genmask	Flags	Iface
172.30.1.0	**0.0.0.0**	**255.255.255.0**	**U**	**eth0**
172.30.2.0	**0.0.0.0**	**255.255.255.0**	**U**	**eth0**
172.30.3.0	**0.0.0.0**	**255.255.255.0**	**U**	**eth1**
172.30.4.0	**0.0.0.0**	**255.255.255.0**	**U**	**eth1**
172.30.5.0	**0.0.0.0**	**255.255.255.0**	**U**	**eth1**
172.30.6.0	**0.0.0.0**	**255.255.0.0**	**U**	**eth2**
12.1.0.0	**0.0.0.0**	**255.255.0.0**	**U**	**eth2**
12.2.0.0	**0.0.0.0**	**255.255.0.0**	**U**	**eth3**
12.3.0.0	**0.0.0.0**	**255.255.0.0**	**U**	**eth3**
12.4.0.0	**0.0.0.0**	**255.255.0.0**	**U**	**eth3**
192.168.1.0	**172.30.1.254**	**255.255.255.0**	**UG**	**eth0**
192.168.2.0	**172.30.1.254**	**255.255.255.0**	**UG**	**eth0**
192.168.3.0	**172.30.2.254**	**255.255.255.0**	**UG**	**eth0**
192.168.4.0	**172.30.2.253**	**255.255.255.0**	**UG**	**eth0**
192.168.5.0	**192.168.1.254**	**255.255.255.0**	**UG**	**eth0**
192.168.6.0	**172.30.4.252**	**255.255.255.0**	**UG**	**eth1**
192.168.7.0	**172.30.5.252**	**255.255.255.0**	**UG**	**eth1**
192.168.8.0	**172.30.6.252**	**255.255.255.0**	**UG**	**eth1**
192.168.9.0	**12.1.0.1**	**255.255.255.0**	**UG**	**eth2**
12.5.0.0	**12.2.0.1**	**255.255.0.0**	**UG**	**eth2**
12.6.0.0	**12.4.250.255**	**255.255.0.0**	**UG**	**eth3**
12.7.0.0	**12.3.250.200**	**255.255.0.0**	**UG**	**eth3**
12.8.0.0	**12.3.250.200**	**255.255.255.0**	**UG**	**eth3**
192.168.10.0	**12.4.1.111**	**255.255.255.0**	**UG**	**eth3**
192.168.20.0	**12.4.1.112**	**255.255.255.0**	**UG**	**eth3**
0.0.0.0	**12.1.78.98**	**0.0.0.0**	**UG**	**eth2**

Ej. Práctico 1.2.8: Tabla de Enrutamiento a través de Esquema de Red III

Observa el se esquema de red que muestra a continuación y responde a las siguientes cuestiones prácticas:

A) ¿Cuál sería la **tabla de enrutamiento del router Nº1**, dando por hecho que en su tabla de enrutamiento existen reglas para alcanzar cualquiera de las redes del esquema?

B) ¿Cuál sería el **contenido del script de configuración** que debería ejecutarse en la máquina router Nº1 al iniciarse para que se autoconfigurara correctamente como gateway de la red.



Solución Ej. Pr. 1.2.8.I.- Tabla de Enrutamiento y Script de Configuración de Red

Tras observar el esquema de red, deberíamos concluir que la tabla de enrutamiento del router Nº1 debería ser la siguiente:

Redes Destino	Máscara	Gateway	Interfaz Red
10.1.0.0	255.255.0.0	0.0.0.0	eth0
10.2.0.0	255.255.0.0	0.0.0.0	eth1
10.3.0.0	255.255.0.0	0.0.0.0	eth2
10.4.0.0	255.255.0.0	0.0.0.0	eth3
10.21.0.0	255.255.0.0	10.2.1.253	eth1
10.22.0.0	255.255.0.0	10.2.1.253	eth1
10.23.0.0	255.255.0.0	10.2.1.252	eth1
10.32.0.0	255.255.0.0	10.3.1.252	eth2
10.31.0.0	255.255.0.0	10.3.1.253	eth2
10.33.0.0	255.255.0.0	10.3.1.253	eth2
0.0.0.0	0.0.0.0	10.1.254.254	eth0

Para la configurar la tabla de enrutamiento anterior en el equipo gateway podremos hacer uso de **(1)** los scripts de configuración disponibles en el sistema o **(2)** crear nuestro propio script de configuración:

(1) En el caso de querer hacer uso de los scripts ya proporcionados por el sistema haríamos los

siguiente:

a) Editaríamos el archivo **/etc/network/interfaces** para asignar las direcciones IP a las interfaces de red del equipo, indicar quien es su gateway por defecto, quienes son sus servidores DNS y para añadir las reglas de enrutamiento estático que le informen de como alcanzar todas las redes del entorno:

```
[root@linux]# nano /etc/network/interfaces
# Contenido del archivo interfaces:
auto lo eth0 eth1 eth2 eth3
iface lo inet loopback

iface eth0 inet static
address 10.1.1.254
netmask 255.255.0.0
gateway 10.1.254.254
dns-nameservers 8.8.8.8

iface eth1 inet static
address 10.2.1.254
netmask 255.255.0.0
post-up route add -net 10.21.0.0 netmask 255.255.0.0 gw 10.2.1.253
post-up route add -net 10.22.0.0 netmask 255.255.0.0 gw 10.2.1.253
post-up route add -net 10.23.0.0 netmask 255.255.0.0 gw 10.2.1.252
pre-down route del -net 10.21.0.0 netmask 255.255.0.0 gw 10.2.1.253
pre-down route del -net 10.22.0.0 netmask 255.255.0.0 gw 10.2.1.253
pre-down route del -net 10.23.0.0 netmask 255.255.0.0 gw 10.2.1.252

iface eth2 inet static
address 10.3.1.254
netmask 255.255.0.0
post-up route add -net 10.31.0.0 netmask 255.255.0.0 gw 10.3.1.253
post-up route add -net 10.32.0.0 netmask 255.255.0.0 gw 10.3.1.252
post-up route add -net 10.33.0.0 netmask 255.255.0.0 gw 10.3.1.253
pre-down route del -net 10.31.0.0 netmask 255.255.0.0 gw 10.3.1.253
pre-down route del -net 10.32.0.0 netmask 255.255.0.0 gw 10.3.1.252
pre-down route del -net 10.33.0.0 netmask 255.255.0.0 gw 10.3.1.253

iface eth3 inet static
address 10.4.1.254
netmask 255.255.0.0
```

Para que surta efecto la configuración anterior deberemos reactivar las interfaces de red o directamente reiniciar el equipo (*init 0*):

```
[root@linux]# ifdown -a
[root@linux]# ifup -a
```

b) Tras la configuración IP tan sólo deberemos activar el ip_forwarding para permitir el reenvío de paquetes entre las diferentes interfaces de red, además de activar el enmascaramiento de los paquetes que le atraviesen con la finalidad de que los paquetes de retorno puedan llegar a su destino (*cambia la dirección IP de origen del equipo de la intranet por la dirección IP de la interfaz de salida del router*). Una posibilidad sería añadir los comandos necesarios al script **/etc/init.d/rc.local**, el cual se ejecuta al iniciarse la máquina:

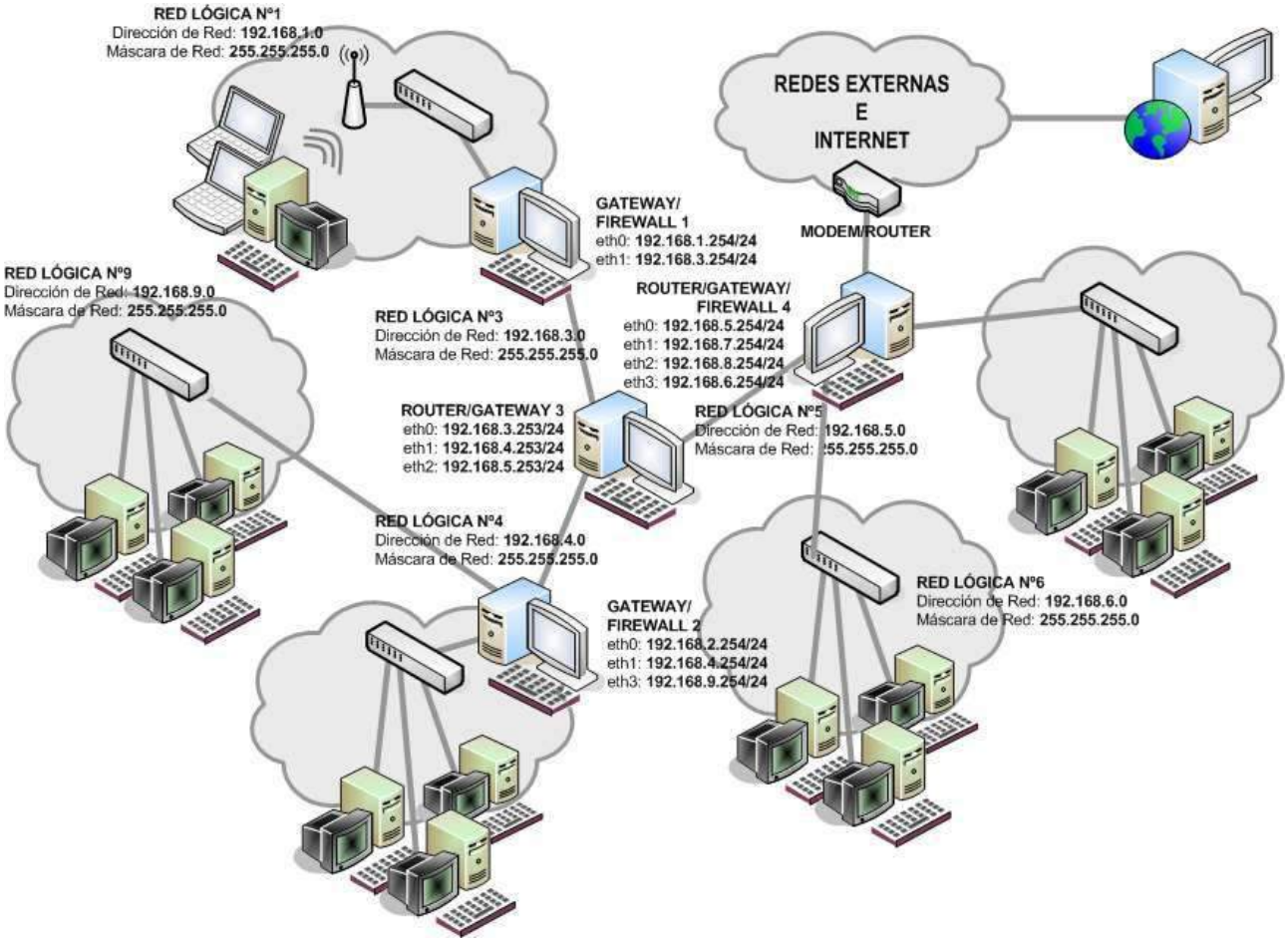
```
[root@linux]# echo 'echo "1" > /proc/sys/net/ipv4/ip_forward' >> /etc/init.d/rc.local
[root@linux]# echo 'iptables -t nat -A POSTROUTING -j MASQUERADE' >> /etc/init.d/rc.local
```

(2) En relación al script de configuración, tal como se ha mostrado en ejercicios anteriores, debería tener el siguiente aspecto:

```
[root@linux]# nano /etc/init.d/conf-red

#!/bin/bash
/etc/init.d/networking stop
ifconfig eth0 10.1.1.254 netmask 255.255.0.0
ifconfig eth1 10.2.1.254 netmask 255.255.0.0
ifconfig eth2 10.3.1.254 netmask 255.255.0.0
ifconfig eth3 10.4.1.254 netmask 255.255.0.0
route add -net 10.21.0.0 netmask 255.255.0.0 gw 10.2.1.253 dev eth1
route add -net 10.22.0.0 netmask 255.255.0.0 gw 10.2.1.253 dev eth1
route add -net 10.23.0.0 netmask 255.255.0.0 gw 10.2.1.252 dev eth1
route add -net 10.31.0.0 netmask 255.255.0.0 gw 10.3.1.253 dev eth2
route add -net 10.32.0.0 netmask 255.255.0.0 gw 10.3.1.252 dev eth2
route add -net 10.33.0.0 netmask 255.255.0.0 gw 10.3.1.253 dev eth2
route add default gw 10.1.254.254
echo "nameserver 8.8.8.8" > /etc/resolv.conf
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -j MASQUERADE
```

Ej. Práctico 1.2.9: Tabla de enrutamiento a través de Esquema de Red IV

Si tenemos en cuenta que las direcciones IP privada y pública del router son 192.168.8.200 y 68.123.27.231, **indica como serían las tablas de enrutamiento** de los equipos "**ROUTER/GATEWAY/FIREWALL 3**" y "**ROUTER/GATEWAY/FIREWALL 4**", teniendo en cuenta que deben poder alcanzar cualquier equipo de la Intranet e Internet. De igual forma, **indica los comandos que serían necesarios ejecutar** en cada uno de ellos (*ifconfig, route e iptables*).



1.3.- Sistemas de Virtualización

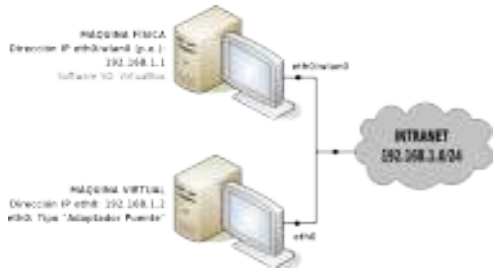
A lo largo del curso, por cuestiones de sencillez en su implementación y requisitos hardware, usaremos la herramienta software de virtualización **Virtualbox** para diseñar la mayoría de los sistemas de seguridad perimetral (*diseño de firewall y proxy*), aunque para la implementación de servicios en servidores (*HTTP/HTTPS, FTP/FTPS, SMTP, VPN, etc.*), por cuestiones de eficiencia sería más conveniente hacer uso de herramientas software como **XenServer** (*se ha convertido actualmente en un proyecto de código abierto, al igual que el proyecto Xen*), **proXmoX** o **Vmware**.

En relación a Virtualbox cabría recordar que pueden configurarse hasta cuatro interfaces red por máquina virtual, pudiendo seleccionarse entre uno de los siguientes tipos:

- **Adaptador puente** (*modo bridge*): este modo nos resultará útil cuando deseemos que nuestra máquina virtual forme parte de la misma red de área local que nuestro equipo físico. Es decir, en este modo de configuración de red, la máquina virtual hace uso de la interfaz de red real del equipo físico, quitándole parte de su ancho de banda, pudiendo agregar la máquina virtual a la misma red lógica en el caso que se le asigne una dirección IP del mismo rango que la que el equipo físico tiene, recibiendo la misma puerta de enlace y direcciones de los DNS.

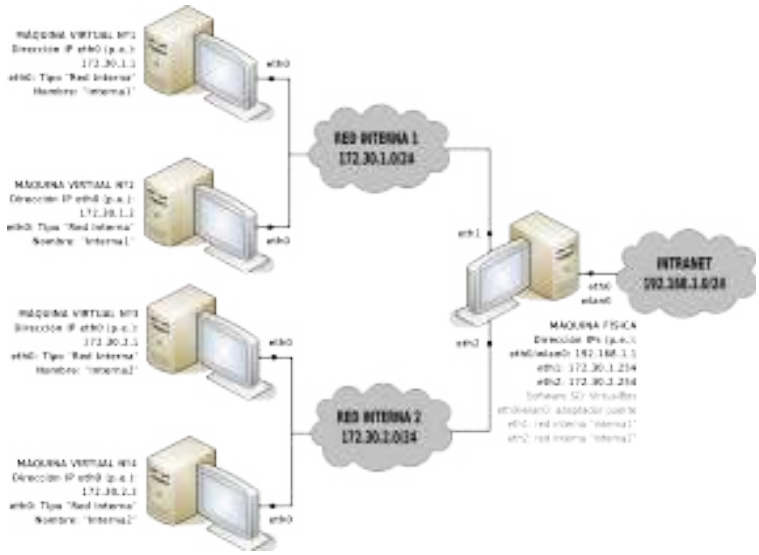
En el caso de que nuestro equipo físico disponga de más de una interfaz de red, deberemos

decidir sobre cual de ellas establecer el modo bridge, siendo este aspecto totalmente transparente para la máquina virtual. Por ejemplo, si nuestro equipo físico es un portátil y dispone de una interfaz cableada (*eth0*), y una interfaz inalámbrica (*wlan0*), deberemos seleccionar aquella por la que este saliendo al exterior nuestra máquina física, e independientemente cual de ellas acabe siendo seleccionada, de cara a la máquina virtual será detectada una única interfaz de red *eth0* como si esta fuera cableada.



- **NAT** (*Network Address Translation*): en este modo, Virtualbox crea internamente mediante software un router que hace de intermediario entre la máquina física y la virtual, permitiendo a ésta última salir hacia el exterior a través de la primera haciendo uso de una IP privada suministrada dinámicamente por dicho router virtual. Esta opción se caracteriza por garantizar salida al exterior a la máquina virtual sin que el usuario tenga que llevar a cabo ningún tipo de configuración, pero con el inconveniente, de que esta no es accesible directamente desde la máquina física.

- **Red Interna**: Opción muy interesante que nos permite formar varias Intranets independientes dentro de nuestro equipo entre las distintas máquinas virtuales que se encuentren en ejecución. Para posibilitar el acceso al exterior de aquellas máquinas que dispongan únicamente de este tipo de interfaz, será necesario configurar nuestra máquina física (*equipo anfitrión*) como gateway/router, configuración muy común en los esquemas de red que se implementarán en las prácticas que en este manual se presentarán.

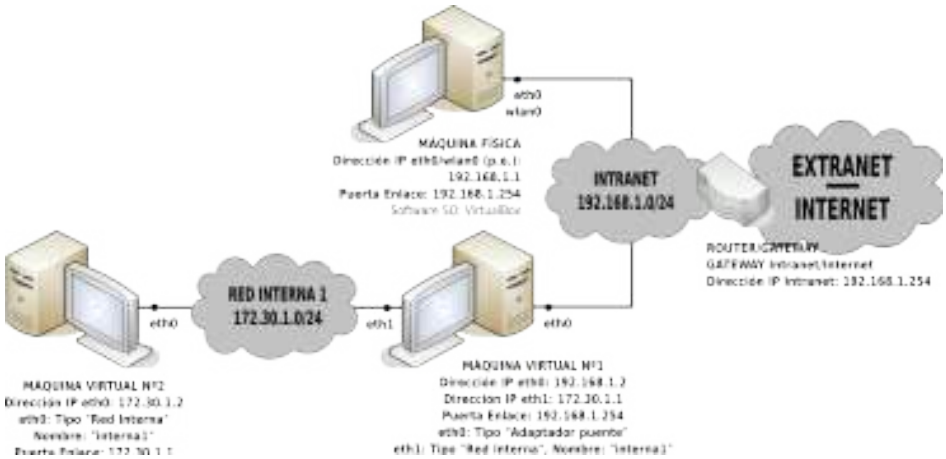


- **Adaptador Sólo Anfitrión**: Opción que permite establecer una red privada interna formada únicamente por la máquina virtual y el equipo físico. Por tanto, mediante este tipo de interfaz dos máquinas virtuales no pueden comunicarse de manera directa, y no tienen acceso al exterior, a no ser que configuremos al equipo físico como gateway.

Con la finalidad de llevar a la practica entornos de red de manera virtualizada, a continuación se propondrán diversos escenarios.

Ej. Práctico 1.3.1: Configuración de entorno de red con 1 Gateway

Implementa el siguiente entorno de red formado por tres equipos (*nuestro equipo físico, más dos máquinas virtuales*), donde una de ellas hará las funciones de gateway (*máquina virtual Nº1*), permitiendo tráfico en ambos sentidos, desde la Intranet privada (*red interna 1*) hacia el exterior, y desde el exterior hacia la Intranet (*las direcciones IP indicadas son tan sólo un ejemplo, pueden sustituirse por aquellas que se considere conveniente*), y la otra las funciones de servidor (*zona desmilitarizada, DMZ*).



Las características que tendrán las máquinas virtuales serán las siguientes:

- Sistema operativo: Ubuntu Server 12.04 (*p.e.*). La máquina virtual Nº1 hará de gateway/firewall, mientras la máquina virtual Nº2 ofrecerá un servicio HTTP:8000 hacia el exterior.

Solución Ej. Pr. 1.3.1.I.- Virtualización de Entornos de Red

Antes de dar la solución al ejercicio, se recomienda a aquellos que no tengan conocimientos sobre previos sobre NAT o forwarding, que consulten la introducción a la segunda práctica que se plantea en este manual de prácticas.

Comenzaremos configurando el entorno de red en ambas máquinas virtuales. En concreto, realizaremos un script que la configure a nuestro gusto al iniciar la máquina. Para ello:

1) Comenzaremos editando el script (*llamado por ejemplo "conf-red.sh"*) en la máquina virtual Nº1. Para que surta efecto deberemos tener en cuenta el tipo de interfaz de red configurado a través de VirtualBox en la máquina virtual Nº1. En concreto, tal como se puede observar en el esquema anterior, su primera interfaz deberá ser de tipo "**Adaptador Puente**" para garantizar un contacto con la red física externa, y su segunda interfaz deberá ser de tipo "**Red Interna**" con la finalidad de formar una Intranet:

¡¡Importante!! A la hora de configurar el entorno de red de nuestro gateway (p.e. Ubuntu Server) tenemos al menos dos posibles opciones de configuración: **(1)** modificando los scripts disponibles de sistema o **(2)** crear nuestro propio script (*solución adoptada en la resolución del ejercicio*). A modo de ejemplo, a continuación se mostrará como configurar la máquina virtual nº1 "mv1" haciendo uso de los scripts disponibles en el sistema **/etc/network/interfaces** y **/etc/init.d/rc.local**:

Para la configurar la tabla de enrutamiento anterior en el equipo gateway podremos hacer uso de **(1)** los scripts de configuración disponibles en el sistema o **(2)** crear nuestro propio script de configuración (*solución adoptada como solución del ejercicio*):

(1) En el caso de querer hacer uso de los scripts ya proporcionados por el sistema haríamos los siguiente:

a) Editaríamos el archivo **/etc/network/interfaces** para asignar las direcciones IP a las interfaces de red del equipo, indicar quien es su gateway por defecto, quienes son sus servidores DNS y para añadir las reglas de enrutamiento estático que le informen de como alcanzar todas las redes del entorno:

```
[root@linux]# nano /etc/network/interfaces
# Contenido del archivo interfaces de la máquina virtual nº1 mv1:
auto lo eth0 eth1
iface lo inet loopback

iface eth0 inet static
address 192.168.1.2
netmask 255.255.255.0
gateway 192.168.1.254
```

```
dns-nameservers 8.8.8.8

iface eth1 inet static
address  172.30.1.1
netmask 255.255.255.0
```

Para que surta efecto la configuración anterior deberemos reactivar las interfaces de red o directamente reiniciar el equipo (*init 0*):

```
[root@linux]# ifdown -a
[root@linux]# ifup -a
```

b) Tras la configuración IP tan sólo deberemos activar el ip_forwarding para permitir el reenvío de paquetes entre las diferentes interfaces de red, además de activar el enmascaramiento de los paquetes que le atraviesen con la finalidad de que los paquetes de retorno puedan llegar a su destino (*cambia la dirección IP de origen del equipo de la intranet por la dirección IP de la interfaz de salida del router*). Una posibilidad sería añadir los comandos necesarios al script **/etc/init.d/rc.local**, el cual se ejecuta al iniciarse la máquina:

```
[root@linux]# echo 'echo "1" > /proc/sys/net/ipv4/ip_forward' >> /etc/init.d/rc.local
[root@linux]# echo 'iptables -t nat -A POSTROUTING -j MASQUERADE' >> /etc/init.d/rc.local
[root@linux]# echo 'iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 8000 -j DNAT --to 172.30.1.2' >> /etc/init.d/rc.local
```

```
[root@mv1]# nano /etc/init.d/conf-red.sh
#!/bin/bash
# Como Ubuntu Server no dispone de interfaz gráfica, el servicio que se encarga de gestionar de manera automática la red es "networking". Por tanto, en primer lugar pararemos este servicio para que no entre en conflicto con nuestra configuración
/etc/init.d/networking stop
# Pasamos a configurar las direcciones IP del equipo, puerta de enlace y DNS
ifconfig eth0 192.168.1.2
ifconfig eth1 172.30.1.1 netmask 255.255.255.0
route add default gw 192.168.1.254
echo "nameserver 8.8.8.8" > /etc/resolv.conf
# Activamos el forwarding y el enmascaramiento de paquetes para que pueda realizar correctamente su función como gateway
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
# Redireccionamos las peticiones recibidas desde el exterior (eth0) al servicio HTTP:8000 hacia el equipo servidor
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 8000 -j DNAT --to 172.30.1.2
```

2) A continuación, crearemos el script de configuración en la segunda máquina virtual, teniendo en cuenta que esta máquina no es un gateway/router/firewall, sino un servidor, por lo que no es necesario configurar ninguna NAT postrouting o prerouting:

```
[root@mv2]# nano /etc/init.d/conf-red.sh
#!/bin/bash
/etc/init.d/networking stop
ifconfig eth0 172.30.1.2 netmask 255.255.255.0
route add default gw  172.30.1.1
echo "nameserver 8.8.8.8" > /etc/resolv.conf
```

3) Daremos permiso de ejecución al script anterior:

```
[root@mv1|2]# chmod +x /etc/init.d/conf-red.sh
```

4) Teniendo en cuenta que el script "/etc/init.d/rc.local" se ejecuta en último lugar al iniciarse la máquina (*podemos comprobarlo al listar los servicios que se inician en nuestro modo de arranque*), añadiremos una última línea al final de dicho script (*doble redireccionamiento, >>*) para que se configure la red en base a nuestro script:

```
[root@mv1|2]# echo "bash /etc/init.d/conf-red.sh" >> /etc/init.d/rc.local
```

5) Si reiniciamos la máquina (*init 6*) podremos comprobar que se configura perfectamente la red, con lo cual ya podremos probar que la máquina virtual Nº2 sale a Internet a través de la máquina virtual Nº1 (*gateway*).

6) Por último, tan sólo nos queda comprobar que desde el exterior se accede al servicio HTTP:8000 ofrecido por el equipo servidor a través de la máquina virtual Nº1 que hay de router. Para ello, en la máquina virtual Nº2 crearemos el servicio de una manera muy simple mediante python mediante el módulo **SimpleHTTPServer** (*SimpleHTTPServer, distingue entre mayúsculas y minúsculas*) el cual sirve los documentos ubicados en el directorio desde el cual se ejecuta el comando por el puerto que se le indique:

```
[root@mv2]# mkdir /mnt/compartir
[root@mv2]# touch /mnt/compartir/archivo1.pdf /mnt/compartir/archivo2.pdf
[root@mv2]# chmod -R +rx /mnt/compartir
[root@mv2]# cd /mnt/compartir
[root@mv2]:/mnt/compartir# python -m SimpleHTTPServer 8000
```

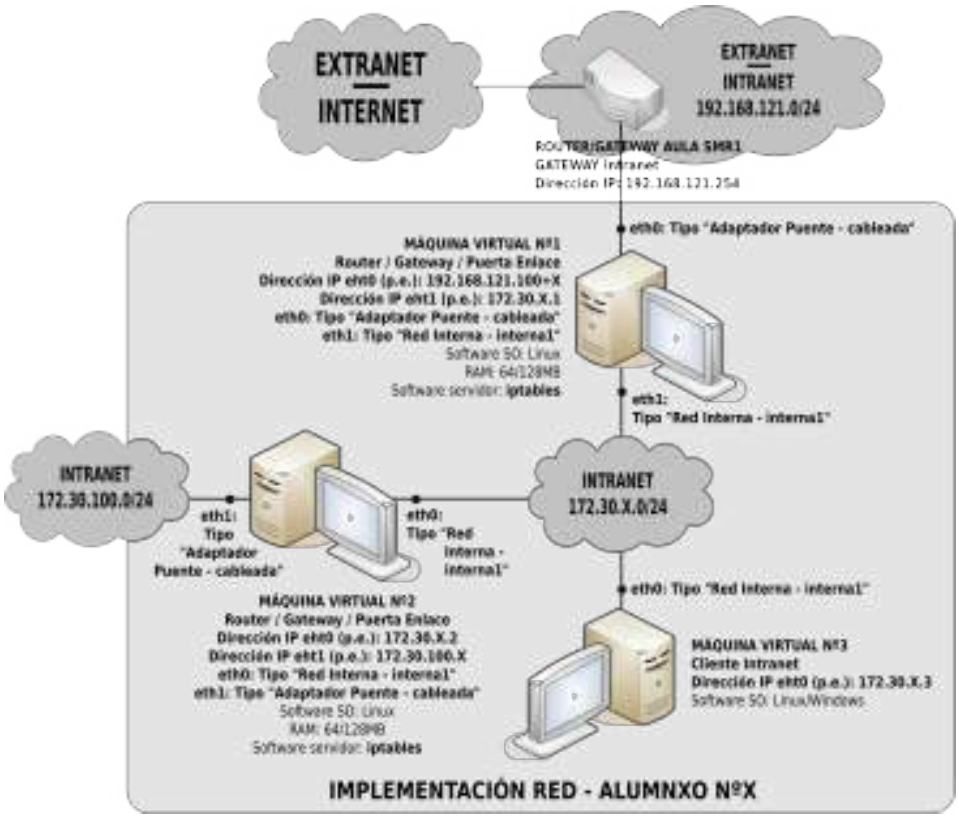
7) Para terminar, desde una máquina externa, por ejemplo la máquina física comprobaremos el acceso al servicio anterior desde un cliente web (*p.e. Chromium, o mozilla*) poniendo en la barra de direcciones del navegador la dirección IP del equipo gateway y el puerto de servicio: "**http://192.168.1.2:8000**". La máquina virtual Nº1 al recibir la petición, tal como se le ha programado a través del anterior NAT prerouting reenviará la petición a la máquina virtual Nº2.

Ej. Práctico 1.3.2: Configuración de entorno de red con 2 Gateways

Siguiendo la línea del ejercicio anterior, implementa el siguiente esquema de red compuesto por tres máquinas virtuales, de las cuales dos de ellas **serán dos equipos GNU/Linux que actuaran como Router/Gateway/Puerta de enlace**. **La tercera máquina virtual será un equipo**

cliente Windows/Linux, que accederá al resto de redes (*Internet y redes privadas del resto de alumnos*), haciendo uso de las máquinas virtuales que hacen de routers (*Las redes lógicas y direcciones IP indicadas son tan sólo un ejemplo. En caso de ser necesario pueden modificarse*).

Advertir a través del esquema, que la subred **172.30.100.0/24** es común a todos los usuarios, y a través de ella, mediante el gateway **172.30.X.2** que cada uno tiene, la tercera máquina virtual puede comunicarse con el resto de equipos del compañero.



Solución Ej. Pr. 1.3.2.I.- Virtualización de Entornos de Red

Para configurar las interfaces de red de los equipos Linux y su tabla de enrutamiento, al igual que en el ejercicio anterior, puede implementarse un script o hacer uso de los siguientes comandos (*p.e. para el primer router*):

```
[root@linux]# ifconfig eth0 192.168.121.100+X
[root@linux]# ifconfig eth1 172.30.X.1
```

Para configurar la puerta de enlace puede hacerse a través de cualquiera de los dos siguientes comandos (*son equivalentes*):

```
[root@linux]# route add -net 0.0.0.0 netmask 0.0.0.0 gw 192.168.121.254 dev eth0
[root@linux]# route add default gw 192.168.121.254
```

Además, recordar tal como vimos en el ejercicio práctico anterior que para configurar un

Linux como router/gateway tan sólo son necesarios dos comandos (*el tercero sólo será necesario si se ha implementado a parte un filtrado de tipo DROP por defecto a los paquetes que le atraviesan*):

```
[root@linux]# echo "1" > /proc/sys/net/ipv4/ip_forward
[root@linux]# iptables -t nat -A POSTROUTING -j MASQUERADE
[root@linux]# iptables -t filter -A FORWARD -m state --state ESTABLISHED,RELATED \
-j ACCEPT
```

Para configurar la máquina virtual Nº2 haremos lo mismo que con la Nº1.

Comprueba, tras haber configurado el equipo cliente Windows/Linux, que este puede navegar por Internet haciendo uso de la máquina virtual Nº1 de puerta de enlace.

Por último, para que la máquina virtual Nº3 acceda a las Intranet de los compañeros haciendo uso de la segunda máquina virtual como puerta de acceso, será necesario añadir las correspondientes reglas de enrutamiento estáticas. Si la maquina Nº3 es un Linux (*"Y" representa el número de un compañero*):

```
[root@linux]# route add -net 172.30.Y.0 netmask 255.255.255.0 gw 172.30.X.2
```

En el caso de que sea un Windows, ejecutaremos el siguiente comando (*la opción "-p" nos permitirá que la regla sea persistente*):

```
[root@windows]# route -p add 172.30.Y.0 mask 255.255.255.0 172.30.X.2
```

Práctica Nº2.- Seguridad Perimetral. Diseño de Firewalls. Iptables y Zentyal

La interconexión global de todas las redes informáticas a través de Internet, se ha traducido simultáneamente en una inagotable fuente de servicios (mensajería electrónica, transferencia de archivos, control remoto de equipos, etc.), pero también en un agujero permanente de seguridad.

En este capítulo vamos a centrarnos esencialmente en dos aspectos fundamentales:

1. Cómo habilitar el acceso a los servicios y recursos que se ofrecen por las distintas redes, que se encuentran conectadas a Internet, en función de donde se localice el origen y el destino ⇒ NAT.

El problema de establecer una comunicación entre dos equipos informáticos, surge en el momento en que uno de los implicados (*emisor-receptor, origen-destino*) pertenece a una red lógica privada, y trata de identificarse mediante una dirección IP privada únicamente reconocida dentro de la propia Intranet a la que pertenece. Es decir, el problema sería equivalente a decir que deseamos poner en comunicación dos usuarios mediante el tradicional correo postal, y alguno de ellos, no tiene asignada una dirección postal pública, y por tanto reconocida por el servicio postal, lo cual derivaría en que no habría forma de que las cartas repartidas por carteros pudiesen alcanzar su correspondiente destino, o remitente en caso de devolución (los paquetes serían descartados). Según esto, sólo cuando los dos extremos de la comunicación tengan direcciones reconocidas (*direcciones IP públicas*) podrán establecer comunicación sin ningún tipo de problemas, y cuando esto no se cumpla, necesitaremos de una estrategia denominada NAT.

2. Cómo filtrar las comunicaciones que se establecen a través de Internet, pudiendo decidir qué servicios son accesibles y cuáles no, con la única pretensión de proteger nuestra red (*integridad y confidencialidad*). A esto se lo denomina filtrado (*filter*).

Toda red informática conectada a Internet esta expuesta a posibles ataques que pongan en jaque tanto los datos e información confidencial que podamos almacenar (troyanos, sniffing, spoofing, hijacking, etc.), como el correcto funcionamiento (virus, gusanos, applets java, etc.) de los equipos que la forman. Con la finalidad de salvaguardar nuestra red, una posible soluciñon es filtrar toda comunicación que se establece tanto desde nuestra red, como hacia ella.

2.1.- Firewall GNU/Linux: IPTABLES

Para convertir nuestro equipo GNU/Linux en un equipo firewall será necesario configurar un paquete software que se encuentra integrado con el propio kernel del sistema operativo GNU/Linux, lo cual va a garantizar una rápida actuación sobre las decisiones de las interfaces de red nuestro equipo, permitiéndonos decidir que paquetes TCP/IP aceptar y cuales denegar, además de poder alterar datos tan importantes de la cabecera de los paquetes TCP/IP como las direcciones IP de origen o destino, o puertos de comunicación: **iptables**.

En concreto, este software nos permitirá filtrar los paquetes TCP/IP en los niveles de red y transporte (*capas 3 y 4*) del modelo de referencia TCP/IP. Es decir, esencialmente podremos filtrar en función de quien sea el origen o destino (*dirección IP de origen o destino ⇒ capa 3/ nivel red*), del protocolo de comunicación (*tcp/udp/icmp ⇒ capa 4 / nivel transporte*) y el puerto de comunicación (*21/ftp, 53/dns, 80/http, 137:139/netbios, 443/https, 10000/webmin, etc. ⇒ capa 4 / nivel transporte*). Todo ello se explicará más detenidamente en la siguiente sección.

2.2.- Alteración de las direcciones IP del paquete TCP/IP mediante iptables: NAT.

Además iptables nos va a permitir llevar a cabo **NAT** y mapeo de puertos (*mapping*): **Network Address Translation**. Mediante NAT se nos permite realizar alteraciones en las direcciones IP de origen (*SNAT, Source Network Address Translation*) y destino (*DNAT, Destination Network Address Translation*) de los paquetes TCP/IP. Es decir, si observáramos la estructura de los paquetes TCP/IP que viajan por la red, advertiríamos que tienen la siguiente estructura: cabecera (*header*) y datos. En la cabecera se encuentra la información referente a las direcciones y puertos de comunicación de origen y destino. Es decir, su estructura es similar a la carta tradicional, compuesta por un sobre (*cabecera*), donde se informa de la dirección postal del destinatario y remitente, y un contenido (*datos*). La finalidad de NAT es poder alterar las direcciones IP y puertos de comunicaciones de origen y destino con el único objetivo de que la comunicación pueda establecerse.

Paquete TCP/IP

A) **SNAT** (*Source NAT*): La alteración de la dirección IP de origen suele llevarse a cabo por parte de los equipos que hacen de gateway entre redes, donde intervienen direcciones IP de tipo privado.

Para comprender SNAT supondremos encontrarnos en la siguiente situación: un equipo que se encuentra dentro de una Intranet (*red privada*), que desea conectarse a Internet a través del modem-router suministrado por un proveedor de servicios de Internet. En este supuesto, el equipo que quiere establecer conexión con un servidor cualquiera de la Internet *(por ejemplo, www.google.com*) dispone de una dirección IP privada únicamente reconocida dentro de la Intranet donde se encuentra. Esto significa que en el momento en que le llegase la petición emitida por este al equipo servidor de la Internet y observase el remitente no reconocería la dirección IP, y por tanto, no sabría contestarle.

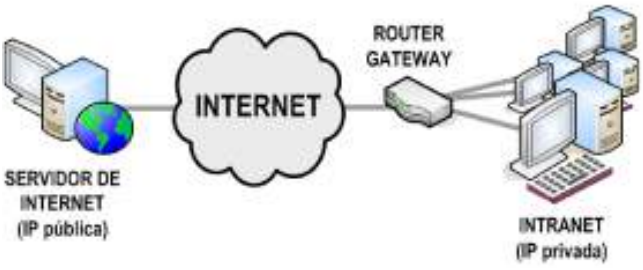
Para solucionar este problema, debemos observar que el único equipo dentro de nuestra red que dispone una dirección IP pública, y por tanto, reconocida en Internet es el router/gateway a través del cual nos conectamos, lo que implica que todos los paquetes TCP/IP que atraviesan el router procedentes de la Intranet deben ser alterados por este, intercambiando la dirección IP de

origen privada por su dirección IP pública (*SNAT*), de tal forma que al recibir los paquetes el servidor de Internet destinatario, pueda reconocer la dirección IP del remitente y así poderle contestar. A esta modificación de los paquetes TCP/IP se le denomina también enmascaramiento, **MASQUERADE**.

En definitiva, nuestro modem/router/gateway es un dispositivo que al menos dispone de dos interfaces de comunicación, una pública y una privada, que le permite hacer de intermediario entre la Intranet y la Internet, separando desde el punto de vista lógico ambas redes.

En el caso de implementar SNAT a través de iptables, este nos va a permitir decidir que paquetes alterar en función de:

- La interfaz de red por donde vayan a salir los paquetes TCP/IP (*opción -o, output*) tras su enrutamiento (POSTROUTING).
- Quién sea el equipo origen (*opción -s, source, dirección IP de origen*).
- Quién sea el equipo destino (opción -d, destination, dirección IP de destino).
- El protocolo de comunicación utilizado (*opción -p, protocol*), o el servicio o puerto de comunicaciones al que se desea acceder (*opción --dport, destination port*), permitiéndonos decidir por que dirección IP intercambiar la dirección IP de origen (*opción -j SNAT --to*).



SNAT:

1. Un equipo de la Intranet (direcciones IP privadas) genera un paquete destinado hacia Internet.
2. Al ser el destinatario del paquete TCP/IP un equipo externo a la red lógica interna es enviado al `gateway/router` para que este decida que hacer con él.
3. Al recibir el `router` el paquete TCP/IP, lo enruta hacia Internet, pero justo antes de expulsar el paquete altera el paquete enmascarando la dirección IP de origen (SNAT) intercambiándola por su dirección IP pública (SNAT POSTROUTING, posterior al enrutamiento, se altera el paquete).
4. El servidor de la Internet responde a la petición contestando con un nuevo paquete TCP/IP dirigido a la dirección IP pública, y por tanto, reconocida del `router`. El `router` al recibir la respuesta desenmascara el paquete y se lo entrega al equipo de la intranet que llevó a cabo la petición.

En cuanto a la sintaxis del comando SNAT, de manera genral sería la siguiente, aunque lógicamente muchos de los parámetros son opcionales, pudiéndose omitir (*opción -A ⇒ "Añadir regla a la lista POSTROUTING SNAT"*):

```
[root@linux] # iptables -t nat -A POSTROUTING -o 'interfaz red' \
                -s 'dirección IP origen' -d 'dirección IP destino' \
                -p 'protocolo' --dport 'puerto de destino' -j SNAT --to 'nueva IP de origen'
```

A modo de ejeplo, si nuestro router tuviera la dirección IP 43.26.112.56 asociada la interfaz de red de salida, el comando podría ser la siguiente:

```
[root@linux] # iptables -t nat -A POSTROUTING -o eth0 \
               -s 192.168.1.0/24 -d www.google.com \
               -p tcp --dport 80 -j SNAT --to 43.26.112.56

[root@linux] # iptables -t nat -A POSTROUTING -o eth0 \
               -s 192.168.1.0/24 -d www.google.com \
               -p tcp --dport 80 -j MASQUERADE
```

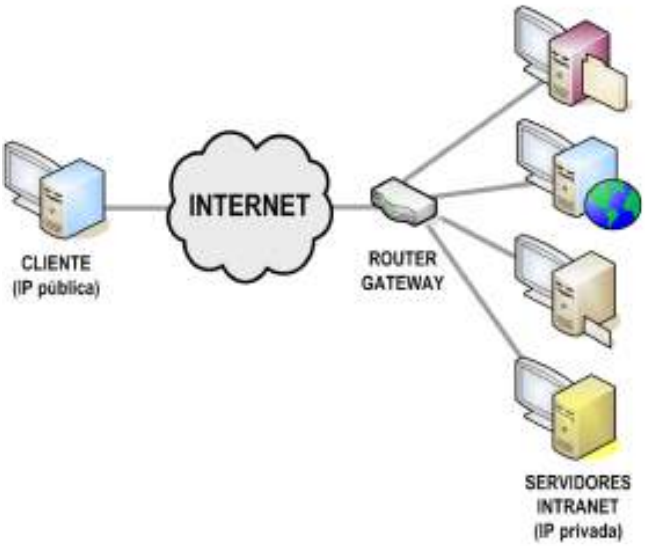
Si quisiéramos enmascarar con carácter general, independientemente del destino, origen y protocolo de destino:

```
[root@linux] # iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

B) **DNAT** (*Destination NAT*): La alteración de la dirección IP de destino suele llevarse a cabo por parte de los equipos que hacen de gateway en el momento en que este se encarga de encubrir los servicios proporcionados por una red con direcciones IP privadas (intranet).

Para comprender DNAT supondremos encontrarnos en la siguiente situación: un equipo conectado a Internet desea acceder a un servicio ofrecido por otro equipo ubicado dentro de una Intranet. En este supuesto el equipo servidor al pertenecer a una red interna privada tiene asignada una dirección IP no reconocida en Internet, y por tanto, inaccesible.

Esto se traduce en que la única manera de que este servidor pueda recibir solicitudes de conexión es a través de un intermediario que se encargue de recoger las solicitudes emitidas desde cualquier punto de la Internet (mediante una dirección IP pública), y que posteriormente se las entregue a este (que posee una dirección IP privada). Este dispositivo intermediario no es otro más que el router/gateway.



DNAT:

1. Un equipo conectado a Internet realiza una solicitud a un servicio ofrecido por una Intranet (dirección IP privada ⇒ inaccesible), colocando como destinatario del paquete TCP/IP la dirección IP pública del `router/gateway` que hace de intermediario.
2. El dispositivo `router/gateway` recibe la solicitud y reconoce el servicio al que desea conectarse a través del puerto de comunicación de destino.
3. Intercambia la dirección IP pública de destino (DNAT) por la dirección IP privada dentro de la Internet correspondiente al equipo que ofrece el servicio solicitado. Una vez alterado (PREROUTING) el paquete TCP/IP es enrutado por el `router/gateway` hacia la interfaz de red que le permite a éste establecer comunicación con el servidor privado (DNAT PREROUTING, previo al enrutamiento se realiza la alteración del paquete TCP/IP).
4. El servidor de la intranet recibe la solicitud y emite una respuesta entregándosela nuevamente al `router/gateway` para que se la haga llegar al equipo de la Internet que inicio la comunicación.

En cuanto a la sintaxis y opciones de iptables a la hora de llevar a cabo una DNAT, son similares a las vistas en SNAT.

Por ejemplo, si quisiéramos que aquellos paquetes recibidos por nuestro router/gateway a través de su interfaz de red eth0 con protocolo tcp/udp destinados al puerto de comunicaciones 8080 fuesen alterados con la finalidad de intercambiar la dirección IP de destino (*la de su interfaz de entrada*) por la dirección IP del servidor FTP de la Intranet (*por ejemplo, 192.168.1.5*) deberíamos ejecutar los siguientes comandos:

```
[root@linux] # iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 8080 \
               -j DNAT --to 192.168.1.5
[root@linux] # iptables -t nat -A PREROUTING -i eth0 -p udp --dport 8080 \
               -j DNAT --to 192.168.1.5
```

En el caso de que el puerto de solicitud al router, no sea el mismo que el de servicio, será necesario indicar el mapeo o redireccionamiento de puertos. Por ejemplo, si en el caso anterior recibimos la solicitud de conexión por el puerto 8080, pero el servidor da servicio por el 80, el comando iptables sería el siguiente:

```
[root@linux] # iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 8080 \
               -j DNAT --to 192.168.1.5:80
[root@linux] # iptables -t nat -A PREROUTING -i eth0 -p udp --dport 8080 \
               -j DNAT --to 192.168.1.5:80
```

2.3.- Filtrado de paquetes mediante iptables: FILTER.

Ipables, como va a poder probarse, es un perfecto firewall que es capaz de filtar los paquetes TCP/IP en función de las reglas que se le hayan programado. Con la finalidad de facilitar su configuración, iptables cuenta con tres tablas o listas de reglas correspondientes a los tres tipos de filtrado que puede llevar a cabo: INPUT, OUTPUT o FORWARD. En concreto, los paquetes TCP/IP manipulados por el equipo firewall podríamos clasificarlos de la siguiente manera:

1. **INPUT**: Paquetes TCP/IP que vayan destinados al propio equipo firewall. Es decir, en la cabecera de estos paquetes debe figurar la dirección ip del firewall como dirección ip de destino.
2. **OUTPUT**: Paquetes TCP/IP generados por el propio equipo firewall. Es decir, en la cabecera de estos paquetes debe figurar la dirección ip del firewall como dirección ip de origen.
3. **FORWARD**: Paquetes TCP/IP que atraviesan al equipo firewall. Es decir, en la cabecera de estos paquetes deben figurar como direcciones ip de origen y destino que no se corresponden con las del equipo firewall.

De una manera general, a modo de ejemplo, la sintaxis del comando iptables que sería necesario ejecutar para agregar (*-A, add*) una nueva regla de filtrado sería la siguiente:

```
[root@linux] # iptables -t filter -A [INPUT|OUTPUT|FORWARD] \
                [ -i 'interfaz de red de entrada ] [ -o 'interfaz de red de salida' ] \
                [ -s 'dirección ip de origen' ] [ -d 'dirección ip de destino' ] \
                [ -p tcp|udp|icmp ] [ --dport 'puerto de destino' ] [ --sport 'puerto de origen' ] \
                [ -j ACCEPT|DROP|REJECT ]
```

¡¡Observación!! Al filtrar paquetes TCP/IP mediante **iptables** se nos permite tomar una decisión entre tres posibles:

(1) ACCEPT: Aceptará todos aquellos paquetes a los que hagan referencia los parámetros especificados.

```
[root@firewall]# iptables -t filter -A FORWARD -i eth1 -o eth0 \
                 -p tcp --dport 80 -s 192.168.2.0/24 -j ACCEPT
```

(2) DROP: Denegará todos aquellos paquetes a los que hagan referencia los parámetros especificados.

```
[root@firewall]# iptables -t filter -A INPUT -p tcp --dport 22 -s 192.168.1.180 -j DROP
```

(3) REJECT: Similar a **DROP** pero con la salvedad de que podemos especificar el motivo del rechazo, el cual será enviado al equipo emisor de los paquetes rechazados. En concreto, si el protocolo utilizado es **ICMP** o **UDP** pueden usarse las siguientes opciones (*icmp-port-unreachable es la opción por defecto*): **--reject-with icmp-net-unreachable | icmp-host-unreachable | icmp-port-unreachable | icmp-proto-unreachable | icmp-net-prohibited | icmp-host-prohibited | icmp-admin-prohibited**.

```
[root@firewall]# iptables -t filter -A FORWARD -p udp --dport 53 \
                 -d 8.8.8.8 -j REJECT --reject-with icmp-host-unreachable
```

En el caso de que el protocolo que se este filtrando sea **TCP** se recomienda usar directamente **DROP** o **REJECT** con la opción **--reject-with tcp-reset**.

```
[root@firewall]# iptables -t filter -A FORWARD -p tcp --dport 443 \
                 -d www.facebook.com -j REJECT --reject-with tcp-reset
```

Por ejemplo, si quisiéramos denegar el acceso a un equipo con dirección IP 192.168.1.1 al servicio SSH (*tcp/22*) que ofrece un equipo GNU/Linux ejecutaríamos lo siguiente:

```
[root@linux] # iptables -t filter -A INPUT -p tcp --dport 22 -s 192.168.1.1 -j DROP
```

A continuación, se plantearán diferentes casos prácticos con su solución, lo cual ayudará a comprender mejor la sintáxis y utilidad de iptables. Por último, comentar que además de la opción "**-A**" (añadir regla), disponemos de las siguientes opciones:

- Opción "**-L**": Muestras las reglas que hay programadas en una determinada lista de filtrado o nat. Si queremos que se muestre un número que identifique la posición de una regla dentro de la lista añadiremos la opción "--line-numbers".

```
[root@linux] # iptables -t filter -L
[root@linux] # iptables -t filter -L INPUT
[root@linux] # iptables -t filter -L FORWARD --line-numbers
[root@linux] # iptables -t nat -L POSTROUTING --line-numbers
```

- Opción "**-I**": Inserta una regla a una lista de filtrado o nat de nuestro firewall en la posición indicada como parámetro. Esta opción resulta muy útil cuando queremos insertar una regla en una posición concreta para que sea tenida en cuenta por el firewall antes que otras que se programaron previamente.

```
[root@linux] # iptables -t filter -I FORWARD 2 -i eth0 -o eth1 -p tcp --dport 443 -j ACCEPT
```

- Opción "**-D**": Borra una regla de una lista. Para ello indicaremos la lista que queremos que se vea afectada y su posición.

```
[root@linux] # iptables -t filter -D OUTPUT 1
```

- Opción "**-F**": Borrar todas las reglas de la lista indicada.

```
[root@linux] # iptables -t filter -F OUTPUT
[root@linux] # iptables -t filter -F
[root@linux] # iptables -t nat -F
```

- Opción "**-P**": Indica la política por defecto. Es decir, mediante esta opción le indicaremos al firewall cual queremos que sea el criterio por defecto a aplicar a un paquete TCP/IP que no se vea afectado por ninguna de las reglas programadas.

```
[root@linux] # iptables -t filter -P INPUT DROP
[root@linux] # iptables -t filter -F FORWARD ACCEPT
```

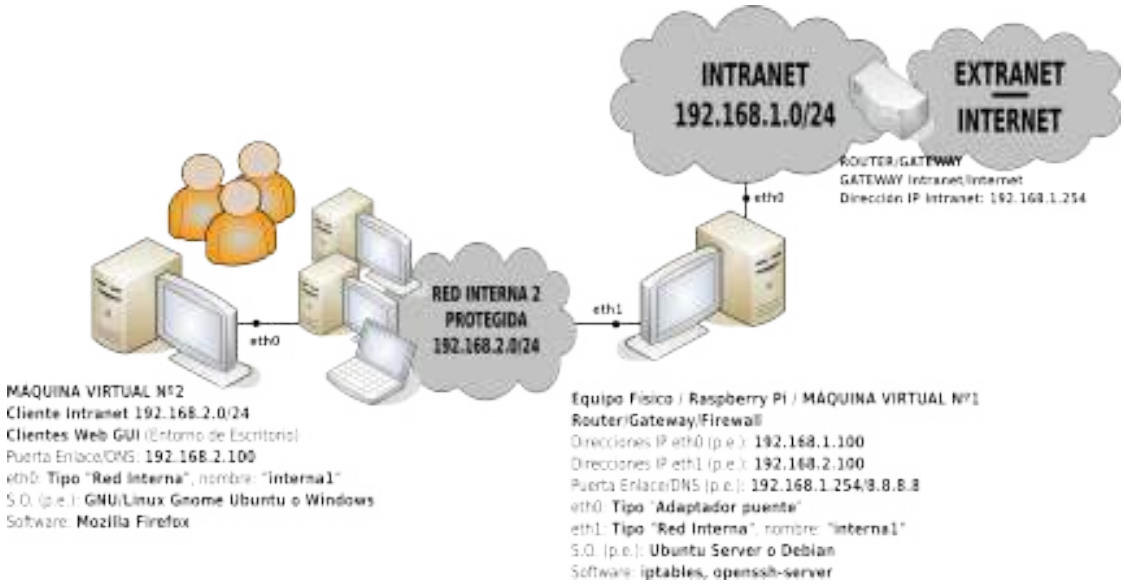
¡¡Aclaración!! iptables en un software que ofrece infinidad de posibilidades, tal como se puede advertir a través de su manual de uso (*man iptables*). Tratar de ver todas sus posibilidades sería imposible por lo que en los ejercicios prácticos propuestos a continuación tan sólo se va a mostrar algunos de usos más habituales.

Ej. Práctico 2.3.1: Filtrado Básico de Paquetes TCP/IP DNS/HTTP/HTTPS

Observa la red que se muestra en la siguiente figura. En ella se muestra la disposición típica de un router/gateway/firewall dentro de una Intranet privada. En este ejercicio práctico mostraremos una configuración básica y sencilla para que dicho equipo permita únicamente tráfico **tcp/80 HTTP**, **tcp/443 HTTPS** y **udp/53 DNS** desde la Intranet hacia Internet. Cualquier otro puerto o protocolo quedará cerrado hacia el exterior.

Advertir que el servicio **udp/53 DNS** será necesario habilitarlo independientemente del servicio de Internet que se desea que sea accesible, ya que actualmente para acceder referencia a cualquier servicio de Internet se hace uso de nombres de dominio en lugar de direcciones IP (*p.e.* *www.google.com*, *amartinromero@gmail.com*, *etc.*).

Origen	Destinos Permitidos	Protocolos y Puertos	Decisión
192.168.2.0/24	**8.8.8.8**	**dns 53/udp**	**ACCEPT**
	0.0.0.0/0 *excepto* **www.tvporno.com**	**http 80/tcp**	**ACCEPT**
	0.0.0.0/0 *excepto* **www.facebook.com**	**https 443/tcp**	**ACCEPT**



Solución Ej. Pr. 2.3.1.I.- Como Configurar un Firewall Básico

Para dar solución al ejercicio práctico planteado se creará un script de configuración para el equipo **gateway/firewall** (*p.e. máquina virtual nº1 bajo Ubuntu Server*) del esquema, p.e. /etc/init.d/**conf-red.sh**, en el cual se incluirá la configuración de red (*asignación de direcciones IP a las interfaces de red, puerta de enlace y servidor DNS predeterminado*), la configuración como gateway (*activar ip_forward y enmascaramiento*) y la configuración del firewall:

```
[root@firewall]# nano /etc/init.d/conf-red.sh
#!/bin/bash
clear
echo "Comenzamos con la configuración IP del equipo (por ejemplo):"
/etc/init.d/networking stop
ifconfig eth0 down
ifconfig eth1 down
echo "Asignamos las Ips, Puerta de Enlace y Servidor DNS preferido ..."
ifconfig eth0 192.168.1.100 netmask 255.255.255.0
ifconfig eth1 192.168.2.100 netmask 255.255.255.0
route add default gw 192.168.1.254
echo "nameserver 8.8.8.8" > /etc/resolv.conf
…
```

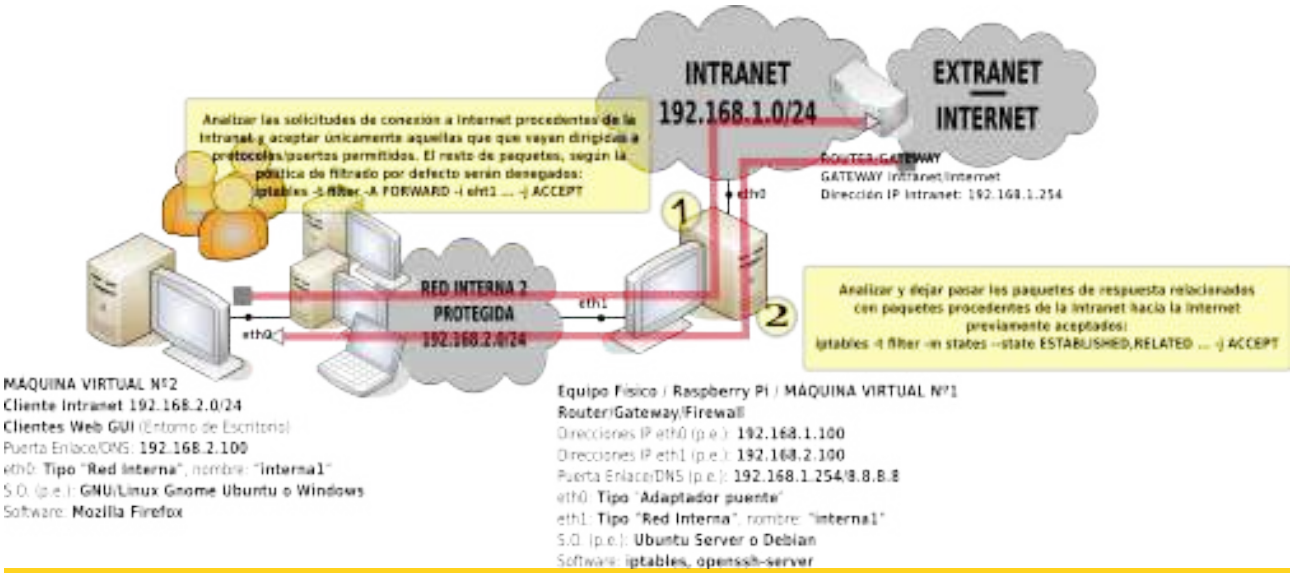
Tras la configuración IP del equipo gateway/firewall será necesario **(1)** activar el **ip_forward** para permitir el reenvío de paquetes TCP/IP entre las diferentes interfaces de red del equipo y **(2)** enmascarar las direcciones IP privadas de origen de la Intranet sustituyéndolas por la dirección IP de la interfaz externa del gateway con la finalidad de que los paquetes TCP/IP que salen hacía Internet puedan llegar a ser devueltos posteriormente a su origen, ya que en caso de no hacerlo el siguiente router por el que pasen los paquetes TCP/IP no sabría enrutarlos/devolverlos hacía el origen al tratarse de IP de origen privadas no unívocas.

```
…
echo "Continuamos con la configuración como gateway o puerta de enlace:"
echo "Activamos el forwarding de paquetes para que actúe como un gateway ..."
echo "1" > /proc/sys/net/ipv4/ip_forward
echo "Enmascaramos los Paquetes que salen de la Intranet hacía Internet ..."
iptables -t nat -F
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
…
```

Por último estableceremos las reglas de filtrado de tipo FORWARD (*iptables -t filter -A FORWARD*) necesarias que permitan filtrar los paquetes TCP/IP que atraviesan al firewall entre la Intranet e Internet permitiendo únicamente las conexiones a los servicios especificados en la tabla del enunciado del ejercicio DNS/HTTP/HTTPS:

```
…
echo "Por último, Terminamos el script configurando el firewall:"
echo "Establecemos como Política por Defecto Denegar Todo desde la Intranet hacia Internet!!"
iptables -t filter -F
iptables -t filter -P FORWARD DROP
echo "Permitimos Resoluciones DNS udp/53 a los equipos de la Intranet Protegida ..."
iptables -t filter -A FORWARD -i eth1 -o eth0 -s 192.168.2.0/24 -d 8.8.8.8 \
        -p udp --dport 53 -j ACCEPT
echo "Dejamos pasar los paquetes de respuesta DNS udp/53 procedentes de Internet"
iptables -t filter -A FORWARD -i eth0 -o eth1 -p udp -s 8.8.8.8 --sport 53 \
```

```
        -m state --state ESTABLISHED,RELATED -j ACCEPT
echo "Permitimos Conexiones HTTP y HTTPS tcp/80/443 con algunas excepciones ..."
iptables -t filter -A FORWARD -i eth1 -o eth0 -p tcp --dport 80 \
        -s 192.168.2.0/24 -d www.tvporno.com -j DROP
iptables -t filter -A FORWARD -i eth1 -o eth0 -p tcp --dport 443 \
        -s 192.168.2.0/24 -d www.facebook.com -j DROP
iptables -t filter -A FORWARD -i eth1 -o eth0 -p tcp -m multiport --dport 80,443 \
        -s 192.168.2.0/24 -j ACCEPT
echo "Dejamos pasar los paquetes de respuesta HTTP/HTTPS tcp/80/443 procedentes de Internet"
iptables -t filter -A FORWARD -i eth0 -o eth1 -p tcp -m multiport --sport  80,443 \
        -m state --state ESTABLISHED,RELATED -j ACCEPT
```



¡¡Observaciones!! Para comprender mejor las reglas de filtrado anteriores deberían tenerse en cuenta los siguientes aspectos:

a) Para permitir a los clientes de la Intranet únicamente el acceso a los servicios DNS/HTTP/HTTPS ofrecidos en la Internet y excluir el resto de servicios (*cerrar los puertos*) se ha asignado como política por defecto para los paquetes TCP/IP que atraviesan al firewall denegar todo, **iptables -t filter -P FORWARD DROP**. De esta forma, sólo se permiten el paso de aquellos paquetes que se especifican explícitamente.

b) Para permitir el acceso a un cliente a un servicio ofrecido en la Internet será necesario dejar pasar tanto los paquetes que van dirigidos desde el cliente hacía el servidor, como los paquetes de vuelta del servidor al cliente. Estos últimos son identificados fácilmente por el firewall consultando unos flags o indicadores de la cabecera de los paquetes TCP/IP a los cuales podemos hacer referencia en iptables mediante **-m state --state ESTABLISHED,RELATED**. **RELATED** hace referencia a paquetes asociados a una solicitud de conexión realizada y **ESTABLISHED** hace referencia a todos los paquetes que están relacionados con una conexión ya establecida.

c) Para poder hacer uso de nombres de dominio en una regla de filtrado (*p.e. **-d www.tvporno.com -j DROP***) tenemos que tener en cuenta que el firewall debe tener acceso al servicio DNS. Es decir, iptables resuelve los nombres de dominio que se indiquen en las reglas de filtrado y en el caso de

que un nombre de dominio tenga asignados más de una dirección IP pública (*p.e. dig www.google.com*) se programará una regla de filtrado por cada una de ellas. Esto se podrá comprobar al listar las reglas programadas en el firewall mediante **iptables -t filter -n -L FORWARD**, o **iptables -t filter -n -L FORWARD --line-numbers** si queremos que nos muestre la posición que ocupa cada regla (*la opción -**n** evita que el firewall haga resolución inversa al listar mostrándonos de esta forma las direcciones IP públicas asociadas a los nombres de dominio indicados*).

Para comprobar el correcto funcionamiento del script anterior, le daremos permiso de ejecución, lo ejecutaremos y mediante un cliente de la Intranet (*una segunda máquina virtual con interfaz de red en la misma red interna que la eth1 de la primera máquina virtual*) comprobaremos el acceso a diferentes servicios de la Internet:

```
[root@firewall]# chmod +x /etc/init.d/conf-red.sh && /etc/init.d/conf-red.sh
```

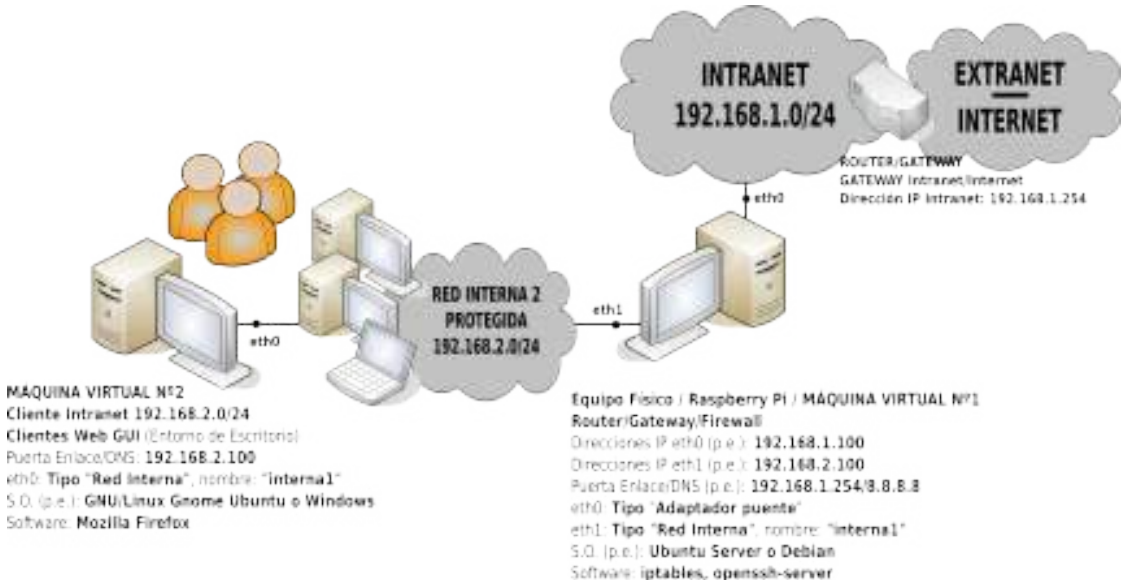
¡¡Importante!! La configuración establecida mediante la ejecución del script anterior se pierde al reiniciar el equipo. Por ello, si queremos que el script anterior se autoejecute al iniciarse la máquina, la forma más fácil es lanzarlo a través del script "**rc.local**" que se ejecuta al final del inicio de la máquina, añadiéndolo al final de dicho script (*doble redireccionamiento, ">>"*):

```
[root@firewall]# echo "/etc/init.d/conf-red.sh" >> /etc/init.d/rc.local
```

Ej. Práctico 2.3.2: Filtrado de Paquetes ICMP/DNS/HTTP/HTTPS/SSH

Observa la figura e indica que comandos incluirías en el script de configuración del equipo **gateway/firewall** (*máquina virtual nº1*) del esquema, p.e. /etc/init.d/**conf-red.sh**, para que el tuviera la configuración de red que se muestra, y además permitiera **únicamente** las conexiones que se describen en la siguiente tabla (*el resto denegadas por defecto*):

Origen	Destinos	Protocolos y Puertos	Decisión
127.0.0.1 *(firewall)* **192.168.2.0/24**	**8.8.8.8**	**dns 53/udp**	**ACCEPT**
192.168.2.0/24	**8.8.8.8** **www.google.com**	**icmp** *(ping)* *Pruebas de Conectividad*	**ACCEPT**
192.168.2.0/24	**0.0.0.0/0** *excepto* **www.tvporno.com** **www.sexfilms.com**	**http 80/tcp**	**ACCEPT**
192.168.2.0/24	**0.0.0.0/0** *excepto* **www.facebook.com** **www.twitter.com**	**https 443/tcp**	**ACCEPT**
192.168.2.101	**192.168.3.100** *(firewall)* **73.45.123.76**	**ssh/scp 22/tcp**	**ACCEPT**



Solución Ej. Pr. 2.3.2.I.- Como Filtrar Protocolos ICMP/DNS/HTTP/HTTPS/SSH

Para dar solución al ejercicio práctico planteado se creará un script de configuración para el equipo **gateway/firewall** (*p.e. máquina virtual en VirtualBox nº1 bajo Ubuntu Server*) del esquema,

p.e. /etc/init.d/**conf-red.sh**, en el cual se incluirá la configuración de red (*asignación de direcciones IP a las interfaces de red, puerta de enlace y servidor DNS predeterminado*), la configuración como gateway (*activar ip_forward y enmascaramiento*) y la configuración del firewall:

```
[root@firewall]# nano /etc/init.d/conf-red.sh
#!/bin/bash
/etc/init.d/networking stop
ifconfig eth0 down
ifconfig eth1 down
echo "Asignamos las Ips, Puerta de Enlace y Servidor DNS preferido ..."
ifconfig eth0 192.168.1.100 netmask 255.255.255.0
ifconfig eth1 192.168.2.100 netmask 255.255.255.0
route add default gw 192.168.1.254
echo "nameserver 8.8.8.8" > /etc/resolv.conf
…
```

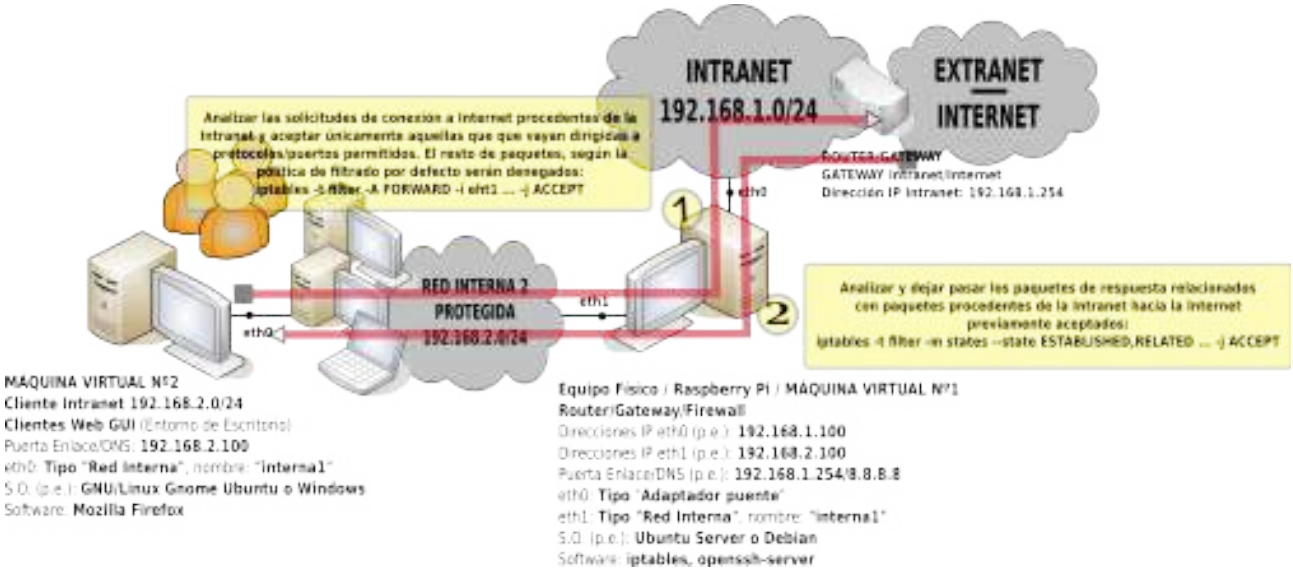
Tras la configuración IP del equipo gateway/firewall será necesario **(1)** activar el **ip_forward** para permitir el reenvío de paquetes TCP/IP entre las diferentes interfaces de red del equipo y **(2)** enmascarar las direcciones IP privadas de origen de la Intranet sustituyéndolas por la dirección IP de la interfaz externa del gateway con la finalidad de que los paquetes TCP/IP que salen hacía Internet puedan llegar a ser devueltos posteriormente a su origen, ya que en caso de no hacerlo el siguiente router por el que pasen los paquetes TCP/IP no sabría enrutarlos/devolverlos hacía el origen al tratarse de IP de origen privadas no unívocas.

```
…
echo "Activamos el forwarding de paquetes para que actúe como un gateway ..."
echo "1" > /proc/sys/net/ipv4/ip_forward
echo "Enmascarmos los Paquetes que salen de la Intranet hacía Internet ..."
iptables -t nat -F
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
…
```

Continuaremos con la parte del script relacionada con el filtrado de los paquetes. En concreto, comenzaremos asignando como **políticas por defecto** no aceptar ningún paquete TCP/IP que vaya dirigido a nosotros como destinatarios (**-P INPUT DROP**), no aceptar ningún paquete que salga de nuestro equipo generado por el propio equipo (**-P OUTPUT DROP**) y no aceptar ningún paquete que vaya a atravesar al firewall entre la Intranet e Internet (**-P FORWARD DROP**). A consecuencia de esta política por defecto, cualquier acceso a un servicio que queramos que sea permitido deberá aceptarse explícitamente:

```
…
echo "Establecemos como Política por Defecto Denegar Todo!!"
iptables -t filter -F
iptables -t filter -P INPUT DROP
iptables -t filter -P OUTPUT DROP
iptables -t filter -P FORWARD DROP
echo "Permitimos Conexiones udp/53 desde el propio Firewall para poder Resolver Nombres ..."
```

```
iptables -t filter -A OUTPUT -o eth0 -p udp --dport 53 -d 8.8.8.8 -j ACCEPT
echo "Permitimos los Paquetes de Respuesta a las solicitudes DNS udp/53 anteriores ..."
iptables -t filter -A INPUT -i eth0 -s 8.8.8.8 -p udp --sport 53 \
        -m state --state ESTABLISHED,RELATED -j ACCEPT
echo "Permitimos Resoluciones DNS a los equipos de la Intranet Protegida ..."
iptables -t filter -A FORWARD -i eth1 -o eth0 -s 192.168.2.0/24 -d 8.8.8.8 \
        -p udp --dport 53 -j ACCEPT
iptables -t filter -A FORWARD -i eth0 -o eth1 -p udp -s 8.8.8.8 --sport 53 \
        -m state --state ESTABLISHED,RELATED -j ACCEPT
echo "Permitimos el uso de Ping en la Intranet para hacer Pruebas de Conectividad con Internet ..."
iptables -t filter -A FORWARD -i eth1 -o eth0 -p icmp -s 192.168.2.0/24 -d 8.8.8.8 -j ACCEPT
iptables -t filter -A FORWARD -i eth1 -o eth0 -p icmp -s 192.168.2.0/24 -d www.google.com \
        -j ACCEPT
iptables -t filter -A FORWARD -i eth0 -o eth1 -p icmp \
        -m state --state ESTABLISHED,RELATED -j ACCEPT
echo "Permitimos Conexiones HTTP y HTTPS tcp/80/443 con algunas excepciones ..."
iptables -t filter -A FORWARD -i eth1 -o eth0 -p tcp --dport 80 \
        -s 192.168.2.0/24 -d www.tvporno.com -j DROP
iptables -t filter -A FORWARD -i eth1 -o eth0 -p tcp --dport 80 \
        -s 192.168.2.0/24 -d www.sexfilms.com -j DROP
iptables -t filter -A FORWARD -i eth1 -o eth0 -p tcp --dport 443 \
        -s 192.168.2.0/24 -d www.facebook.com -j DROP
iptables -t filer -A FORWARD -i eth1 -o eth0 -p tcp --dport 443 \
        -s 192.168.2.0/24 -d www.twitter.com -j DROP
iptables -t filter -A FORWARD -i eth1 -o eth0 -p tcp -m multiport --dport 80,443 \
        -s 192.168.2.0/24 -j ACCEPT
iptables -t filter -A FORWARD -i eth0 -o eth1 -p tcp -m multiport --sport  80,443 \
        -m state --state ESTABLISHED,RELATED -j ACCEPT
echo "Por último, Permitimos determinadas Conexiones SSH tcp/22 desde 192.168.2.101..."
iptables -t filter -A INPUT -i eth1 -s 192.168.2.101 -p tcp --dport 22 -j ACCEPT
iptables -t filter -A OUTPUT -o eth1 -d 192.168.2.101 -p tcp --sport 22 -j ACCEPT
iptables -t filter -A FORWARD -i eth1 -o eth0 -s 192.168.2.101 -d 73.45.123.76 \
        -p tcp --dport 22 -j ACCEPT
iptables -t filter -A FORWARD -i eth0 -o eth1 -d 192.168.2.101 -s 73.45.123.76 \
        -p tcp --sport 22 -j ACCEPT
```



¡¡Observaciones!! Para comprender mejor las reglas de filtrado anteriores deberían tenerse en cuenta los siguientes aspectos:

a) Para permitir únicamente a los clientes de la Intranet el acceso a los servicios DNS/HTTP/HTTPS ofrecidos en la Internet y excluir el resto de servicios (*cerrar los puertos*) se ha asignado como política por defecto para los paquetes TCP/IP que atraviesan al firewall denegar todo, **iptables -t filter -P FORWARD DROP**. De esta forma, sólo se permiten el paso de aquellos paquetes que se especifican explícitamente.

b) Para permitir el acceso a un cliente a un servicio ofrecido en la Internet será necesario dejar pasar tanto los paquetes que van dirigidos desde el cliente hacía el servidor, como los **paquetes de vuelta del servidor al cliente**. Estos últimos son identificados fácilmente por el firewall consultando unos flags o indicadores de la cabecera de los paquetes TCP/IP a los cuales podemos hacer referencia en iptables mediante **-m state --state ESTABLISHED,RELATED**. **RELATED** hace referencia a paquetes asociados a una solicitud de conexión realizada y **ESTABLISHED** hace referencia a todos los paquetes que están relacionados con una conexión ya establecida.

iptables -t filter ... -m state --state ESTABLISHED,RELATED -j ACCEPT

c) Para poder hacer uso de nombres de dominio en una regla de filtrado (*p.e. **-d** **www.tvporno.com** **-j DROP***) tenemos que tener en cuenta que el firewall debe tener acceso al servicio DNS. Es decir, iptables resuelve los nombres de dominio que se indiquen en las reglas de filtrado y en el caso de que un nombre de dominio tenga asignados más de una dirección IP pública (*p.e. dig www.google.com*) se programará una regla de filtrado por cada una de ellas. Esto se podrá comprobar al listar las reglas programadas en el firewall mediante **iptables -t filter -n -L FORWARD**, o **iptables -t filter -n -L FORWARD --line-numbers** si queremos que nos muestre la posición que ocupa cada regla (*la opción **-n** evita que el firewall haga resolución inversa al listar mostrándonos de esta forma las direcciones IP públicas asociadas a los nombres de dominio indicados*).

d) Cuando queremos especificar una lista de puertos, y estos no son consecutivos, puede hacerse

uso de la opción "**-m multiport --dport 80,443**". En el caso de que fueran consecutivos, por ejemplo del 8000 al 8008, sería suficiente con poner "**--dport 8000:8008**"

Para comprobar el correcto funcionamiento del script de configuración anterior le daremos permisos de ejecución y lo ejecutaremos. Posteriormente será necesario hacer las pertinentes pruebas desde el propio equipo firewall o desde un cliente de la Intranet protegida, comprobando que tan sólo pueden acceder a los servicios especificados:

```
[root@firewall]# chmod +x /etc/init.d/conf-red.sh && /etc/init.d/conf-red.sh
```

¡¡Importante!! La configuración establecida mediante la ejecución del script anterior se pierde al reiniciar el equipo. Por ello, si queremos que el script anterior se autoejecute al iniciarse la máquina, la forma más fácil es lanzarlo a través del script "**rc.local**" que se ejecuta al final del inicio de la máquina, añadiéndolo al final de dicho script (*doble redireccionamiento, ">>"*):

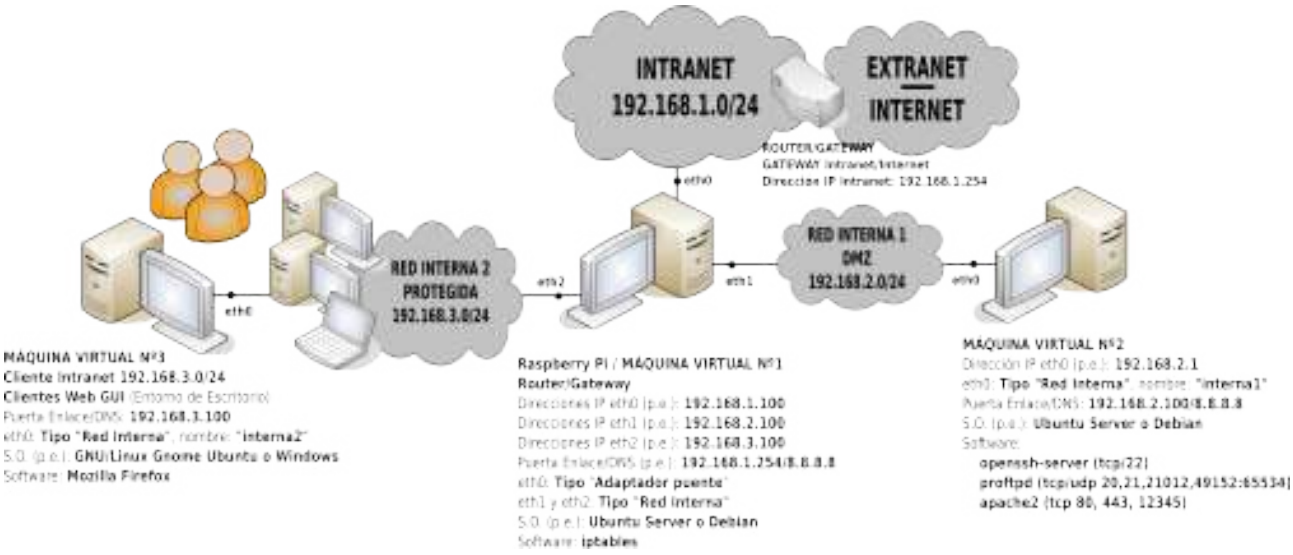
```
[root@firewall]# echo "bash /etc/init.d/conf-red.sh" >> /etc/init.d/rc.local
```

Ej. Práctico 2.3.3: Redireccionamiento Puertos (*DNAT PreRouting*)

Si en los ejercicios prácticos anteriores se ha hecho énfasis en el uso de **iptables** para el filtrado de las comunicaciones, en éste aprenderemos a usarlo para el redireccionamiento de puertos hacía un equipo servidor. Esta faceta de **iptables** nos va a permitir, una vez recibida por el firewall una solicitud de conexión por un puerto concreto, redireccionar dicha petición hacia otro equipo encargado de ofrecer un servicio por el mismo puerto u otro diferente.

Teniendo en cuenta el esquema de red que se muestra en la siguiente figura, ¿Qué comandos añadirías a un script de configuración para **permitir que desde el exterior** de una Intranet, Extranet o Internet, pueda accederse a los siguientes servicios ofrecidos por un servidor localizado en la Zona Desmilitarizada **DMZ** (*192.168.2.0/24*) de una red privada (*p.e. equipo servidor 192.168.2.1*)?

Equipos Clientes de Internet Permitidos	Servicios Ofrecidos por la DMZ	
	Servicio Accesible desde el Exterior	Servicio Ofrecido DMZ
Cualquiera	**HTTP tcp/80**	**HTTP tcp/80**
Cualquiera	**HTTP tcp/8088**	**HTTP tcp/12345**
Cualquiera	**HTTPS tcp/443**	**HTTPS tcp/443**
112.43.3.67	**FTP tcp/upd 20/21/21012/49152:65534**	**FTP tcp/upd 20/21/21012/49152:65534**
83.211.54.76	**SSH/SCP/SFTP tcp/22222**	**SSH/SCP/SFTP tcp/22**

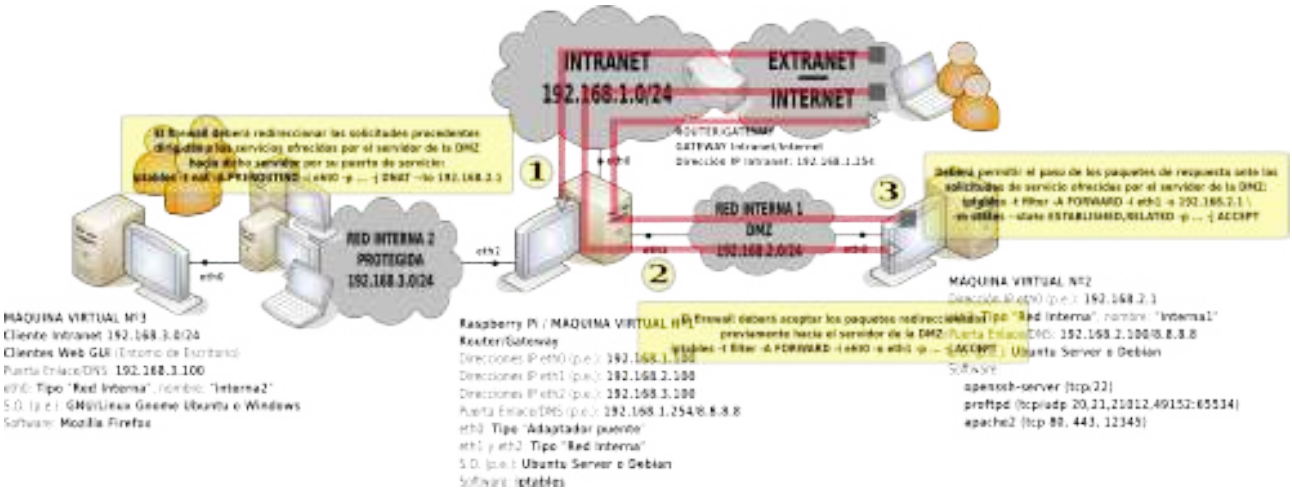


Solución Ej. Pr. 2.3.3.I.- Como Configurar DNAT PreRouting

Para poder dar solución al ejercicio practico planteado deberemos comprender previamente en que consiste la **DNAT PreRouting**. Como bien indican sus siglas, DNAT hace referencia a una Traslación de Direcciones de Red de Destino, lo que significa que **iptables** nos permite cambiar la

dirección IP de destino de la cabecera de los paquetes TCP/IP. PreRouting indica que ese cambio se hará nada más recibir el paquete, previo a cualquier enrutamiento. En otras palabras, **DNAT PreRouting** nos va a permitir analizar los paquetes que le llegan al firewall por alguna de sus interfaces de red (*p.e. -i eth0*) y en función de si se cumplan las premisas establecidas en la regla iptables que se describa se podrán modificar tanto la dirección IP de destino, como el puerto de destino, para posteriormente ser enrutado hacía el destino indicado.

Según lo anterior, si tenemos en cuenta que todo DNAT implica un posterior enrutamiento del paquete TCP/IP modificado hacia su nuevo destino, deberá permitirse dicho envío mediante una regla de filtrado de tipo FORWARD en el caso de que la política por defecto sea DROP. Por tanto, tal como se va a mostrar en la siguiente solución, toda regla de DNAT irá acompañada de las reglas de filtrado FORWARD necesarias para que los paquetes de ida y vuelta puedan atravesar al firewall.




```
[root@mv1]# nano /etc/init.d/conf-nat.sh
#!/bin/bash
echo "Redireccionamos los Puertos tcp/80 y tcp/443 al servidor ..."
iptables -t nat -A PREROUTING -i eth0 -p tcp -m multiport --dport 80,443 \
        -j DNAT --to 192.168.2.1
iptables -t filter -A FORWARD -i eth0 -o eth1 -p tcp -m multiport --dport 80,443 \
        -d 192.168.2.1 -j ACCEPT
echo "Redireccionamos el Puerto tcp/8088 al servidor bajo el puerto tcp/12345 ..."
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 8088 -j DNAT --to 192.168.2.1:12345
iptables -t filter -A FORWARD -i eth0 -o eth1 -p tcp --dport 12345 -d 192.168.2.1 -j ACCEPT
echo "Redireccionamos los Puertos del Servicio FTP tcp/udp/20/21/21012/49152:65534 ..."
iptables -t nat -A PREROUTING -i eth0 -s 112.43.3.67 \
        -p tcp -m multiport --dport 20,21,21012,49152:65534 -j DNAT --to 192.168.2.1
iptables -t filter -A FORWARD -i eth0 -o eth1 -s 112.43.3.67 \
        -p tcp -m multiport --dport 20,21,21012,49152:65534 -j ACCEPT
iptables -t nat -A PREROUTING -i eth0 -s 112.43.3.67 \
        -p udp -m multiport --dport 20,21,21012,49152:65534 -j DNAT --to 192.168.2.1
iptables -t filter -A FORWARD -i eth0 -o eth1 -s 112.43.3.67 \
        -p udp -m multiport --dport 20,21,21012,49152:65534 -j ACCEPT
echo "Redireccionamos el Puerto tcp/22222 hacia el tcp/22 para Conexiones SSH ..."
iptables -t nat -A PREROUTING -i eth0 -s 83.211.54.76 -p tcp --dport 22222 \
```

```
        -j DNAT --to 192.168.2.1:22
iptables -t filter -A FORWARD -i eth0 -o eth1 -p tcp --dport 22 \
        -s 83.211.54.76 -d 192.168.2.1 -j ACCEPT
echo "Permitimos el paso de todos los paquetes de vuelta asociados a los servicios anteriores: "
iptables -t filter -A FORWARD -i eth1 -o eth0 \
        -m state --state ESTABLISHED,RELATED -j ACCEPT
```

¡¡Observación!! La solución al ejercicio práctico anterior guarda relación con la mostrada en los ejercicios prácticos anteriores, dando por hecho que la **política por defecto** de **FORWARD** es **DROP**. En caso de no ser así, si fuera ACCEPT la política por defecto de FORWARD obviamente todas las reglas anteriores de FORWARD no serían necesarias.

Para comprobar el correcto funcionamiento del script de configuración anterior le daremos permisos de ejecución y lo ejecutaremos. Posteriormente será necesario hacer las pertinentes pruebas de acceso a los servicios anteriores desde un equipo de la Extranet (*p.e. la máquina física desde la que se ha desplegado el entorno de máquinas virtuales VirtualBox*):

```
[root@mv1]# chmod +x /etc/init.d/conf-nat.sh && /etc/init.d/conf-nat.sh
```

¡¡Recordatorio!! La configuración establecida mediante la ejecución del script anterior se pierde al reiniciar el equipo. Por ello, si queremos que el script anterior se autoejecute al iniciarse la máquina, la forma más fácil es lanzarlo a través del script "**rc.local**" que se ejecuta al final del inicio de la máquina, añadiéndolo al final de dicho script (*doble redireccionamiento, ">>"*):

```
[root@mv1]# echo "bash /etc/init.d/conf-nat.sh" >> /etc/init.d/rc.local
```

Para poder hacer las pruebas será necesario poner en marcha servicios por los puertos indicados en el equipo que haga de servidor (*p.e. 192.168.2.1*):

- **HTTP tcp/80 tcp/12345**: Puede instalarse en el servidor **apache2** (*apt-get install apache2*) y configurarlo. El puerto 80 es el puerto por defecto de escucha en Apache, por lo que tan sólo habría que decirle que escuche también por el 12345. Para ello, lo más sencillo sería editar el sitio web por defecto servidor por apache y añadir las siguientes líneas (*mostraría la misma página de inicio que el sitio web por defecto a no ser que indiquemos otro documento en DirectoryIndex o ruta en DocumentRoot*):

```
[root@mv1]# apt-get install apache2
[root@mv1]# nano /etc/apache2/sites-enabled/000-default
Listen 12345
<VirtualHost *:12345>
        DocumentRoot /var/www
        DirectoryIndex index.html
</VirtualHost>
...
```

```
[root@mv1]# /etc/init.d/apache2 restart
```

- **HTTPS tcp/443**: Si hacemos uso de Apache, lo más sencillo sería activar el sitio web en modo seguro que por defecto sirve Apache.

```
[root@mv1]# a2ensite default-ssl
[root@mv1]# a2enmod ssl
[root@mv1]# /etc/init.d/apache2 restart
```

- **FTP tcp/udp/20/21/21012/49152:65534** : Para dar servicio FTP por esos puertos podríamos instalar **proftpd** e indicarle a través de su archivo de configuración que además de dar el servicio por los puertos tcp/udp/20/21 (*por defecto*), lo haga también por los puertos 21012, 49152 o el que queramos.

```
[root@mv1]# apt-get install proftpd
[root@mv1]# nano /etc/proftpd/proftpd.conf
Port 21012
Port 49152
…
```

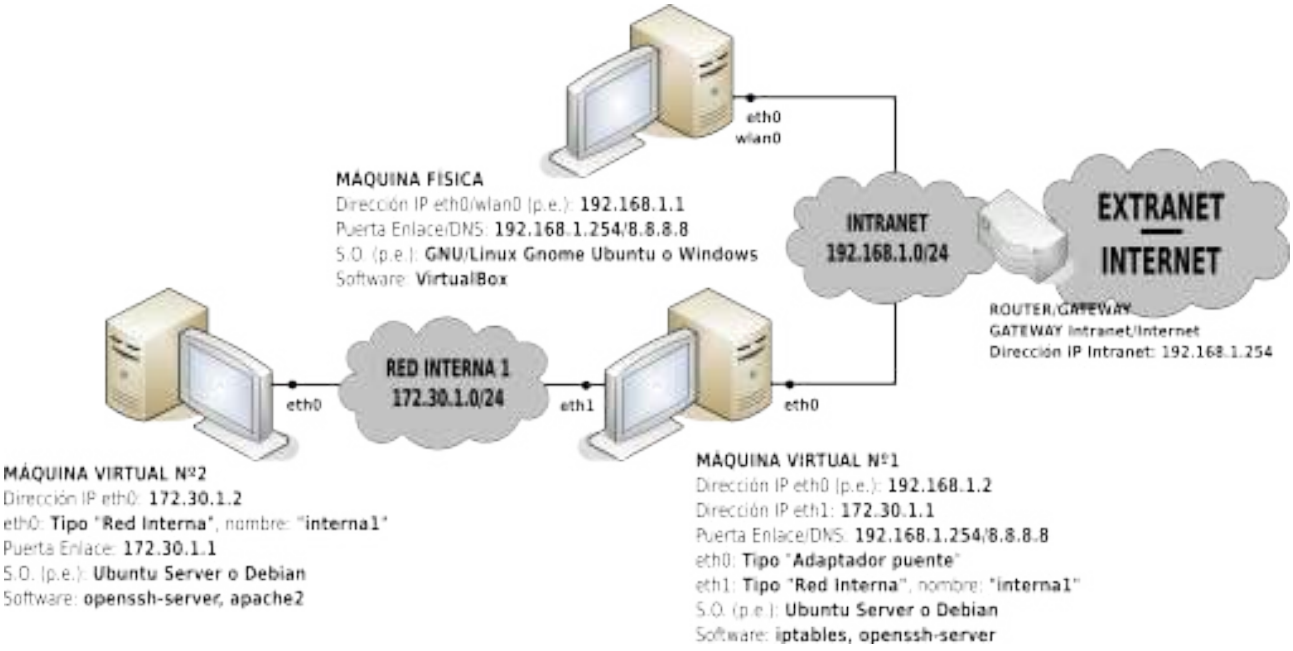
- **SSH/SCP/SFTP tcp/22** : Para ello tan sólo es necesario instalar **openssh-server**. El servicio SFTP lo ofrecerán de manera conjunta proftpd + openssh-server. Para comprobar el correcto funcionamiento tato del servicio FTP como SFTP lo podemos hacer mediante el cliente FileZilla.

```
[root@mv1]# apt-get install openssh-server
```

Ej. Práctico 2.3.4: Reglas de Filtrado y NAT PreRouting

Con la finalidad de repasar los ejercicios prácticos anteriores, implementa el esquema de red mostrado en la siguiente figura haciendo uso de máquinas virtuales VirtualBox, y configura la máquina virtual nº1 (*p.e. bajo Ubuntu Server*), **mv1**, con la finalidad de que haga las veces de firewall de tal forma que cumpla los siguientes requisitos:

a) La **política por defecto** tanto de entrada **(INPUT),** salida **(OUTPUT),** como reenvío **(FORWARD)** de paquetes será denegarlos, **DROP**.
b) Redireccionará (*NAT PreRouting*) las solicitudes recibidas por la propia mv1 procedentes del exterior (*extranet/Internet*) dirigidas hacia el servicio **tcp/8000** a la máquina virtual nº2, **mv2**, que es la que ofrece realmente dicho servicio.
c) Con la finalidad de poder hacer pruebas de conectividad desde la mv2 hacia el exterior, se permitirá el uso del servicio "**icmp**" (*tipo ping*) desde la DMZ (*mv2*) hacia el exterior, tanto mediante el uso de direcciones ip, como mediante el uso de nombres de dominio (*será necesario permitir la resolución de nombres DNS, udp/53*).



Solución Ej. Pr. 2.3.4.I.- Configurar Filtrado y NAT PreRouting

La solución al ejercicio planteado, al igual que se hizo en el ejercicio práctico 1.3.1, se va a mostrar en formato script (*p.e. "conf-red"*), lo cual nos permitirá ejecutarlo y modificarlo fácilmente en cualquier momento, además de posibilitar su ejecución al iniciarse la máquina tal como se mostró en el ejercicio 1.3.1.

[root@mv1]# **nano /etc/init.d/conf-red.sh**

```
#!/bin/bash
# Como Ubuntu Server no dispone de interfaz gráfica, el servicio que se encarga de gestionar de manera automática la red es "networking".  Por tanto, en primer lugar pararemos este servicio
```

para que no entre en conflicto con nuestra configuración
/etc/init.d/networking stop
 # Pasamos a configurar las direcciones IP del equipo, puerta de enlace y DNS
ifconfig eth0 192.168.1.2
ifconfig eth1 172.30.1.1 netmask 255.255.255.0
route add default gw 192.168.1.254
echo "nameserver 8.8.8.8" > /etc/resolv.conf
 # Activamos el forwarding y el enmascaramiento de paquetes para que pueda realizar correctamente su función como gateway
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -t nat -F
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
 # Redireccionamos las peticiones recibidas desde el exterior (eth0) al servicio HTTP:8000 hacia el equipo servidor
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 8000 -j DNAT --to 172.30.1.2
 # Establecemos las políticas por defecto para los paquetes entrantes, salientes o de reenvío. Esto dejará todos los puertos cerrados, tanto de entrada como de salida. Nuestro equipo quedará totalmente aislado. Solución al aparatado a):
iptables -t filter -F
iptables -t filter -P INPUT DROP
iptables -t filter -P INPUT DROP
iptables -t filter -P INPUT DROP
 # Permitimos el reenvío de paquetes dirigidos al servicio ofrecido en la DMZ tcp/8000. Solución al apartado b) y c):
iptables -t filter -A FORWARD -i eth0 -o eth1 -p tcp --dport 8000 -d 172.30.1.2 -j ACCEPT
 # Permitimos el reenvío de paquetes ping desde la mv2 hacía el exterior
iptables -t filter -A FORWARD -i eth1 -s 172.30.1.2 -o eth0 -p icmp -j ACCEPT
 # Para permitir el ping a nombres de máquinas, y no sólo a direcciones ip habrá que permitir las solicitudes al servicio exterior DNS udp/53
iptables -t filter -A FORWARD -i eth1 -s 172.30.1.2 -o eth0 -p udp --dport 53 -j ACCEPT
 # Permitimos igualmente los paquetes de respuesta a las solicitudes del servicio 8000, al ping y a las solicitudes de resolución DNS anteriores
iptables -t filter -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

¡¡Recordatorio!! Tal como ya se ha recalcado en los ejercicios prácticos anteriores, cuando abrimos un puerto en un firewall con la finalidad de ofrecer el acceso a un servicio, tenemos que tener en cuenta que tenemos que dejar pasar igualmente los paquetes de vuelta que están relacionados con la conexión que estamos permitiendo establecer. Para detectar estos paquetes que están relacionados con conexiones previamente establecidas **iptables** puede observar unos **flags** o indicadores que se encuentran en las cabeceras de los paquetes TCP/IP que le informan de ello. De ahí la última regla anterior:

iptables -t filter -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

Para comprobar su correcto funcionamiento, tan sólo será necesario ejecutar el script anterior:

```
[root@mv1]# chmod +x /etc/init.d/conf-red.sh
[root@mv1]# /etc/init.d/conf-red.sh
```

¡¡Recordatorio!! Para que el script anterior se autoejecute al iniciarse la máquina, la forma más fácil es lanzarlo a través del script "**rc.local**" que se ejecuta al final del inicio de la máquina, añadiéndolo al final de dicho script (*doble redireccionamiento, ">>"*):

```
[root@mv1]# echo "/etc/init.d/conf-red.sh" >> /etc/init.d/rc.local
```

Ahora podremos probar los siguientes aspectos prácticos:

- La **mv1** no acepta pings dirigidos a ella, ni tampoco generarlos.
- Desde el equipo servidor **mv2** no se alcanza ningún servicio de Internet, a excepción del uso de la herramienta "ping" para pruebas de conectividad. Por ejmplo, "ping 8.8.8.8" o "ping www.google.com".
- Desde el exterior (*p.e. la máquina física*) poniendo en la barra de direcciones del navegador "http://192.168.1.2:8000" se debe acceder al servicio ofrecido por la **mv2** por ese puerto. Para ofrecer este servicio tan sólo será necesario ejecutar desde un directorio a servir el siguiente comando (*si el directorio desde el que se ejecuta el siguiente comando tiene una página de inicio, index.html, la mostrará, en caso contrario mostrará el contenido del directorio*):

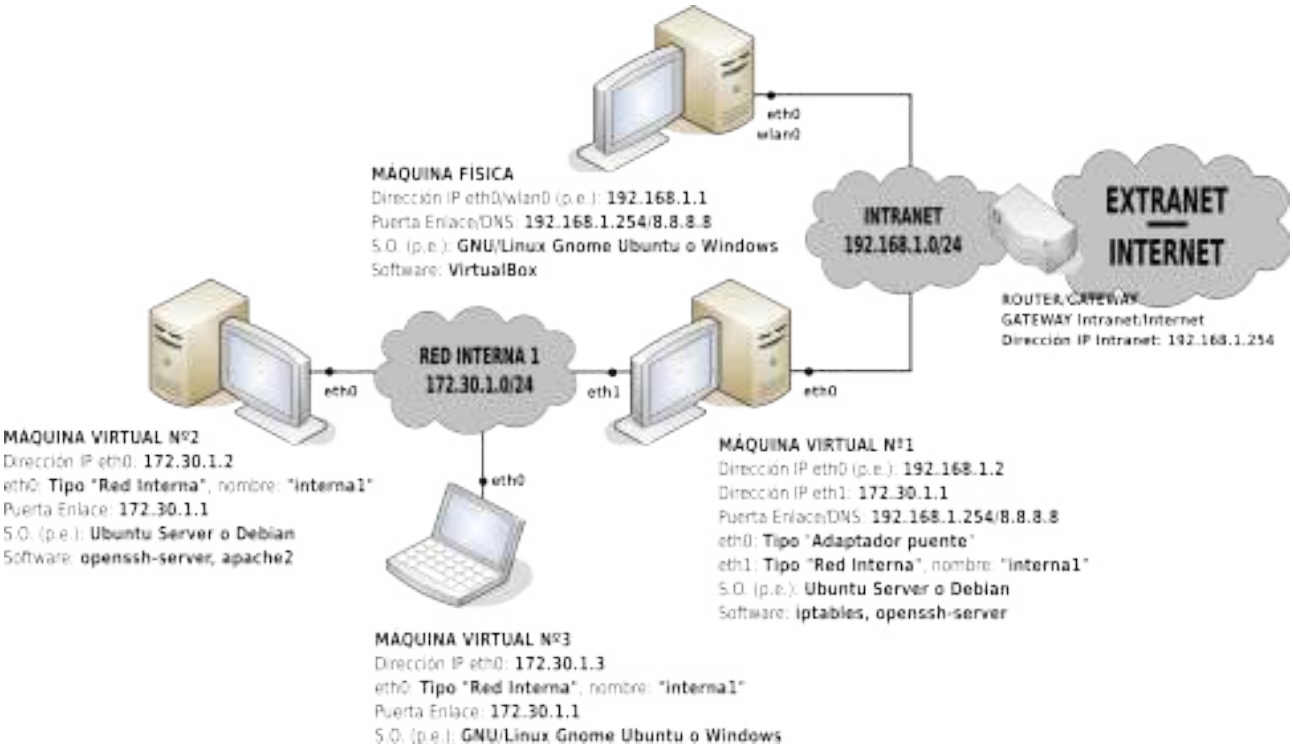
```
[root@mv2]# mdkir prueba
[root@mv2]# cd prueba
[root@mv2:prueba]# touch peli1.avi peli2.avi peli3.avi
[root@mv2:prueba]# echo "<h1 style='text-align: center; background-color: black; color: white;'>Sitio Web – Servido desde el puerto 8000</h1>" > index.html
[root@mv2:prueba]# python -m SimpleHTTPServer 8000
```

¡¡Aclaración!! Tener en cuenta que la **mv1**, con la configuración anterior, se encuentra totalmente aislada. Es decir, únicamente reenvía los servicios configurados, tcp/8000, udp/53 y icmp (ping), pero desde el equipo no se podría ni siquiera hacer un ping desde la propia máquina, ni recibirlo. Por ello, se proponen a continuación los siguientes ejercicios.

Ej. Práctico 2.3.5: Otro Ejemplo de Filtrado de Paquetes y Redireccionamiento

Añade un nuevo equipo en la Intranet del ejercicio práctico anterior, **mv3** con dirección IP 172.30.1.3, tal como se puede apreciar en el siguiente esquema de red, ¿qué reglas añadirías al script "**conf-red.sh**" de la **mv1** para cumplir los siguientes requisitos:

a) Reglas de firewall necesarias en la **mv1** para que **acepte solicitudes desde el exterior** *(interfaz eth0)* **al servicio SSH** para su control remoto (**tcp/22**).
b) Queriendo controlar igualmente por **SSH** la **mv2**, configurar las reglas de firewall necesarias para que este servicio sea accesible desde el exterior a través del puerto **tcp/22022**. Para ello será necesario un redireccionamiento de las peticiones recibidas por la mv1 a ese puerto hacía el puerto 22 de la mv2.
c) Permite que desde la **mv1** pueda controlarse vía SSH a la **mv2**.
d) Configura el firewall de la **mv1**, para posibilitar que desde la máquina virtual **mv3** (*máquina con un sistema operativo con escritorio, GNU/Linux o Windows*), se pueda navegar haciendo uso del servicio HTTP (**tcp/80**) y HTTPS (**tcp/443**), a excepción a modo de ejemplo, de los servicios ofrecidos por http://www.peliculasyonkis.com y https://www.facebook.com.



Solución Ej. Pr. 2.3.5.I.- Opciones de Redireccionamiento y Filtrado paquetes

Para dar solución al problema, continuaremos con el script del ejercicio anterior, "**conf-red.sh**":

```
[root@mv1]# nano /etc/init.d/conf-red.sh

#!/bin/bash
```

```
# Comenzaremos con la solución del ejercicio práctico anterior, al cual añadiremos la del actual:
/etc/init.d/networking stop
ifconfig eth0 192.168.1.2
ifconfig eth1 172.30.1.1 netmask 255.255.255.0
route add default gw 192.168.1.254
echo "nameserver 8.8.8.8" > /etc/resolv.conf
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -t nat -F
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 8000 -j DNAT --to 172.30.1.2
iptables -t filter -F
iptables -t filter -P INPUT DROP
iptables -t filter -P INPUT DROP
iptables -t filter -P INPUT DROP
iptables -t filter -A FORWARD -i eth0 -o eth1 -p tcp --dport 8000 -d 172.30.1.2 -j ACCEPT
iptables -t filter -A FORWARD -i eth1 -s 172.30.1.2 -o eth0 -p icmp -j ACCEPT
iptables -t filter -A FORWARD -i eth1 -s 172.30.1.2 -o eth0 -p udp --dport 53 -j ACCEPT
iptables -t filter -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
# Solución del ejercicio práctico:
# Apartado a): Permitimos el acceso al servicio SSH y los paquetes de vuelta
iptables -t filter -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
iptables -t filter -A OUTPUT -o eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT
# Apartado b): Redireccionamos las solicitudes procedentes del exterior al puerto 22022 hacia el
puerto 22 de la mv2, y posteriormente permitimos su reenvío
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 22022 -j DNAT --to 172.30.1.2:22
iptables -t filter -A FORWARD -i eth0 -o eth1 -p tcp --dport 22 -d 172.30.1.2 -j ACCEPT
# Apartado c): Permitimos conexiones SSH desde el propio firewall mv1 a la mv2, y los paquetes
de vuelta de respuesta
iptables -t filter -A OUTPUT -o eth1 -p tcp --dport 22 -d 172.30.1.2 -j ACCEPT
iptables -t filter -A INPUT -i eth1 -m state --state ESTABLISHED,RELATED -j ACCEPT
# Apartado d): Abrimos los puertos tcp/80 y tcp/443 para la mv3, a excepción de los sitios web
indicados en el enunciado.  Para poder hacer uso de nombres de dominio en la regla, deberemos
permitir que el firewall pueda realizar resoluciones de nombre DNS (udp/53)
iptables -t filter -A OUTPUT -o eth0 -p udp --dport 53 -j ACCEPT
iptables -t filter -A INPUT -i eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -t filter -A FORWARD -i eth1 -o eth0 -p tcp --dport 80 \
        -s 172.30.1.3 -d www.peliculasyonkis.com -j DROP
iptables -t filter -A FORWARD -i eth1 -o eth0 -p tcp --dport 443 \
        -s 172.30.1.3 -d www.facebook.com -j DROP
ipiptables -t filter -A FORWARD -i eth1 -o eth0 -p tcp -m multiport --dport 80,443 \
        -s 172.30.1.3 -j ACCEPT
```

¡¡Aclaración!! Tal como se puede observar en la última regla del script anterior, cuando queremos especificar una lista de puertos, y estos no son consecutivos, puede hacerse uso de la opción "-m multiport". En el caso de que fueran consecutivos, por ejemplo del 8000 al 8008, hubiera sido suficiente con poner "**--dport 8000:8008**".

Antes de comprobar su correcto funcionamiento, será necesario ejecutar el script anterior:

```
[root@mv1]# chmod +x /etc/init.d/conf-red.sh
[root@mv1]# bash /etc/init.d/conf-red.sh
```

¡¡Recordatorio!! Para que el script anterior se autoejecute al iniciarse la máquina, la forma más fácil es lanzarlo a través del script "**rc.local**" que se ejecuta al final del inicio de la máquina, añadiéndolo al final de dicho script (*doble redireccionamiento, ">>"*):

```
[root@mv1]# echo "bash /etc/init.d/conf-red.sh" >> /etc/init.d/rc.local
```

Para comprobar el correcto el funcionamiento del script de configuración realizaremos acciones similares a las llevadas a cabo en el ejercicio anterior. Por ejemplo:

- Probar a establecer conexiones **SSH** desde un equipo externo (*p.e. desde la máquina física*) tanto a la **mv1**, como a la **mv2**:

```
[root@mf]# ssh usuario_mv1@192.168.1.2
[root@mf]# ssh usuario_mv2@192.168.1.2 -p 22022
```

¡¡Observación!! Para la comprobación de este ejercicio se asume que en las mv1 y mv2 se ha instalado previamente el software que permite su control remoto vía SSH **openssh-server**:

```
[root@mv1|mv2]# apt-get install openssh-server
```

En el caso de que no haya sido instalado previamente, para su instalación será necesario deshacer la configuración del firewall, ya que las máquinas **mv1** y **mv2** requieren acceder a los repositorios de Internet para descargar e instalar el software **openssh-server**.

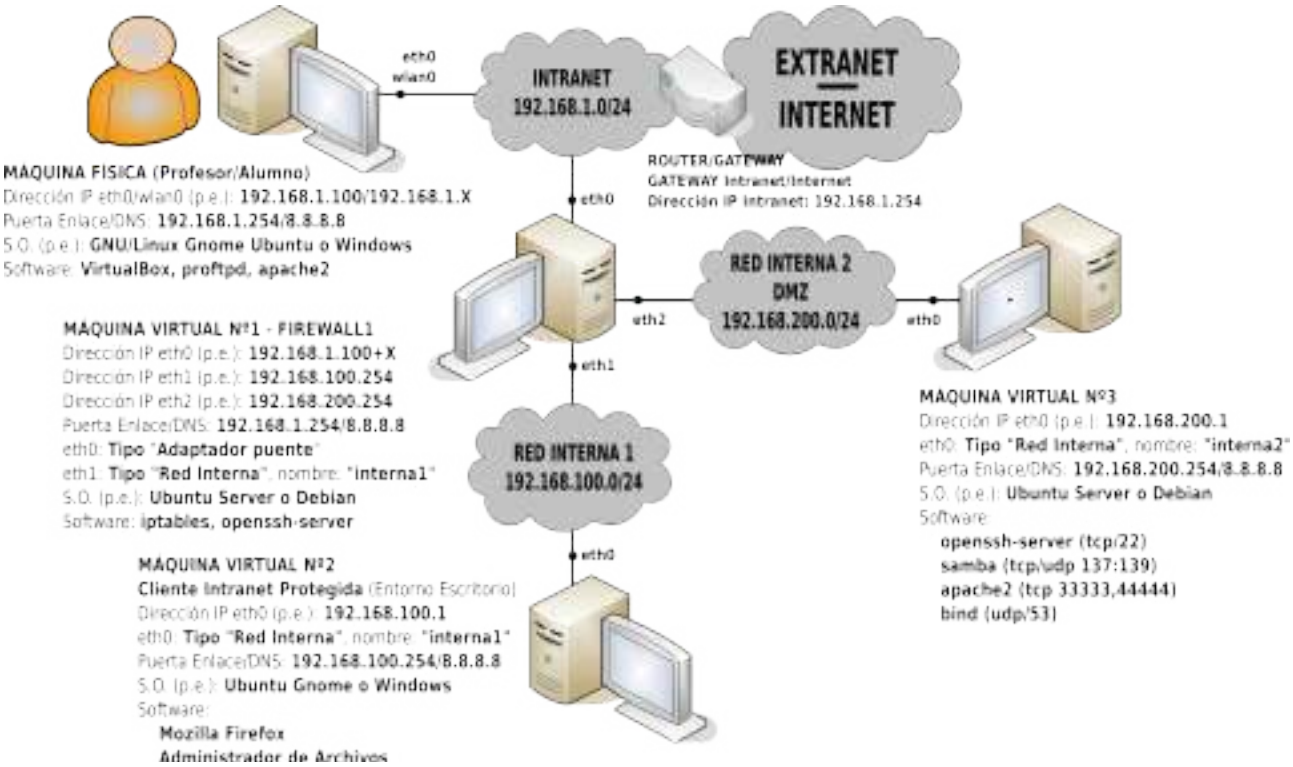
- Probar a establecer conexiones **SSH** desde la mv1 hacia la mv2:

```
[root@mv1]# ssh usuario_mv2@172.30.1.2
```

- Probar a navegar desde el equipo cliente mv3. Abrir un navegador, y comprobar que se puede acceder a cualquier sitio web, a excepción de los indicados en las reglas de filtrado.

Ej. Práctico 2.3.6: Seguridad Perimetral con Zona Desmilitarizada

Configura el siguiente esquema de red compuesto por un firewall (*máquina virtual Nº1, mv1*) encargado de proteger a una red interna (*máquina virtual Nº2, mv2*) y simultáneamente de dar acceso a los servicios ofrecidos en una zona desmilitarizada (*máquina virtual Nº3, mv3*) mediante la utilización de máquinas virtualizadas en VirtualBox. Las direcciones IP indicadas junto a los equipos del ejercicio práctico son unas a modo de ejemplo, cada usuario al configurarlas debería ser consciente de cuales podría utilizar.



En concreto, en este ejercicio práctico, a modo de ejemplo, deberán cumplirse las siguientes características:

A) Indica cuales serían los primeros comandos que deberían aparecer en el script de configuración de red (*script que se ejecutaría al arrancar la máquina*) para que el equipo firewall del esquema, **mv1**, se comporte como un correcto gateway, con una **política firewall por defecto** (*-t filter -P*) de **no aceptar nada que vaya dirigido al propio equipo**, **no permitir la salida de ningún paquete TCP/IP desde el propio equipo**, **ni permitir el reenvío de ningún paquete TCP/IP**.
B) ¿Qué comandos añadirías al script anterior **para permitir la respuesta a pings procedentes tanto desde tu máquina física, como desde el equipo de tu profesor** (*p.e. 192.168.1.X y 192.168.1.100, donde X representa el número de equipo donde habitualmente trabajas*)?
C) ¿Qué comandos añadirías al script anterior para **permitir el envío de pings desde la mv2** (*cliente protegido de la Intranet*) **hacía Internet haciendo uso de nombres de dominio** (*p.e. ping www.google.com*)?
D) ¿Qué comandos añadirías al script anterior para **permitir el control remoto vía SSH desde el exterior** (*desde tu máquina física y la del profesor únicamente*) tanto de la **mv1** (*firewall*), como de

la **mv3** (*servidor DMZ*)? Tener en cuenta que para controlar vía **SSH tcp/22** la **mv3** se hará a través de la **mv1** a través de su puerto **tcp/22222**.
E) ¿Qué comandos añadirías al script anterior para permitir que los equipos de la red interna Nº1 **puedan hacer uso de los protocolos HTTP y HTTPS a excepción del acceso a sitios Web como http://www.sport.es y https://www.facebook.com**? Advierte que para que se pueda hacer uso de nombres de dominio en la configuración de las reglas que impongas en el firewall, el propio equipo firewall deberá poder traducirlos a su correspondiente dirección IP, por lo que deberás permitirle resoluciones **DNS udp/53**.
F) ¿Qué comandos añadirías al script anterior para que desde el exterior sea accesible a través del router un servicio HTTP que se ofrece por el puerto **tcp/33333**?
G) ¿Qué comandos añadirías al script anterior para que desde el exterior (*únicamente desde tu máquina física y la del profesor*) sea accesible un directorio que se comparte de manera pública mediante **Samba tcp/udp/137:139** en la **mv3** llamado "misdocX" (*p.e. /media/documentos*)?
H) ¿Qué comandos añadirías al script anterior para que desde el exterior sea accesible un servicio HTTP que se ofrece en el equipo servidor **mv3** vía **tcp/44444**, aunque a través de la **mv1** vía **tcp/50000**?

Solución Ej. Pr. 2.3.6.I.- Configuración de un Firewall con una DMZ

Al igual que en los ejercicios anteriores, la solución se dará en formato de script, el cual se ejecutará al iniciarse la máquina, o cuando nosotros deseemos ejecutarlo desde una terminal:

[root@mv1]# **nano /etc/init.d/conf-red.sh**

```
#!/bin/bash
## Solución apartado A):
# Paramos el servicio networking para permitir la configuración mediante nuestro script:
/etc/init.d/networking stop
# Configuramos las direcciones IP, puerta de enlace y servidor DNS preferido
ifconfig eth0 192.168.1.100
ifconfig eth1 192.168.100.100
ifconfig eth2 192.168.200.100
route add default gw 192.168.1.254
echo "nameserver 8.8.8.8" > /etc/resolv.conf
# Activamos el forwarding y el MASQUERADE para que actúe como un gateway
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -t nat -F
iptables -t nat -A POSTROUTING -j MASQUERADE
# Indicamos la políticas por defecto: DROP. De esta forma el equipo queda totalmente aislado, con todos sus puertos cerrados
iptables -t filter -F
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
## Solución apartado B):
# Permitimos que nos puedan hacer un ping desde el exterior a las máquinas físicas del profesor y
```

del alumno (X es el número de equipo)
iptables -A INPUT -p icmp -i eth0 -s 192.168.1.100 -j ACCEPT
iptables -A INPUT -p icmp -i eth0 -s 192.168.1.*X* -j ACCEPT
 # Aceptamos igualmente los paquetes de vuelta de respuesta ante el ping anterior
**iptables -A OUTPUT -o eth0 -d 192.168.1.100 **
-m state --state ESTABLISHED,RELATED -j ACCEPT
**iptables -A OUTPUT -o eth0 -d 192.168.1.*X* **
-m state --state ESTABLISHED,RELATED -j ACCEPT
Solución apartado C):
 # Permitimos la resolución de nombres a la mv2 para poder hacer pings a nombres de dominio
iptables -A FORWARD -p udp --dport 53 -i eth1 -o eth0 -j ACCEPT
iptables -A FORWARD -p icmp -i eth1 -o eth0 -j ACCEPT
 # Permitimos los paquetes de respuesta al servicio ping y DNS anteriores
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -i eth0 -o eth1 -j ACCEPT
Solución apartado D):
 # Permitimos el control vía SSH de la mv1 desde el exterior (máquinas físicas profesor y alumno)
iptables -A INPUT -p tcp --dport 22 -s 192.168.1.100 -j ACCEPT
iptables -A INPUT -p tcp --dport 22 -s 192.168.1.*X* -j ACCEPT
 # Permitimos salir los paquetes de respuesta a la solicitud de conexión SSH anterior
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
 # Redireccionamos las peticiones externas tcp/22222 al puerto SSH tcp/22 de la mv3
**iptables -t nat -A PREROUTING -i eth0 -s 192.168.1.100 **
-p tcp --dport 22222 -j DNAT --to 192.168.200.1:22
**iptables -t nat -A PREROUTING -i eth0 -s 192.168.1.*X* **
-p tcp --dport 22222 -j DNAT --to 192.168.200.1:22
 # Permitimos que atraviesen a la mv1 las solicitudes de conexión SSH tcp/22 redireccionadas hacia la mv3
iptables -t filter -A FORWARD -i eth0 -o eth2 -p tcp --dport 22 -d 192.168.200.1 -j ACCEPT
 # Permitimos que atraviesen a la mv1 los paquetes de respuesta procedentes de la mv3 ante la conexión SSH tcp/22 permitida en las reglas anteriores
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -i eth2 -o eth0 -j ACCEPT
Solución apartado E):
 # Permitimos que la mv1 resuelva los nombres de dominio que utilizarán a continuación en sus reglas de filtrado
iptables -A OUTPUT -p udp --dport 53 -o eth0 -j ACCEPT
 # Permitimos los paquetes de vuelta de respuesta ante la solicitud de resolución DNS anterior
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
 # Restringimos el acceso a la mv2 a determinados sitios Web
iptables -A FORWARD -p tcp --dport 80 -d www.sport.es -i eth1 -o eth0 -j DROP
iptables -A FORWARD -p tcp --dport 443 -d www.facebook.com -i eth1 -o eth0 -j DROP
 # Permitimos el acceso a cualquier otro sitio Web mediante los protocolos HTTP y HTTPS
iptables -A FORWARD -p tcp --dport 80 -i eth1 -o eth0 -j ACCEPT
iptables -A FORWARD -p tcp --dport 443 -i eth1 -o eth0 -j ACCEPT
 # Permitimos que atraviesen a la mv1 los paquetes de vuelta de las conexiones HTTP y HTTPS
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -i eth0 -o eth1 -j ACCEPT
Solución apartado F):
 # Redireccionamos las solicitudes de conexión externas al puerto 33333 a la mv3, y dejamos que

```
atraviesen (FORWARD) a la mv1 tanto esas solicitudes como sus paquetes de vuelta asociados
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 33333 -j DNAT --to 192.168.200.1
iptables -t filter -A FORWARD -i eth0 -o eth2 -p tcp --dport 33333 \
                -d 192.168.200.1 -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -i eth2 -o eth0 -j ACCEPT
## Solución apartado G):
# Lo mismo que el anterior pero para el servicio Samba tcp/udp/137:139
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 137:139 -j DNAT --to 192.168.200.1
iptables -t nat -A PREROUTING -i eth0 -p udp --dport 137:139 -j DNAT --to 192.168.200.1
iptables -t filter -A FORWARD -i eth0 -o eth2 -p tcp --dport 137:139 \
                -d 192.168.200.1 -j ACCEPT
iptables -t filter -A FORWARD -i eth0 -o eth2 -p udp --dport 137:139 \
                -d 192.168.200.1 -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -i eth2 -o eth0 -j ACCEPT
## Solución apartado H):
# Redireccionamos peticiones tcp/50000 recibidas por la mv1 hacía la mv3 vía tcp/44444
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 50000 \
                -j DNAT --to 192.168.200.1:44444
iptables -t filter -A FORWARD -i eth0 -o eth2 -p tcp --dport 44444 \
                -d 192.168.200.1 -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -i eth2 -o eth0 -j ACCEPT
```

Antes de comprobar su correcto funcionamiento, será necesario ejecutar el script anterior:

```
[root@mv1]# chmod +x /etc/init.d/conf-red.sh
[root@mv1]# /etc/init.d/conf-red.sh
```

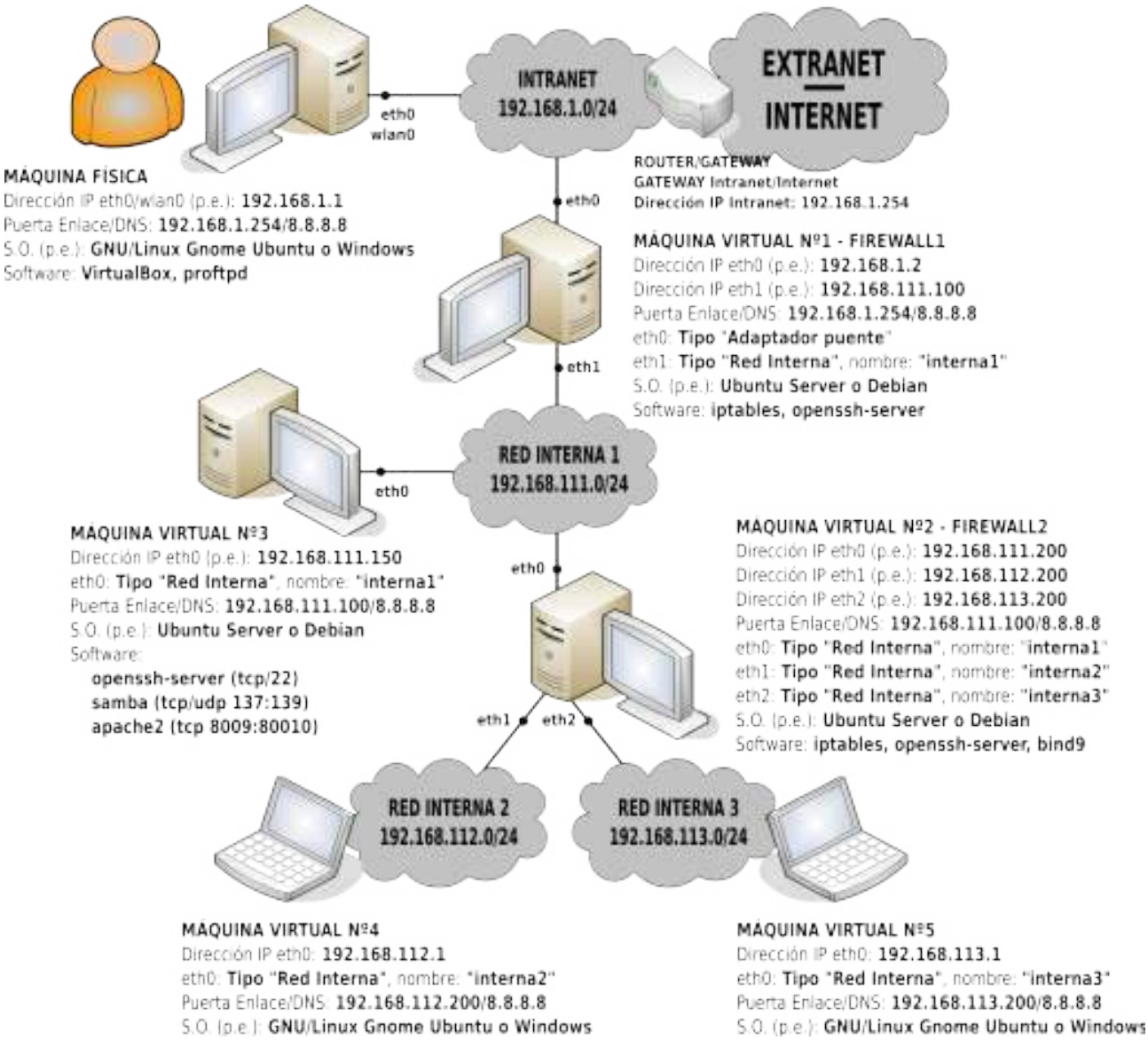
¡¡Recordatorio!! Para que el script anterior se autoejecute al iniciarse la máquina, la forma más fácil es lanzarlo a través del script "**rc.local**" que se ejecuta al final del inicio de la máquina, añadiéndolo al final de dicho script (*doble redireccionamiento, ">>"*):

```
[root@mv1]# echo "bash /etc/init.d/conf-red.sh" >> /etc/init.d/rc.local
```

Por último, deberíamos hacer las correspondientes comprobaciones del correcto comportamiento del firewall tal como ha hecho en ejercicios anteriores, o tal como se muestra en el siguiente ejercicio de una forma muy detallada.

Ej. Práctico 2.3.7: Seguridad Perimetral con doble Firewall

Configura el siguiente esquema de red compuesto por un doble firewall, una zona desmilitarizada y dos redes internas protegidas, mediante la utilización de máquinas virtualizadas en VirtualBox (*obviamente no es necesario que todas las máquinas virtuales están activas simultáneamente, sino que dependiendo de las pruebas a realizar se activarán unas u otras*), y a continuación configura los firewall. Advierte que el segundo firewall dispone de tres interfaces de red las cuales deben configurarse en VirtualBox como red interna (*interna1, interna2 e interna3 respectivamente*).



En concreto, las requisitos a cumplir serán los siguientes:

a) Incluye en los scripts de configuración de los dos firewall los comandos necesarios para programar como **política por defecto el no aceptar nada que vaya dirigido a los propios firewall, el no permitir la conexión desde el propio firewall a cualesquiera que sea el servicio, y**

no reenviar nada entre las diferentes interfaces de red del firewall, aunque **permite momentáneamente el uso de ping** para realizar las correspondientes pruebas de conectividad, y posibilita el control remoto desde la máquina física vía SSH de las máquinas virtuales **mv1 vía SSH tcp/22, mv2 vía SSH tcp/22002 y mv3 vía SSH tcp/22003**.

b) Incluye en los scripts de configuración de los firewall los comandos necesarios para permitir acceder desde las redes internas protegidas a los siguientes servicios:

b1) **Permitir el acceso desde las redes internas 2 y 3 al servicio HTTP tcp/80** que se ofrece en el exterior, Internet. Lo deberás comprobar accediendo con éxito por ejemplo a "www.google.com", pero sin éxito a "www.gmail.com" al ser HTTPS.
b2) **Permitir el acceso desde la red interna 2, pero no desde la 3, al servicio FTP** que se ofrece en el equipo de la Extranet **192.168.1.1 por el puerto tcp/udp/21012**. Para ofrecer este servicio tan sólo es necesario instalar por ejemplo "**proftpd**" (*apt-get install proftpd*) y añadir en su archivo de configuración "**proftpd.conf**" la directiva "**Port 21012**". Deberás comprobar que desde un equipo de la red interna 2 se accede, pero no desde la 3 con un cliente FTP como puede ser el propio administrador de archivos del sistema.
b3) **Permitir el acceso desde la red interna 3, pero no desde la 2, al servicio HTTP** que se ofrece en el equipo de la Extranet **192.168.1.1 por el puerto tcp/8888**. Para ofrecer este servicio se puede instalar el software "**apache2**" (*apt-get install apache2*), y en su archivo de configuración "ports.conf" añadir la directiva "**Listen 8888**", o simplemente ejecutar el comando "**python -m SimpleHTTPServer 8888**" desde el directorio a servir. Deberás comprobar que desde un equipo de la red interna 3 se accede, pero no desde la 2 con un cliente HTTP como es el navegador Web mozilla o chrome.
b4) **Permitir el acceso únicamente al equipo de la Intranet 192.168.112.1 al servicio HTTPS tcp/443** que se ofrece en Internet. Comprueba el correcto funcionamiento desde ese equipo conectándote a "www.gmail.com", verificando igualmente que si su dirección IP se modifica por ejemplo a 192.168.112.2 ya no accede.

c) Configura el **firewall1** para que desde el exterior (*p.e. desde la máquina física*) pueda accederse a los servicios **Samba tcp/udp/137:139** y **HTTP tcp/8009** que se ofrecen a través del equipo servidor que se localiza en la DMZ, teniendo en cuenta que desde el exterior el único equipo que es visible es el propio **gateway/firewall1**. Para ofrecer los servicios anteriores en la **mv3** pueden llevarse a cabo las siguientes acciones:

- Para compartir un directorio llamado "compartir" mediante el nombre "recurso1" con permisos de sólo lectura de manera pública a través de Samba, podríamos llevar a cabo las siguientes acciones:

```
[root@mv3]# apt-get install samba
[root@mv3]# mkdir /mnt/compartir
[root@mv3]# cd /mnt/compartir
[root@mv3:/mnt/compartir]# touch peli1.avi peli2.avi peli3.avi
[root@mv3:/mnt/compartir]# nano /etc/samba/smb.conf
#Añadimos al comienzo del archivo "smb.conf" la siguiente sección:
[recurso1]
path = /mnt/compartir
browseable = yes
```

```
read only = yes
public = yes
guest ok = yes
```

```
[root@mv3]# testparm
[root@mv3]# /etc/init.d/smbd restart
```

- Para ofrecer el servicio HTTP tcp/8009 podemos instalar y configurar "apache2", o de una manera mucho más simple, haciendo uso del módulo de python "SimpleHTTPServer". En el siguiente ejemplo, daríamos servicio al contenido de la carpeta creada anteriormente:

```
[root@mv3:/mnt/compartir]# python -m SimpleHTTPServer 8009
```

Podrás comprobar su correcto funcionamiento desde la máquina física accediendo a los servicios anteriores.

d) Configura el **firewall1** para que desde el exterior (*p.e. desde la máquina física*) pueda accederse a un segundo servicio **HTTP** (**tcp/8010**) que se ofrece en el interior de la DMZ, teniendo en cuenta que desde el exterior el único equipo que es visible es el propio gateway/firewall1, y que para acceder a él se accederá por un puerto diferente, el **tcp/8000** *(http://192.168.1.2:8000)*. Para ofrecer dicho servicio, al igual que en el ejercicio anterior podríamos hacer lo siguiente:

```
[root@mv3]# mkdir /mnt/compartir2
[root@mv3]# cd /mnt/compartir2
[root@mv3:/mnt/compartir2]# echo "<h1 style='text-align: center; background-color: pink; color: blue;'>Sitio Web – Accesible por el puerto 8000 – Servido por el 8010</h1>" > index.html
[root@mv3:/mnt/compartir2]# python -m SimpleHTTPServer 8010
```

Una vez configurado el correspondiente redireccionamiento en el firewall1 se podrá comprobar su correcto funcionamiento desde una máquina externa escribiendo en la barra de direcciones del navegador "*http://192.168.1.2:8000*".

e) Configura el **firewall2** teniendo en cuenta que en ese equipo se ofrece un servicio **DNS caché udp/53**, de tal forma que puedan solicitar resolución de nombres de dominio al servicio tanto desde el equipo del exterior 192.168.1.1 a través del firewall1 (*será necesario configurar el correspondiente redireccionamiento de puertos*) y desde las redes internas protegidas, pero a modo de ejemplo, no desde la DMZ. Para ofrecer este servicio en la mv3 tan sólo será necesario instalar el software servidor "**bind9**", y convertirá a nuestro equipo en un servidor DNS caché:

```
[root@mv3]# apt-get install bind9
```

Para comprobar su correcto funcionamiento, puede hacer desde el propio equipo servidor mediante la herramienta cliente DNS "**dig**" disponible en el paquete "**dnsutils**":

```
[root@mv3]# apt-get install dnsutils
[root@mv3]# dig @127.0.0.1 www.google.com
```

Una vez configurado los firewall se podrá comprobar su correcto funcionamiento desde el resto de máquinas virtuales, indicando como DNS preferido la dirección IP de la **mv3** si estas se encuentran en las redes internas protegidas, o la dirección IP externa del firewall1 si queremos comprobarlo desde una máquina externa (*el firewall1 la redireccionará a la mv3*).

Solución Ej. Pr. 2.3.7.I.- Configuración de una Zona Perimetral compuesta de un doble Firewall – DMZ – Intranet Protegida

Al igual que en los ejercicios anteriores, la solución se dará en formato script. A continuación se muestran los scripts que habría que ejecutar en ambos firewall, **mv1** y **mv2**:

```
[root@mv1]# nano /etc/init.d/conf-red.sh
#!/bin/bash
# Paramos el servicio networking para permitir la configuración mediante nuestro script:
/etc/init.d/networking stop
# Configuramos las direcciones IP, puerta de enlace y servidor DNS preferido
ifconfig eth0 192.168.1.2
ifconfig eth1 192.168.111.100
route add default gw 192.168.1.254
echo "nameserver 8.8.8.8" > /etc/resolv.conf
# Activamos el forwarding y el MASQUERADE para que actúe como un gateway
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -t nat -F
iptables -t nat -A POSTROUTING -j MASQUERADE
# Solución a):
# Indicamos la políticas por defecto
iptables -t filter -F
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
# Permitimos el control remoto de mv1, mv2 y mv3 desde la máquina física 192.168.1.1
iptables  -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 22002 -j DNAT --to 192.168.111.200:22
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 22003 -j DNAT --to 192.168.111.150:22
iptables -t filter -A FORWARD -i eth0 -o eth1 -p tcp --dport 22 -d 192.168.111.200 -j ACCEPT
iptables -t filter -A FORWARD -i eth0 -o eth1 -p tcp --dport 22 -d 192.168.111.150 -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
# Permitimos el uso de ping para pruebas de conectividad
iptables -A INPUT -p icmp -j ACCEPT
iptables -A OUTPUT -p icmp -j ACCEPT
iptables -A FORWARD -p icmp -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
# Para permitir hacer el ping a nombres de dominio, y no sólo a direcciones ip
iptables -A OUTPUT -o eth0 -p udp --dport 53 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -p udp --dport 53 -j ACCEPT
# Solución b):
```

```
# Permitimos el acceso al servicio HTTP que se ofrece en Internet
iptables -t filter -A FORWARD -i eth1 -o eth0 -p tcp --dport 80 -j ACCEPT
# Permitimos el acceso al servicio FTP tcp/udp/21012 y HTTP tcp/8888 que se ofrece en la máquina física
iptables -A FORWARD -i eth1 -o eth0 -p tcp -d 192.168.1.1 --dport 21012 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -p udp -d 192.168.1.1 --dport 21012 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -p tcp --dport 8888 -d 192.168.1.1 -j ACCEPT
# Permitimos el acceso al servicio HTTPS procedente de la máquina 192.168.112.1, aunque esta máquina no es distinguible desde el firewall1 a consecuencia del MASQUERADE ejercido por el firewall2
iptables -A FORWARD -i eth1 -o eth0 -p tcp --dport 443 -j ACCEPT
# Solución c):
# Redireccionamiento y acceso al servicio Samba tcp/udp/137:139 ofrecido por el servidor de la DMZ desde el exterior
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 137:139 -j DNAT --to 192.168.111.150
iptables -t nat -A PREROUTING -i eth0 -p udp --dport 137:139 -j DNAT --to 192.168.111.150
iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 137:139 -d 192.168.111.150 -j ACCEPT
iptables -A FORWARD -i eth0 -o eth1 -p udp --dport 137:139 -d 192.168.111.150 -j ACCEPT
# Redireccionamiento y acceso al servicio HTTP tcp/8009 ofrecido por el servidor de la DMZ desde el exterior
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 8009 -j DNAT --to 192.168.111.150
iptables -A FORWARD -i eth0 -o eth1 -p tcp --dport 8009 -d 192.168.111.150 -j ACCEPT
# Redireccionamiento de puertos
# Solución d):
# Redireccionamiento y acceso al servicio ofrecido por el servidor desde la DMZ por el puerto 8010 accesible desde el exterior a través del firewall1 por el puerto 8000
iptables -t nat -A PREROUTING -i eth0 \
        -p tcp --dport 8000 -j DNAT --to 192.168.111.150:8010
iptables -A FORWARD -i eth0 -o eth1 -d 192.168.111.150 -p tcp --dport 8010 -j ACCEPT
# Solución e):
# Redireccionamiento y acceso al servicio DNS udp/53 ofrecido por el firewall2
iptables -t nat -A PREROUTING -i eth0 -p udp --dport 53 -j DNAT --to 192.168.111.200
iptables -A FORWARD -i eth0 -o eth1 -d 192.168.111.200 -p udp --dport 53 -j ACCEPT
```

En relación al script de configuración del segundo firewall, **mv2** (*al ser similar al script anterior sólo se comentará lo más básico*):

```
[root@mv2]# nano /etc/init.d/conf-red.sh
#!/bin/bash
/etc/init.d/networking stop
ifconfig eth0 192.168.111.200
ifconfig eth1 192.168.112.200
ifconfig eth2 192.168.113.200
route add default gw 192.168.111.100
echo "nameserver 8.8.8.8" > /etc/resolv.conf
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -t nat -F
```

```
iptables -t nat -A POSTROUTING -j MASQUERADE
# Solución a):
iptables -t filter -F
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
iptables -A INPUT -p icmp -j ACCEPT
iptables -A OUTPUT -p icmp -j ACCEPT
iptables -A FORWARD -p icmp -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --dport 53 -j ACCEPT
iptables -A FORWARD -o eth0 -p tcp --dport 53 -j ACCEPT
# Permiso de acceso al servicio SSH
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
# Solución b):
iptables -t filter -A FORWARD -o eth0 -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -p tcp --dport 21012 -d 192.168.1.1 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -p udp --dport 21012 -d 192.168.1.1 -j ACCEPT
iptables -A FORWARD -i eth2 -o eth0 -p tcp --dport 8888 -d 192.168.1.1 -j ACCEPT
iptables -A FORWARD -s 192.168.112.1 -o eth0 -p tcp --dport 443 -j ACCEPT
# Solución e):
iptables -A INPUT -i eth0 -p udp --dport 53 -s 192.168.111.150 -j DROP
iptables -A INPUT -i eth0 -p udp --dport 53 -j ACCEPT
```

Para comprobar su correcto funcionamiento, tan sólo será necesario ejecutar los anteriores scripts:

```
[root@mv1|mv2]# chmod +x /etc/init.d/conf-red.sh
[root@mv1|mv2]# bash /etc/init.d/conf-red.sh
```

¡¡Recordatorio!! Para que el script anterior se autoejecute al iniciarse la máquina, la forma más fácil es lanzarlo a través del script "**rc.local**" que se ejecuta al final del inicio de la máquina, añadiéndolo al final de dicho script (*doble redireccionamiento, ">>"*):

```
[root@mv1|mv2]# echo "bash /etc/init.d/conf-red.sh" >> /etc/init.d/rc.local
```

Ej. Práctico 2.3.8: Implementación de Firewall mediante Zentyal

Como se ha visto a lo largo de los ejercicios prácticos anteriores, **iptables** es el software base en el diseño de firewall mediante el uso de sistemas operativos GNU/Linux. Con la finalidad de facilitar la gestión de **iptables** y no tener que aprenderse su sintaxis, existen herramientas software a un nivel de abstracción superior, entre las cuales cabría destacar "**webmin**" y "**Zentyal**", con un entorno mucho más amigable para el usuario que tiene que implementar un firewall. Debido al abanico tan grande de posibilidades que ofrece Zentyal (P*roxy HTTP, Active Directory, Gestor de correo, Servidor WebMail, etc.*), desde aquí se invita a probarlo en una de sus pequeñas facetas, la de firewall o cortafuegos, que cubre perfectamente.

Con la finalidad de dar a conocer esta distribución de GNU/Linux, en este ejercicio práctico se anima a implementar los tres ejercicios prácticos anteriores, 2.3.1, 2.3.2 y 2.3.3, sustituyendo los equipos firewall basados en Ubuntu Server o Debian, por equipos funcionado bajo Zentyal.

***Solución Ej. Pr. 2.3.8.I.**- Diseño de cortafuegos mediante Zentyal*

Respecto a Zentyal, comentar que se trata de una distribución más de GNU/Linux, basada en Ubuntu Server, pensada para la implementación de todo tipo de servicios, caracterizada por una gestión vía Web "*https://dirección_ip_interfaz_interna_zentyal*".

Para probar Zentyal crea una nueva máquina virtual, tal como si fueras a instalar en ella un Ubuntu Server, aunque con la premisa de que habrá que asignarle más memoria RAM (*al menos 1GB para que funcione de manera aceptable*) ya que esta más sobrecargado.

Un aspecto importante a tener en cuenta durante la instalación de Zentyal es la asignación de la interfaz de red de Zentyal que va a ser primaria. Es decir, la interfaz de red que da acceso al exterior. Este aspecto es sumamente importante para Zentyal, ya que a dicha interfaz, Zentyal le asocia políticas de seguridad mucho más severas, no permitiendo ni siquiera lanzarle un ping a dicha interfaz, relajando la seguridad sobre el resto de interfaces.

A continuación, de una manera bastante somera se explicarán los pasos de instalación y configuración del firewall de Zentyal. No se entrará en mayor profundidad, ya que Zentyal dispone de una interfaz muy amigable y sencilla de asimilar por cualquiera.

1) En al creación de una máquina virtual mediante VirtualBox para Zentyal seleccionaremos como tipo de sistema a "Linux", y como distribución a "Ubuntu", ya que como ya se ha dicho previamente, Zentyal no es más que el resultado de una personalización de Ubuntu Server.

2) Le asignaremos al menos 1GB de memoria RAM para que Zentyal vaya suficientemente fluido, a diferencia de Ubuntu Server o Debian (*sin interfaz gráfica*), que como se habrá podido advertir mediante la implementación práctica de los ejercicios prácticos propuestos en el presente capítulo, con 256MB desempeñaban su función perfectamente. La culpa la tiene la interfaz gráfica y módulos con los cuenta Zentyal.

3) Durante la instalación de Zentyal, se aconseja que el primer adaptador de cada una de las máquinas virtuales donde vayamos a instalar Zentyal sea de tipo NAT, lo que garantiza que la configuración de red será automática, y Zentyal dispondrá de conexión a Internet por si requiriera descargar algún paquete.

4) Hecho lo anterior, podremos pasar a la instalación de Zentyal.

5) Una vez terminada la instalación, accederemos al escritorio de Zentyal. De manera automática se iniciará su navegador Web, ya que toda la administración de Zentyal esta pensada para realizarse vía HTTPS (tcp/443). Tras introducir el nombre de usuario y contraseña que se indicaron durante la instalación accederemos al "DashBoard" de Zentyal.

6) La primera vez que accedamos a la administración de Zentyal, este nos sugerirá que indiquemos el rol que deseamos para nuestro Zentyal: Gateway, Infraestructure, etc. Al elegir uno de estos roles predefinidos se nos instalarán todos los paquetes y módulos que Zentyal considera necesarios para llevar a cabo dicho rol. Como en nuestro caso, únicamente queremos hacerle trabajar como firewall, supliendo de esta forma a nuestros Ubuntu Server o Debian de los ejercicios prácticos, nos saltaremos esta instalación. Esta opción de "**Saltar la Instalación**" la encontraremos al final, debajo de todos los módulos disponibles.

7) Una vez ya en el "**DashBoard**" de Zentyal, observaremos que disponemos de un menú izquierdo con un item llamado "**Gestión de Software**". Pincharemos sobre él, ya que desde allí podremos seleccionar de manera individualizada los módulos que deseamos instalar en nuestros Zentyal.

8) En concreto, instalaremos los siguientes módulos: "**Firewall**", "**Network Configuration**", "**Network Objects**" y "**Network Services**", necesarios todos ellos para poder configurar el entorno de red de nuestro Zentyal a nuestro gusto, y poder configurar su firewall (*por debajo hace uso de iptables, aunque de manera transparente para el usuario que lo administra*).

9) Una vez instalados los módulos anteriores, será necesario habilitarlos, ya que por defecto no se habilitan. Para ello pincharemos al item "**Estado de los Módulos**" que encontraremos en el menú de la izquierda de nuestro "DashBoard". Después deberemos pinchar en el botón "**Guardar cambios**" que se localiza en la parte superior derecha para que surtan efecto los cambios.

10) Ahora, si pinchamos sobre el item "**Red**", podremos advertir que se despliegan diferentes subitems que nos permitirán personalizar la configuración de nuestras interfaces, puertas de enlace o servidores DNS preferidos. En concreto, si deseamos llevar a cabo el ejercicio práctico que aquí

se propone substituyendo los equipos firewall de los ejercicios anteriores por máquinas Zentyal, será necesario modificar el tipo de la primera interfaz de red de nuestro equipo Zentyal, de NAT a adaptador puente o red interna, según corresponda, posteriormente asignarles de manera manual las direcciones IP deseadas.

11) Tras configurar la red, pasaremos a configurar el firewall. Para ello, accederemos al menú "Gateway" que encontraremos a la izquierda y pincharemos sobre "**Cortafuegos**". Podremos ver que se despliegan tres posibles subitems, tal como era de esperar: a) "**Filtrado de paquetes**", desde el cual podremos personalizar reglas de filtrado de entrada, salida y forward de nuestro Zentyal, b) "**Redirecciones de puertos**", desde donde podremos configurar los redireccionamientos de tipo PREROUTING los cuales nos permitirán acceder a servicios de la Intranet desde la Extranet a través de Zentyal, y c) "**SNAT**", la cual nos permitirá configurar la nat POSTROUTING.

12) Si pinchamos en "**Filtrado de paquetes**", Zentyal nos mostrará de una manera muy amigable y sencilla mediante el uso de imágenes los tipos de filtrado que nos ofrece. Como podrá advertirse pinchando sobre cualquier de sus opciones, su configuración es muy fácil e intuitiva.

13) Si pinchamos en "**Redirecciones de puertos**", nos permitirá elegir los mismos parámetros que introducíamos cuando programábamos mediante iptables una regla de nat PREROUTING.

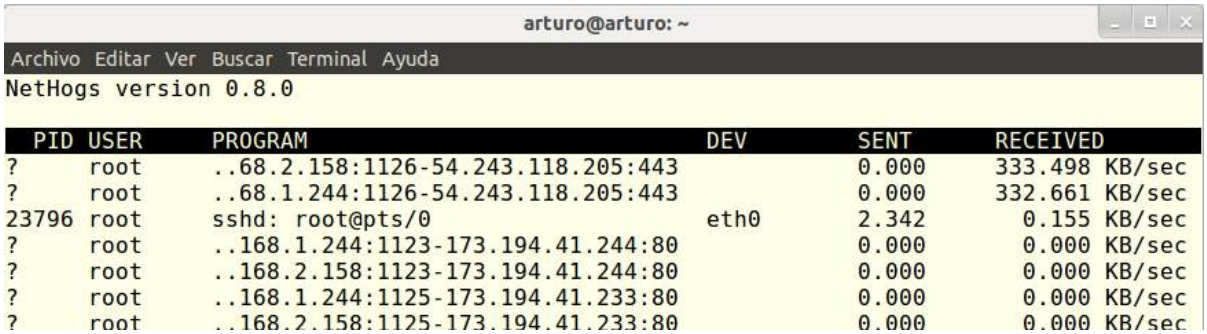
Ej. Práctico 2.3.9: Monitorización del Tráfico de nuestra Red

Tras mostrar a través de los ejercicios anteriores como configurar un firewall dentro de una red, en el siguiente ejercicio se mostrará un par de opciones muy sencillas y básicas de inspeccionar o monitorizar los servicios que son utilizados por les equipos clientes de la red. Aunque existen multitud se diferentes software para hacerlo, por sencillez, aquí se han elegido **nethogs**, **iptraf** y **ntop**.

Solución Ej. Pr. 2.3.9.I.- Monitorización vía Web del Tráfico de Red: ntop

Para realizar la monitorización del tráfico de red empezaremos instalando en el equipo que hace las veces de gateway y firewall el software adecuado: **nethogs**, **iptraf** o **ntop**.

(1ª opción) Instalar **nethogs**. Herramienta supersencilla y útil que nos permitirá monitorizar e identificar en tiempo real el ancho de banda que esta siendo consumido por los diferentes equipos de nuestra red. En concreto, nos permitirá conocer a que servicio están conectados y que cantidad de ancho de banda (*cantidad de bytes enviados y recibidos*) están consumiendo.

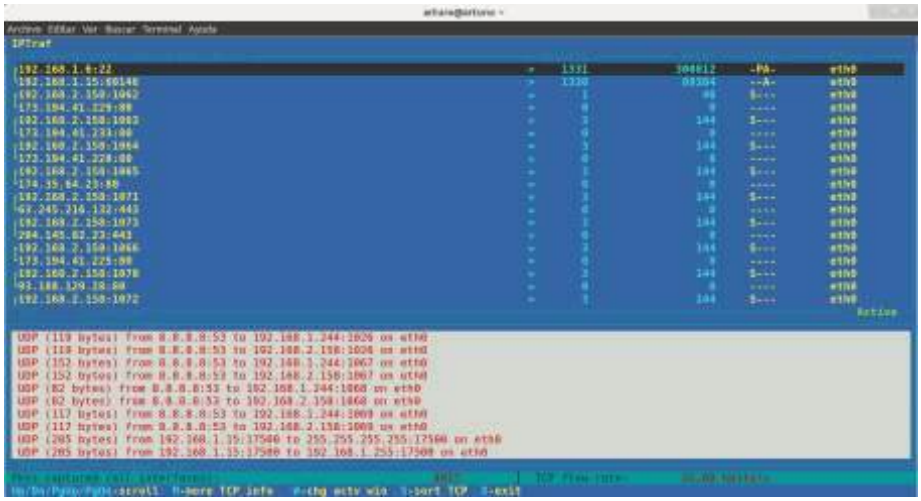


Al tratarse de un software ejecutable directamente desde la consola, lo más recomendable es conectarse remotamente vía ssh desde un equipo cliente al firewall y ejecutarlo:

```
[root@equipored]# ssh root@direccion_ip_gateway/firewall
[root@firewall]# apt-get install nethogs
[root@firewall]# nethogs
```

(2ª opción) Instalar **iptraf**. Al igual que **nethogs**, se caracteriza por ser un software ligero y eficiente que consume pocos recursos del equipo gateway al ser un software de consola, aunque algo más sofisticado que el presentado como primera opción, al permitirnos filtrar protocolos a monitorizar, interfaces de red, etc. Es decir, para monitorizar el tráfico de red deberemos ejecutar desde la consola del equipo firewall el comando **iptraf**, o desde un equipo de la red conectarnos vía **ssh** y a continuación ejecutar **iptraf**.

```
[root@equipored]# ssh root@direccion_ip_firewall
[root@firewall]# apt-get install iptraf
[root@firewall]# iptraf
```

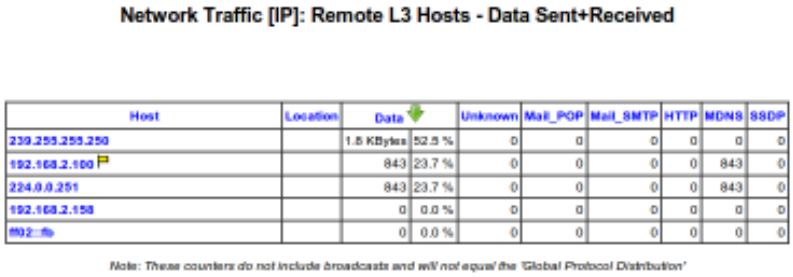


(3ª opción) Instalar **ntop**. Nos permitirá monitorizar el tráfico de nuestra red vía Web a través de un cliente Web (*p.e. Mozilla Firefox*). Al igual que con **iptraf** podremos visualizar las conexiones establecidas por nuestros equipos clientes, el tráfico de red que generan, y por tanto, analizar los servicios que demandan: **ntop**.

```
[root@firewall]# apt-get install ntop
```

Durante la instalación de **ntop** un asistente nos preguntará por las interfaces de red del equipo por las que deseamos que analice el tráfico TCP/IP, y la contraseña para su administración:

Tras la instalación podremos acceder a la interfaz Web de **ntop** y comprobar el tráfico que este analiza. Para ello tan sólo tendremos que colocar en la barra de direcciones de un navegador Web la URL ***http://dirección_ip_firewall:3000*** *(la monitorización vía Web la ofrece a través del puerto tcp/3000)*:



Network Traffic [IP]: Remote L3 Hosts - Data Sent+Received

Host	Location	Data		Unknown	Mail_POP	Mail_SMTP	HTTP	MDNS	SSDP
239.255.255.250		1.8 KBytes	52.5 %	0	0	0	0	0	0
192.168.2.100		843	23.7 %	0	0	0	0	843	0
224.0.0.251		843	23.7 %	0	0	0	0	843	0
192.168.2.158		0	0.0 %	0	0	0	0	0	0
[illegible]		0	0.0 %	0	0	0	0	0	0

Note: These counters do not include broadcasts and will not equal the 'Global Protocol Distribution'

¡¡Aclaración!! Para acceder desde un equipo de la red a la monitorización vía Web ofrecida por **ntop**, deberemos comprobar que el firewall acepta dicha solicitud de conexión al puerto **tcp/3000**. En caso de tener aplicada como política por defecto de entrada (*INPUT*) DROP, deberemos ejecutar el siguiente comando:

[root@firewall]# **iptables -A INPUT -p tcp --dport 3000 -j ACCEPT**

Práctica Nº3.- Proxy HTTP Caché. Squid

En esta práctica configuraremos nuestro equipo GNU/Linux (*Ubuntu*, *Debian*, *Raspberry*, *etc.*) como servidor Proxy HTTP Caché. Para ello se hará uso del afamado software de servicio Proxy Squid.

¡¡Aclaración!! Se denomina Proxy a todo equipo de ofrece un servicio de intermediario entre un equipo cliente y un equipo servidor. Es decir, el cliente hace una solicitud al equipo Proxy, y este se encarga de suplantarle y hacerla llegar al servidor. Este tipo de entorno suele ser muy habitual, ya que nos permite filtrar las conexiones del cliente, forzar una autenticación, o directamente garantizar una determinada calidad de servicio (*QoS*).

Al igual que cuando se configura dentro de una red un servidor DNS caché, con la configuración de un Proxy HTTP se pretende optimizar al máximo el ancho de banda contratado, reduciendo en lo máximo posible la salida a Internet de los equipos cuando ya no es necesario.

Por ejemplo, en el caso del servidor DNS Caché, nuestro servidor almacenaba en memoria las resoluciones DNS directas o inversas (*Nombre DNS FQDN con su correspondiente dirección IP*) que son solicitadas por los clientes de la Intranet durante el "*Time To Live*" (TTL) indicado en la definición de la correspondiente zona maestra del dominio solicitado, evitando de esta forma tener que salir a Internet a buscar repetidas veces la correspondiente resolución del nombre a IP cada vez que un cliente lo requiere, cuando previamente ya ha sido solicitado, optimizando de esta manera el ancho de banda consumido.

Al igual que el servidor DNS Caché, el Proxy HTTP Caché almacenará en memoria las distintas páginas Web solicitadas por los clientes, evitando de esta forma tener que salir a Internet a buscar los documentos Web en el caso de que los tenga cacheados debido a que esa solicitud fue previamente solicitada y cacheada, optimizando de esta manera igualmente el ancho de banda. Por tanto, dos ventajas nos ofrece un Proxy a la hora de configurarlo dentro de nuestra red de área local:

a) Acceso a los contenidos de una manera más rápida. Contenidos que hayan sido previamente cacheados no será necesario salir de nuevo a Internet a buscarlos, entregándoselos al cliente de manera directa y rápida.
b) Reducción del consumo de ancho de banda. Al evitar tener que salir a Internet, en el caso de que la información solicitada este cacheada, nos ahorramos el correspondiente consumo de ancho de banda, permitiendo que otras aplicaciones o servicios puedan hacer uso de él.

¡¡Aclaración!! Cuando decimos que **squid** cachea la información que solicitamos a través de Internet, nos referimos únicamente a objetos con carácter no dinámico, ya que si cacheara contenido dinámico la información recibida por el cliente sería obsoleta. Por tanto, **páginas Web dinámicas escritas en PHP, PyThon, ASP, etc. no serán cacheadas** por defecto por **squid**. Por el contrario, contenidos **HTML** estáticos, imágenes **JPG**/**PNG**/..., documentos en formato **PDF**, etc. si serán cacheados, agilizando nuestra conexión a Internet y ahorrándonos el correspondiente ancho de banda. No obstante, desde el lado del servidor, el desarrollador ha podido definir los objetos como cacheable o no, provocando que **squid** lo cachee o no. El desarrollador también puede definir un Time To Live (TTL) para sus objetos, que será el tiempo que permanecerán en la memoria caché de

nuestro Proxy.

De igual forma, también es muy importante señalar que **el Proxy no cacheará nada que sea referente al protocolo HTTPS tcp/443** al corresponderse con una comunicación cifrada, limitándose a actuar como un simple intermediario entre el cliente y el servidor final, redireccionando directamente el tráfico hacia el puerto tcp/443 a la salida del servidor Proxy, ofreciéndonos únicamente la posibilidad de filtrar las URLs indicadas vía HTTPS.

Además, el servidor Proxy nos va a permitir hacer uso de las siguientes funcionalidades:

1. Controlar tanto desde que equipos clientes se permite el acceso a Internet, como a que equipos servidores Web HTTP esta permitida la conexión, como si se tratara de un firewall.
2. Controlar la franja horaria en la que esta permitida el acceso al servicio web.
3. Obligar a los usuarios que hacen uso del servicio a una autenticación previa (*Proxy no transparente*).
4. Filtrar URLs y contenidos evitando que los usuarios puedan conectarse a determinados sitios Web.
5. Controlar el ancho de banda de salida a Internet consumido por los equipos clientes, garantizando una determinada calidad de servicio, QoS.
6. Informar mediante ficheros de auditoría tanto de los accesos realizados desde los clientes, como de los contenidos solicitados.

3.1.- Directivas Básicas de Configuración del Proxy Squid

Para todo ello, en la presente práctica se va a instalar el software Squid, sobre el cual habrá que configurar su comportamiento. Para ello será necesario definir listas de control de acceso (*ACLs*) mediante la directiva "**acl**", y posteriormente tomar la decisión de permitir o denegar mediante la directiva "**http_access**". De manera resumida, destacaríamos los siguientes tipos de ACL:

Sintaxis directiva "**acl**": **acl** *nombre-acl tipo-acl descripción|fichero-descripciones* Sintaxis directiva "**http_access**": **http_access** *allow|deny nombre-acl*		
Tipo ACL	**Significado**	**Ejemplo de ACL**
src	*Source*. Permite hacer referencia a un conjunto de equipos clientes. Si al definir la ACL queremos hacer referencia a cualquier equipo cliente, podemos hacer uso del valor "**all**".	**acl intranet src 192.168.1.0/24** **http_access allow intranet** **acl maquina1 src 192.168.1.200** **acl cualquiera src all** **http_access deny maquina1** **http_access allow cualquiera**
srcdomain	*Source Domain*. Hace alusión a un equipo cliente basándonos en su nombre de dominio. Para la comprobación se requiere resolución inversa	**acl eq1 srcdomain equipo1.intranet.es** **http_access deny eq1**
dst	*Destination*. Hace alusión a un equipo destino	**acl server dst 192.168.1.200** **http_access deny server**

dstdomain	*Destination Domain*	**acl google dst www.google.com**
proxy_auth	Nos permite establecer como requisito para poder navegar desde un cliente la autenticación previa: login y password. Con la opción "-i" no distingue entre minúsculas y mayúsculas	**acl autenticar proxy_auth REQUIRED** **acl todas src all** **http_access allow todos autenticar** **acl todas src all** **acl grupo1 proxy_auth -i usu1 usu2** **acl grupo2 proxy_auth -i usu3** **http_access deny grupo2 todas** **http_access allow grupo1 todas**
time	Permite especificar una franja horaria. La sintaxis es: **acl** *nombre-acl [días] [h1:m1-h2-m2]* Los días se especifican con la abreviatura del día en inglés: **M** (monday), **T** (tuesday), **W** (wednesday), **H** (thursday), **F** (friday), **A** (saturday) y **S** (sunday).	**acl todas src all** **acl horario1 time M T W H F 9:00-18:00** **acl finde time A S 9:00-13:00** **http_access allow horario1 finde todas**
url_regex	Permite **filtrar URL por palabras clave** o mediante el uso de expresiones regulares. Lo más sencillo es simplemente poner una lista de palabras clave que no deseamos permitir que aparezcan en las URLs de los clientes. En el caso de que la lista de palabras clave sea muy extensa, puede hacerse uso de un archivo de texto cuyo contenido se corresponderá con dicha lista. **Este archivo se indicará mediante la ruta donde se ubica encerrándola entre comillas dobles**. En el caso de querer hacer uso de expresiones regulares, convendría repasar previamente su sintaxis (http://es.wikipedia.org/wiki/Expresi%C3%B3n_regular). Por ejemplo, el carácter "^" es para hacer referencia a URLs que empiezan por alguna cadena de caracteres en concreto, y "$" para indicar terminaciones de la URL. Si no se indica ningún carácter especial, la cadena de caracteres indicada se usará como patrón de búsqueda dentro de las URLs.	**acl maquinas src all** **acl palabras1 url_regex sex porn** **acl palabras2 url_regex "ruta_archivo"** **http_access deny palabras1 palabras2** **http_access allow all** **acl webseguras url_regex -i ^https://** **acl multimedia url_regex -i mp3$ mp4$** **acl videos url_regex youtube** **http_access allow webseguras** **http_access deny multimedia videos** **acl a src all** **acl p1 url_regex sex porn** **acl p2 url_regex "ruta_archivo"** **acl w url_regex -i ^https://** **acl m url_regex -i mp3$ mp4$** **acl v url_regex youtube** **http_access allow a !p1 !p2 !w1 !w !m !v** **acl todas src all** **acl filtro1 url_regex "ruta_archivo"** **acl horario1 time M T W H F 8:00-20:00** **http_access allow todas horario1 !filtro1**

	Para este tipo de filtrado puede resultar sumamente útil el uso de la admiración "**!**" en el control de acceso con **http_access**, para indicar lo contrario a lo que hace la referencia la ACL.	

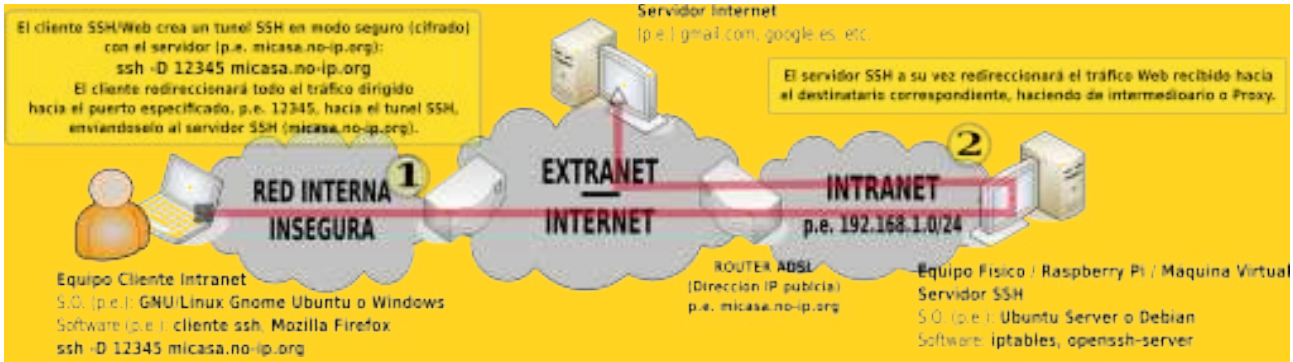
¡¡Observación!! El Proxy Squid que vamos a configurar a lo largo de los ejercicios prácticos que se proponen a continuación nos van a permitir gestionar las conexiones Web que se realizar desde los equipos y usuarios que forman parte de nuestra Intranet privada. A este tipo de Proxy se le denomina Proxy Cache HTTP o directamente Proxy Web o Proxy "a secas", y hay que diferenciarlo de otros tipos de Proxy como es el Proxy SOCKS (*SOCKetS*), el cual esta pensado no sólo para gestionar el tráfico Web, sino para hacer de intermediario entre equipos clientes y servidores permitiéndonos gestionar cualquier tipo de conexión que sea soportada por el protocolo SOCKS (*RFC 3089*). Es decir, el equipo cliente tiene que ser un cliente SOCKS que implemente este tipo de protocolo específico.

Una forma sencilla de comprender como funciona un Proxy SOCKS es utilizar **ssh** con la opción "**-D número_puerto IP_servidor_SSH**", la cual nos permite hacer una redirección de puertos dinámica, convirtiendo nuestro equipo y el túnel SSH creado con el servidor, en un Proxy SOCKS redireccionando todo el tráfico de la aplicación que configuremos por el canal seguro hacía el servidor SSH que hayamos especificado, redirigiendo éste a su vez el tráfico hacia el equipo que indiquemos.

Un caso típico de uso de un Proxy SOCKS podría ser el siguiente: nos encontramos navegando por Internet en una red insegura, o con dudas relativas a que nuestras comunicaciones puedan ser husmeadas por algún "sniffer", y queremos evitarlo creando un túnel SSH cifrado seguro por el cual viajará todo el tráfico generado por nuestro navegador Web. A través de esta configuración todo el tráfico que generemos será redireccionado hacía el servidor SSH que especifiquemos y este hará de intermediario (*Proxy*) entre yo y los servidores a los que me conecte. Por ejemplo, supongamos que tenemos asociado un nombre de dominio público gratuito a la dirección IP pública del router de nuestra casa (*p.e. micasa.no-ip.org*), y que éste tiene redireccionado el puerto 22 hacía un servidor SSH que tenemos internamente (*p.e. una Raspberry Pi*). Al establecer el tunel SSH con el servidor haciendo uso de la opción **-D**, hacemos que el servidor de nuestra casa haga de Proxy entre nosotros y los servidores a los que nos conectemos durante la navegación Web.

A modo de ejemplo, a continuación se muestra como configurar un equipo cliente para hacer uso de un Proxy SOCKS local implementado mediante SSH mientras esta navegando en una red supuestamente insegura, de tal forma que todo el tráfico generado por el cliente (*p.e. Mozilla Firefox integra un cliente SOCKS que soporta el protocolo SOCKS*) y dirigido hacía el puerto que especifiquemos (*p.e. 12345*), será encapsulado y redireccionado hacía el servidor SSH, reenviándolo éste a su vez al servidor Web correspondiente.

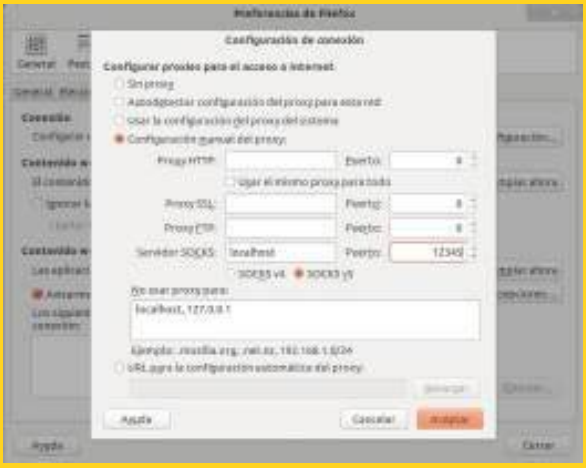
Advertir, que en tal situación, el ancho de banda de la conexión vendrá limitado por la velocidad de upload de la línea ADSL contratada en el lado del servidor SSH.



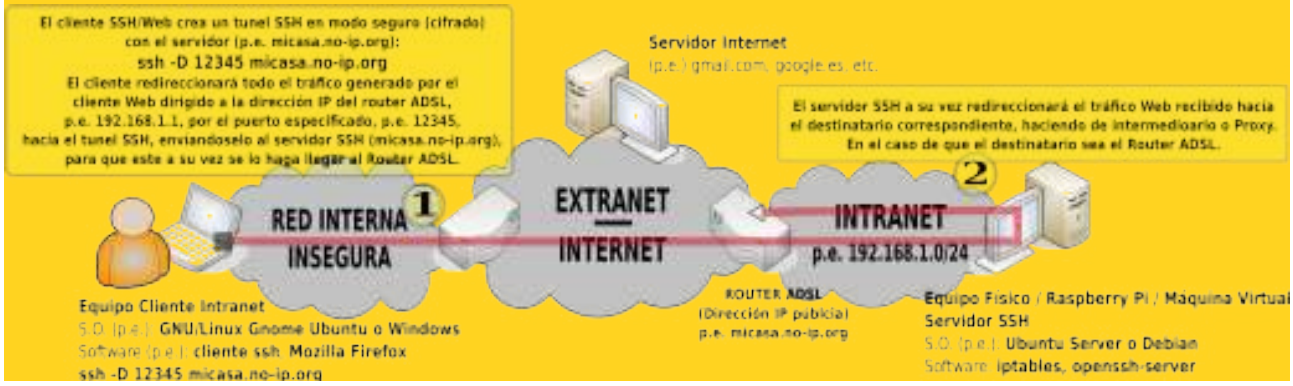
Para ello, tan sólo necesitaremos crear el tunel SSH (*en Windows haríamos uso de putty*), y posteriormente configurar las opciones de nuestro navegador:

```
[usuario-cliente@linux]$ ssh -D 12345 micasa.no-ip.org
```

En el caso de que hagamos uso de Mozilla Firefox (*Editar → Preferencias → Avanzado → Red → Configuración*), aunque podría cualquier otro tipo de navegador:



Otra opción de uso muy interesante de todo lo anterior, es usar el túnel SSH creado a modo de "VPN", ya que a través del Proxy SOCKS podemos alcanzar cualquier equipo de la Intranet donde se localiza el servidor SSH. Por ejemplo, podríamos configurar el router ADSL haciendo uso de la dirección IP privada de la Intranet que tenga asignada (*p.e. 192.168.1.1*):



Ej. Práctico 3.1.1: Definición de ACLs y HTTP_ACCESS en Squid

Suponiendo que disponemos de un **Proxy Squid** No Transparente que da servicio a todos los equipos clientes de una Intranet 192.168.1.0/24 ¿Qué **ACLs** y **http_access** definirías en el fichero de configuración de **Squid** con la finalidad de cumplir las especificaciones que se indican en la siguiente tabla? El cómo se crean las cuentas de usuario, y como se termina de configurar Squid se verá en los próximos ejercicios prácticos. Los nombres de las ACLs puedes inventártelos.

Origen	Destinos Prohibidos	Usuarios Auth. Digest	Palabras Clave Prohibidas	Horario
192.168.1.0	**-**	**adm1 adm2**	**sex, porn**	**8:00 a 14:00**
	www.dropbox.com www.facebook.com www.twitter.com	**empleado1 empleado2**	**comida, recetas, deporte, sport, futbol, baloncesto, porn, sex**	
	-	**adm1 adm2 adm3 empleado1 empleado2 empleado3**	**sex, porn**	**14:00 a 16:00**
192.168.1.1 192.168.1.2	**-**	**adm3**	**-**	**16:00 a 22:00**
	www.dropbox.com www.facebook.com www.twitter.com www.juegos.com www.amigos.com	**empleado3**	**comida, recetas, deporte, sport, futbol, baloncesto, porn, sex**	

Solución Ej. Pr. 3.1.1.I.- Cómo Denifinir ACLs y HTTP_ACCESS en Squid

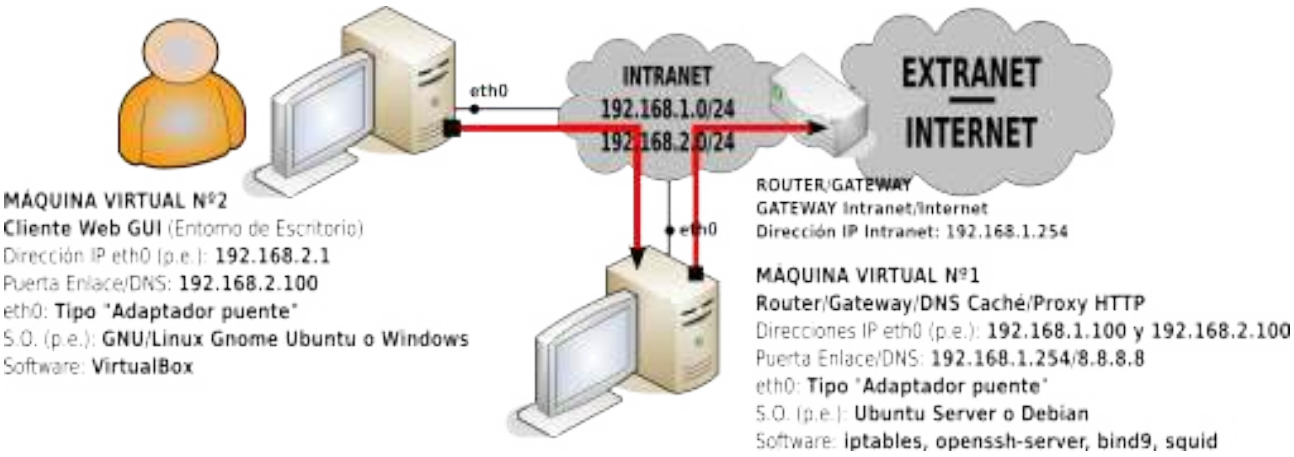
Siguiendo la sintaxis explicada anteriormente, las ACLs y reglas de acceso que definiríamos en el archivo squid.conf para cumplir los requisitos del enunciado serían las siguientes:

```
[root@proxy]# nano /etc/squid3/squid.conf
# Definimos las ACLs que identifican a los equipos
acl red1 src 192.168.1.0/24
acl equipos1 src 192.168.1.1 192.168.1.2
# Definimos las Nombres de Dominio asociados a los destinos a filtrar
acl websprohibidas1 dstdomain www.dropbox.com www.facebook.com www.twitter.com
acl websprohibidas2 dstdomain www.dropbox.com www.facebook.com www.twitter.com
www.juegos.com www.amigos.com
# Definimos las cuentas de usuario Squid
```

```
acl grupo1 proxy_auth -i adm1 adm2
acl grupo2 proxy_auth -i empleado1 empleado2
acl grupo3 proxy_auth -i adm1 adm2 adm3 empleado1 empleado2 empleado3
acl grupo4 proxy_auth -i adm3
acl grupo5 proxy_auth -i empleado3
# Definimos las Palabras Clave no permitidas en las URLs
acl palabrasprohibidas1 url_regex sex porn
acl palabrasprohibidas2 url_regex comida recetas deporte sport futbol baloncesto porn sex
# Definimos las Franjas Horarias donde los usuarios tendrán acceso
acl horario1 time M T W H F 8:00-14:00
acl horario2 time M T W H F 14:00-16:00
acl horario3 time M T W H F 16:00-22:00
# Definimos una ACL global que afecte a todos los equipos
acl todas src all
# Definimos las Reglas de Acceso
http_access allow red1 grupo1 !palabrasprohibidas1 horario1
http_access allow red1 grupo2 horario1 !websprohibidas1 !palabrasprohibidas2
http_access allow red1 grupo3 horario2 !palabrasprohibidas1
http_access allow equipos1 grupo4 horario2
http_access allow equipos1 grupo5 horario3 !websprohibidas2 !palabrasprohibidas2
http_access deny todas
```

3.2.- Configuración previa del Entorno de Red: Gateway/Proxy Squid

Para las prácticas que realizaremos en este capitulo, al igual que en los capítulos anteriores, haremos uso de virtualización mediante VirtualBox. En concreto, si se observa la siguiente figura, configuraremos un entorno de red, donde el equipo servidor (*máquina virtual Nº1*) hará las veces de puerta de enlace o gateway de las Intranet cableada e inalámbrica hacía la Internet, hace de servidor DNS caché para los equipos de las Intranet, y hace de Proxy HTTP.



Además, con la finalidad de hacer una división lógica (*que no física*) entre la Intranet cableada, y el segmento de red correspondiente al router ISP, se asignará a la máquina virtual nº1 (*o un microcomputador Raspberry, por ejemplo*) una interfaz de red virtual con una dirección de red

del rango 192.168.2.0/24. Para todo ello, a excepción de la configuración como Proxy HTTP que veremos más tarde, seguiremos los siguientes pasos:

a) Tal como se hizo en los capítulos anteriores, haciendo uso de un script de configuración configuraremos todo lo relativo al direccionamiento ip. Además, para que haga de gateway, activaremos el "**ip_forward**" para que puedan reenviarse paquetes TCP/IP, y habilitaremos la NAT POSTROUTING para que todos los paquetes que se dirigen hacia el exterior lo hagan con la dirección de origen cambiada por la de su interfaz eth0 192.168.1.100:

```
[root@mv1]# nano /etc/init.d/conf-red.sh
#!/bin/bash
# Paramos el servicio networking para permitir la configuración mediante nuestro script:
/etc/init.d/networking stop
# Configuramos las direcciones IP, puerta de enlace y servidor DNS preferido
ifconfig eth0 192.168.1.100
ifconfig eth0:alias1 192.168.2.100
route add default gw 192.168.1.254
echo "nameserver 8.8.8.8" > /etc/resolv.conf
# Activamos el forwarding y el MASQUERADE para que actúe como un gateway
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -t nat -F
iptables -t nat -A POSTROUTING -j MASQUERADE
```

¡¡Aclaración!! Para añadir más de una dirección IP a una misma interfaz de red se hace uso de alias o interfaces virtuales. La sintaxis para su asignación es simplemente indicar el nombre de la interfaz, seguido de dos puntos ":" y el nombre del alias, **"eth0:*nombre_alias*"**.

b) Para que haga de servidor DNS caché tan sólo necesitaremos instalar el software bind9. A partir de ese momento, sin necesidad de ninguna configuración a posteori, y gracias a la información aportada por el archivo "**/etc/bind/db.root**", haciendo uso de las direcciones IP públicas de los servidores DNS raíz TNS (*Top Name Severs*), nuestro equipo servidor será capaz de resolver cualquier nombre de dominio FQDN público, y cachearlo el tiempo TTL (*Time To Live*) indicado en su correspondiente zona maestra.

```
[root@mv1]# apt-get install bind9
```

Ej. Práctico 3.2.1: Proxy No Trasparente con Autenticación Básica

Siguiendo con el esquema de red mostrado en la anterior figura, formado por un equipo cliente (*máquina física u otra máquina virtual*) y un equipo servidor (*gateway, DNS y Proxy HTTP, que puede ser una máquina virtual o por ejemplo, una Raspberry Pi*), a continuación se implementará una primera configuración de nuestro Proxy squid como P**roxy no transparente**.

Esto significa que el usuario final o cliente Web debe saber que para poder navegar a través de Internet necesita atravesar un equipo en la red que hace de intermediario entre él y el servidor al que trata de acceder llamado Proxy, y que este controla en todo momento que es posible hacer. En concreto, el cliente requerirá configurar su navegador para informarle de la dirección IP del Proxy y de su puerto de servicio a su navegador.

A modo de ejemplo, el Proxy caché HTTP a implementar deberá cumplir las siguientes características:

Puertos de Servicio **Cantidad Memoria RAM**	**Directorio Caché** **Tamaño Caché** **Nº Directorios y Subdirectorios** **Tamaño máximo de archivo a cachear**	**URLs no permitidas** **Auditoria Caché** **Auditoria Objetos Cacheados**	**Autenticación** **Usuarios permitidos** **Equipos permitidos y no permitidos** **Auditoria acceso**
3128 y 8088 32 MB	/var/spool/squid3 1GB 16/256 128MB	www.marca.com www.sport.es www.as.com *youtube* *porn* *sex* cache_log /var/log/squid3/cache.log cache_store_log /var/log/squid3/store.log	Sí usuproxy1, usuproxy2 y usuproxy3 192.168.2.0/24 a excepción del 192.168.2.101 (*p.e.*) access_log /var/log/squid3/access.log

¡¡Importante!! Antes de llevar a cabo la autenticación de usuarios en el servicio Proxy HTTP, deberíamos tener claro dos aspectos:

1) Donde almacenar las cuentas de usuario del servicio y sus contraseñas (*un fichero plano, una base de datos, usuarios del sistema, LDAP, etc.*).
2) Decidir la forma en que queremos que viajen las claves o contraseñas introducidas por los usuarios desde sus equipos clientes hacia el servidor proxy: sin cifrar (*basic*) o cifradas (*digest*).

En relación a los problemas prácticos de configuración del Proxy **squid** que se han planteado y resuelto en el presente capítulo, por cuestiones de sencillez, y evitar de esta forma el tener que introducir un nuevo servicio (*p.e. mysql para la autenticación mediante el uso de una base de datos de usuarios*), se hará uso de un fichero plano para guardar los usuarios y sus contraseñas (*las contraseñas quedarán almacenadas de manera cifrada*), y se mostrará de que manera configurar **squid** para que las contraseñas viajen del cliente al servidor, tanto sin cifrar, como de manera cifrada.

En concreto, para la configuración de los aspectos de autenticación de **squid** haremos uso de la directiva "**auth_param**", la cual adoptará el siguiente aspecto en la resolución de los casos prácticos aquí planteados:

Directiva	Tipo	Subprograma de Squid encargado de realizar la Autenticación de Usuarios		Ruta del **Fichero Plano** donde se almacenan Usuarios y Contraseñas
auth_param	**basic**	**program**	/usr/lib/squid3/**ncsa_auth**	/etc/squid3/claves.dat
	digest		/usr/lib/squid3/**digest_pw_auth**	-c /etc/squid3/claves-digest.dat

- **ncsa_auth**: programa o librería de squid que nos permite corroborar si el login y password introducidos por un cliente son válidos, ante una autenticación de tipo "**basic**". Para generar el fichero plano que almacenará la lista de usuarios y contraseñas válidos utilizaremos el comando "**htpasswd**" disponible tras la instalación del software servidor HTTP apache2 (*apt-get install apache2*).
- **digest_pw_auth**: programa o librería de squid que nos permite corroborar si el login y password introducidos por un cliente son válidos, ante una autenticación de tipo "**digest**". Para generar el fichero plano que almacenará la lista de usuarios y contraseñas válidos utilizaremos el comando "**htdigest**" disponible tras la instalación del software servidor HTTP apache2 (*apt-get install apache2*), o mediante "**md5sum**", tal como se mostrará posteriormente.

Otro aspecto a tener en cuenta en toda zona de autenticación, es la definición identificador de la zona, o **realm**, que le diferenciará de cualquier otra zona de autenticación que pueda estar implementada en el entorno de red donde nos encontremos. Este identificativo es sumamente importante sobre todo en autenticación de tipo "**digest**", ya que este identificador es usado en la generación de cada una de las contraseñas cifradas asociadas a cada uno de los usuarios que se irán almacenando en el fichero plano encargado de almacenar la lista de usuarios. Esto significa, que cada zona de autenticación de tipo "**digest**" tendrá sus propios usuarios, no pudiendo compartirse las cuentas de usuario y contraseñas por varias zonas de autenticación (*aspecto que si que se permite en autenticación de tipo "basic"*).

Solución Ej. Pr. 3.2.1.I.- Configuración de un Proxy No Transparente Auth. Basic

Antes de nada, instalaremos el software "**squid**" el cuál convertirá nuestro equipo servidor en un Proxy HTTP:

```
[root@mv1]# apt-get install squid3
```

Después pasaremos a su configuración editando el archivo de configuración del servicio "**squid.conf**" (*podemos editarlo de cero*):

¡¡Aclaración!! Para la realización del presente ejercicio práctico partiremos de un fichero de configuración vacío. Para no eliminar el fichero "**squid.conf**" que viene por defecto, lo renombraremos. En el caso de querer estudiar cual es comportamiento por defecto del Proxy squid tras su instalación, puede ejecutarse el siguiente comando (*filtramos el contenido del archivo original evitando que nos muestre todas aquellas líneas que empiezan por "#", ya que se encuentra altamente comentado*):

[root@mv1]# **grep -v '^#' squid.conf | grep . | more**

[root@mv1]# **cd /etc/squid3**
[root@mv1:/etc/squid3]# **mv squid.conf squid-anterior.conf**
[root@mv1:/etc/squid3]# **nano squid.conf**

 # Directivas globales
 # Nombre del servidor proxy visible en los mensajes informativos enviados al cliente Web:
visible_hostname miproxy
 # Puertos de servicio del proxy. Podemos hacer que escuche por más de un puerto simultáneamente:
http_port 3128
http_port 8088
 # Datos de la Cache:
Cantidad de memoria RAM que reservamos para este servicio proxy HTTP (p.e.): 32 MB
cache_mem 32 MB *# Cuidado, debe haber un espacio entre el "32" y el "MB"!!!*
Ruta absoluta del directorio de nuestro sistema de almacenamiento donde se swapearán los sitios Web visitados, su tamaño 1024MB, y el número de directorios (16) y subdirectorios (256)
cache_dir ufs /var/spool/squid3 1024 16 256
Máximo tamaño del archivo que una vez descargado será cacheado (p.e.): 128 MB
maximum_object_size_in_memory 128 MB
 # Archivos de auditoría en relación con los accesos, contenidos visitados y almacenados:
access_log /var/log/squid3/access.log
cache_log /var/log/squid3/cache.log
cache_store_log /var/log/squid3/store.log
 # Autenticación de usuarios:
Método de autenticación: ncsa_auth, y archivo de usuarios y contraseñas: claves.dat
auth_param basic program /usr/lib/squid3/ncsa_auth /etc/squid3/claves.dat
Cantidad de procesos hijos de squid involucrados en la autenticación
auth_param basic children 5
Nombre realm identificativo de la zona de autenticación. Si contiene espacios en blanco, hay que indicarlo entre comillas simples o dobles para acotarlo
auth_param basic realm "Servidor HTTP Proxy – Modulo Seguridad"
Periodo de validez de la autenticación. Trascurrido ese tiempo se requerirá de nuevo
auth_param basic credentialsttl 2 hours
 # Definición de Listas de Control de Acceso, ACLs
Definición de ACL para asegurar la autenticación declarada anteriormente
acl autenticacion proxy_auth REQUIRED

```
## Definición de ACLs para controlar el acceso a máquinas
acl cualquier_maquina src all
acl maquina1 src 192.168.2.101
## Definición de ACLs para filtrar URLs de determinados sitios Web por nombre de dominio:
acl websprohibidas1 dstdomain "/etc/squid3/websprohibidas1.dat"
## Defínción de ACLs para filtrar URLs Web según expresiones regulares y palabras clave:
acl websprohibidas2 url_regex "/etc/squid3/websprohibidas2.dat"
# Filtrado en base a las ACLs definidas anteriormente:
http_access deny maquina1
http_access allow cualquier_maquina  autenticacion !websprohibidas1 !websprohibidas2
```

Una vez configurado el servicio Proxy, antes de reiniciar el servicio para que surtan efecto los cambios, editaremos los ficheros indicados en la configuración anterior: "**claves.dat**", "**websprohibidas1.dat**" y "**websprohibidas2.dat**".

Respecto al fichero de autenticación de usuarios "**claves.dat**", haremos uso de la herramienta software "**htpasswd**", tal como utilizábamos para la autenticación de zonas no anónimas en Apache. A modo de ejemplo, crearemos a continuación un fichero que contendrá tres usuarios válidos (*login/password*) llamado "**usuproxy1**", "**usuproxy2**" y "**usuproxy3**" (*la opción "-**c**" crea el fichero de usuarios, y por tanto, sólo se incluye al crear el fichero para el primer usuario*):

```
[root@mv1]# htpasswd -c /etc/squid3/claves.dat usuproxy1
[root@mv1]# htpasswd /etc/squid3/claves.dat usuproxy2
[root@mv1]# htpasswd /etc/squid3/claves.dat usuproxy3
```

¡¡Aclaración!! El comando "**htpasswd**" que nos permite crear las cuentas de usuario sólo estará disponible si esta instalado previamente "**apache2**", ya que es una de sus utilidades. Por tanto, en caso de no disponer de "**apache2**", habrá que instalarlo: "apt-get install apache2". Estas cuentas de usuario que permitirán autenticarse al Proxy HTTP Caché no son cuentas de usuario del sistema, sino cuentas de usuario pensadas para ser usadas por otro software como "**squid**" o "**apache2**".

En relación a los ficheros donde le indicaremos las URLs no permitidas y contenidos que serán filtrados:

```
[root@mv1]# nano /etc/squid3/websprohibidas1.dat
# Para evitar acceder a determinadas URLs
www.marca.com
www.sport.es
www.as.com
```

¡¡Aclaración!! Es importante resaltar que el filtrado de las Web a través de sus URLs también lo podríamos haber obtenido mediante el firewall "**iptables**". Por ejemplo:

```
[root@raspberry]# iptables -t filter -A FORWARD -s 192.168.2.0/24 -d www.sport.es \
                  -j DROP
```

```
[root@mv1]# nano /etc/squid3/websprohibidas2.dat
# Se denegará el acceso a aquellas URLs que contengan una de las siguientes cadenas de caracteres, patrones de búsqueda o palabras clave:
youtube
porn
sex
...
```

Después de todo lo anterior tan sólo nos quedará reiniciar el servicio para que surtan efecto los cambios en la configuración, aunque antes comprobaremos que no hayamos cometido algún error sintáctico o semántico (*"-k check" o "-k parse" nos proporcionan información equivalente*):

```
[root@mv1]# squid3 -k check
[root@mv1]# squid3 -k parse
[root@mv1]# /etc/init.d/squid3 restart
```

¡¡Importante!! Antes de comprobar el correcto funcionamiento del Proxy desde el equipo cliente, comprobarás que el cliente navega sin necesidad de Proxy, ya que el equipo servidor hace de gateway y reenvía por defecto las solicitudes de conexión dirigidas a los servicios HTTP tcp/80 o HTTPS tcp/443 hacia el servidor de Internet que corresponda. Para forzar al navegador del cliente a tener que configurar el Proxy para poder navegar, deberíamos añadir reglas de filtrado en el servidor/firewall para que corte el tráfico TCP/IP que vaya dirigido a los servicios HTTP tcp/80 o HTTPS tcp/443, evitando que puedan navegar sin configuración de Proxy:

```
[root@mv1]# iptables -A FORWARD -p tcp --dport 80 -s 192.168.2.0/24 -j DROP
[root@mv1]# iptables -A FORWARD -p tcp --dport 443 -s 192.168.2.0/24 -j DROP
```

De igual forma, si tenemos en cuenta que todos los navegadores Web actuales también son clientes FTP, podemos forzar a utilizar este tipo de cliente, y por tanto, a configurar el Proxy en el navegador para este tipo de conexiones cerrando igualmente su puerto por defecto **tcp/udp/21**:

```
[root@mv1]# iptables -A FORWARD -p tcp --dport 21 -s 192.168.2.0/24 -j DROP
[root@mv1]# iptables -A FORWARD -p udp --dport 21 -s 192.168.2.0/24 -j DROP
```

Una vez configurado nuestra máquina virtual como servidor Proxy, al tratarse de un Proxy no transparente deberemos configurar a los equipos clientes para advertirles de ello. En concreto, deberemos configurar el navegador predeterminado de cada uno de los clientes indicando la dirección IP y puerto de escucha del Proxy.

¡¡Observación!! Normalmente, cuando se habla de configurar el Proxy HTTP en un equipo cliente, se suele asociar al cliente con un equipo que tenga interfaz gráfica, ya sea GNU/Linux (*entorno GNOME, KDE, etc.*) o Windows. No obstante, no debemos olvidarnos de que hay equipos sin interfaz gráfica, como puede ser un equipo servidor con Ubuntu Server, que aunque no disponen de una navegador Web donde configurar el Proxy, este sí es necesario para poder instalar y actualizar software de los repositorios, al ser muchos de ellos repositorios Web HTTP.

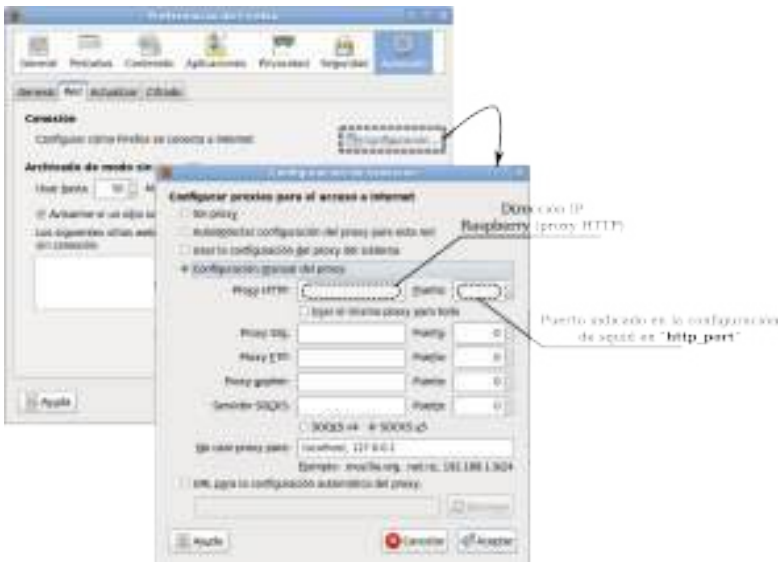
A modo de ejemplo, si quisiéramos configurar el Proxy en un Ubuntu Server tan sólo tendríamos que definir las variables del sistema **http_proxy** y **https_proxy**:

[root@mv1]# **export http_proxy=http://***dirección_ip_proxy:puerto_servicio/*
[root@mv1]# **export https_proxy=https://***dirección_ip_proxy:puerto_servicio/*

En el caso de que queramos que la configuración persista ante reinicios del sistema, deberemos definir dichas variables en algún fichero de configuración del sistema (*p.e. /etc/environment*):

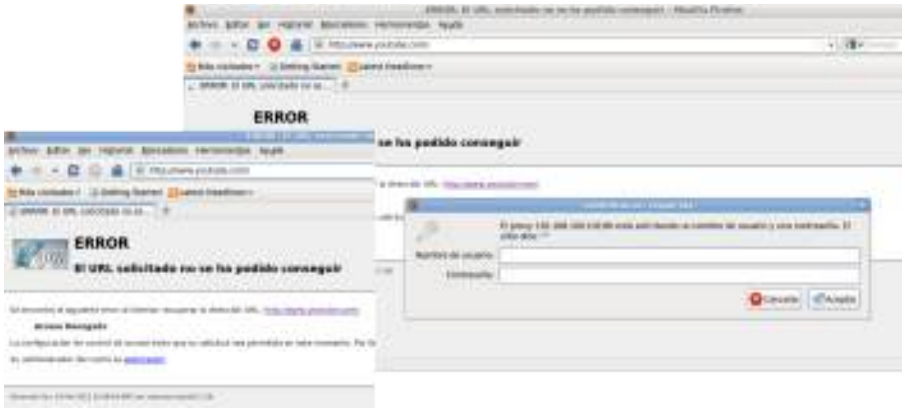
[root@mv1]# **nano /etc/environment**
 # Añadimos las siguientes líneas:
export http_proxy=http://*dirección_ip_proxy:puerto_servicio/*
export https_proxy=https://*dirección_ip_proxy:puerto_servicio/*
...

Por ejemplo, en el caso de estar utilizando Mozilla como navegador deberíamos ir a "**Preferencias**", y en "**Avanzado**", en su pestaña "**Red**" pulsaríamos sobre el botón "**Configuración** …", y allí ya podremos indicar todo lo relativo al Proxy:



¡¡Aclaración!! Cada cliente o navegador Web (incluso cada versión de cada navegador) configura el Proxy de manera diferente. Deberás comprobar en que opción del menú de tu navegador preferido se configura el Proxy para poderlo probar.

Una vez establecida la conexión podremos comprobar tanto la necesidad de autenticación como el filtrado de contenidos:



¡¡Importante!! Para comprobar quien se autentica o los contenidos que va cacheando nuestro Proxy, pueden consultarse los archivos ***.log** de **auditoría** indicados en su configuración:

[root@mv1]# **more /var/log/squid3/store.log**
[root@mv1]# **tail -f /var/log/squid3/store.log** (*nos irá mostrando en tiempo real lo que cachea*)
[root@mv1]# **more /var/log/squid3/access.log**
[root@mv1]# **tail -f /var/log/squid3/access.log**
[root@mv1]# **more /var/log/squid3/cache.log**
[root@mv1]# **tail -f /var/log/squid3/cache.log**

No obstante, al final del presente capítulo, se propondrá un ejercicio práctico para mostrar vía Web el contenido del fichero de auditoria.

Ej. Práctico 3.2.2: Proxy No Transparente con Autenticación Digest

Con la finalidad de completar al ejercicio anterior, en el presente ejercicio se mostrará como configurar squid para ofrecer un servicio Proxy donde la autenticación se hace mediante el método digest, el cual nos garantiza que las contraseñas introducidas por el usuario desde su equipo cliente viajen al servidor de manera cifrada.

***Solución Ej. Pr. 3.2.2.I.**- Configuración de un Proxy No Transparente Auth. Digest*

Para cambiar el tipo de autenticación de **basic** a **digest** en nuestro Proxy será necesario modificar aquellas directivas utilizadas en el ejercicio anterior relativas a "**auth_param basic**". Además la librería encargada de la autenticación ya no será "**ncsa_auth**", sino que ahora utilizaremos "**digest_pw_auth**". El aspecto que deberá tener nuestro nuevo archivo de configuración "**squid.conf**" será ahora el siguiente (*las principales directivas usadas ya se han comentadas en el ejercicio anterior, por lo que no se volverá a hacerlo*):

```
[root@mv1:/etc/squid3]# nano squid.conf
# Directivas globales
visible_hostname miproxy

http_port 3128
http_port 8088
```

```
cache_mem 32 MB
cache_dir ufs /var/spool/squid3 1000 16 256
maximum_object_size_in_memory 128 MB

access_log /var/log/squid3/access.log
cache_log /var/log/squid3/cache.log
cache_store_log /var/log/squid3/store.log

auth_param digest program /usr/lib/squid3/digest_pw_auth -c /etc/squid3/claves-digest.dat
auth_param digest children 5
auth_param digest realm proxy-digest

acl autenticacion proxy_auth REQUIRED
acl cualquier-maquina src all
acl maquina1 src 192.168.2.101
acl websprohibidas1 dstdomain "/etc/squid3/websprohibidas1.dat"
acl websprohibidas2 url_regex "/etc/squid3/websprohibidas2.dat"
http_access deny maquina1
http_access allow cualquier-maquina autenticacion !websprohibidas1 !websprohibidas2
```

Antes de chequear la configuración anterior, y reiniciar el servicio, generaremos el nuevo archivo de autenticación de usuarios, llamado en este ejercicio "**claves-digest.dat**". Para su creación existen dos posibles opciones (*se recomienda por sencillez la primera, htdigest*):

(1ª opción) Mediante el comando "**htdigest**" (*al igual que "htpasswd", disponible a través del software apache2*). Su sintaxis y uso es la siguiente (*la opción "-c" es para crear el fichero de autenticación, y el realm el indicado en la configuración anterior*):

```
[root@mv1:/etc/squid3]# htdigest [-c] [ruta_archivo_de_autenticación] [realm] [nombre_usuario]
```

Por tanto, según lo establecido en los requisitos del enunciado, el fichero de usuarios del Proxy para autenticación digest quedaría de la siguiente manera:

```
[root@mv1:/etc/squid3]# htdigest -c /etc/squid3/claves-digest.dat proxy-digest usuproxy1
[root@mv1:/etc/squid3]# htdigest /etc/squid3/claves-digest.dat proxy-digest usuproxy2
[root@mv1:/etc/squid3]# htdigest /etc/squid3/claves-digest.dat proxy-digest usuproxy3
```

(2ª opción) Mediante el comando "**md5sum**". Este comando nos permite generar una contraseña útil para la autenticación digest. Por ejemplo, si queremos dar de alta un usuario llamado "**usuproxy1**", dentro de la zona de autenticación con realm "**proxy-digest**", y contraseña "1", el siguiente comando nos generaría la contraseña cifrada que habría que introducir el el archivo de contraseñas:

```
[root@mv1:/etc/squid3]# echo -n 'usuproxy1:proxy-digest:1' | md5sum | cut -f1 -d' '
```

Como el resultado habría que dejarlo en el fichero de autenticación de usuarios y contraseñas establecido en "squid.conf", claves-digest.dat siguiendo el formato

"usuario:realm:contraseña_cifrada", podríamos hacer lo siguiente (*es un poco más engorroso, pero es otra opción posible*):

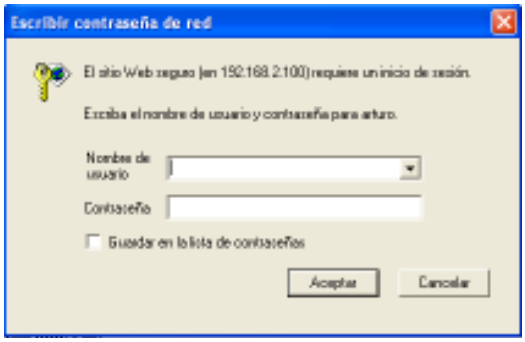
```
[root@mv1:/etc/squid3]# echo "usuproxy1:proxy-digest:`echo -n 'usuproxy1:proxy-digest:1' | md5sum | cut -f1 -d' '`" >> claves-digest.dat
```

¡¡Aclaración!! Por su sencillez se recomienda la utilización del comando **htdigest** para la creación del fichero de autenticación de usuarios **digest**. No obstante, se ha comentado esta segunda opción mediante **md5sum**, para comprender que **htdigest** lo que '*por debajo*' es simplemente hacer uso de **md5sum**. Más tarde, en los ejercicio 3.16 y 3.1.7 se implementará un portal cautivo en PHP bajo apache2 donde la autenticación la realizaremos mediante esta misma técnica de cifrado de claves, **md5**.

Después de todo lo anterior tan sólo nos quedará reiniciar el servicio para que surtan efecto los cambios en la configuración, aunque antes comprobaremos que no hayamos cometido algún error sintáctico o semántico (*"-k check" o "-k parse" nos proporcionan información equivalente*):

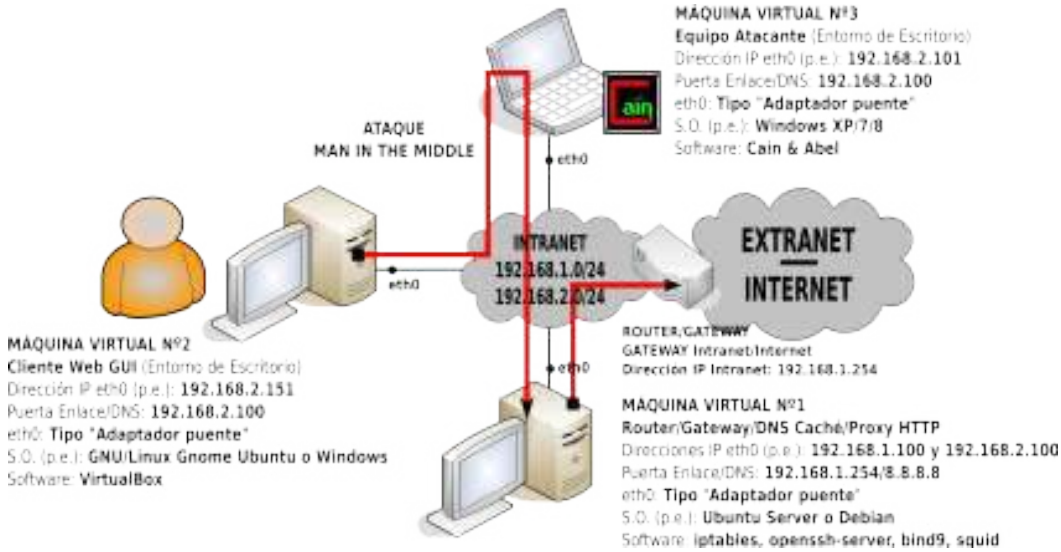
```
[root@mv1]# squid3 -k check
[root@mv1]# squid3 -k parse
[root@mv1]# /etc/init.d/squid3 restart
```

Al comprobar su correcto funcionamiento desde un cliente Web, nos aparecerá la correspondiente ventana de autenticación:



Ej. Práctico 3.2.3: Captura de passwords en modo Basic y Digest

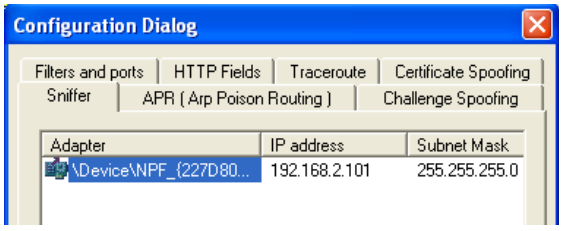
Con la finalidad de mostrar las diferencias entre los dos sistemas de autenticación vistos, **basic** y **digest**, en este ejercicio práctico se propone comprobar mediante la potente herramienta software de sniffing, spoofing e interceptación de passwords "**Cain & Abel**", advertir que en modo **basic** se puede obtener fácilmente **las contraseñas de usuarios del Proxy**, mientras que en modo **digest** no es tan trivial. Para ello, tal como se puede observar en el siguiente esquema de red, introduciremos un nuevo equipo en la red con sistema operativo Windows, ya que **Cain** esta diseñado para funcionar bajo este sistema, que hará de **Man in the Middle**.



Solución Ej. Pr. 3.2.3.I.- Configuración de un Man in The Middle con Cain

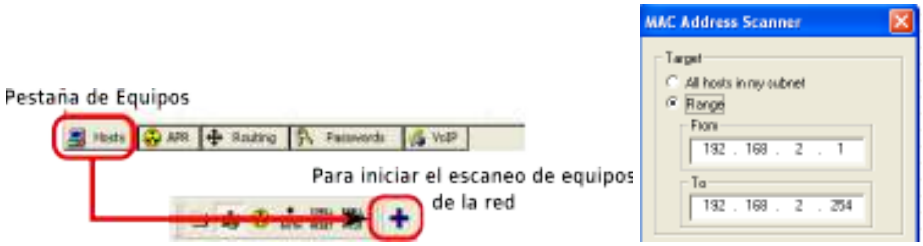
Para comprobar la fortaleza del sistema de autenticación digest frente al modo basic seguiremos los siguientes pasos:

(1) Como ya se ha dicho en el enunciado necesitaremos crear una nueva máquina virtual bajo sistema operativo Windows. Después nos descargaremos de Internet el software "Cain & Abel" y lo instalaremos.
(2) Abriremos Cain y lo configuraremos para que se comporte como un Man in the Middle entre el cliente Web y el servidor Proxy.
(3) Para una correcta configuración de Cain comenzaremos indicando mediante la opción del menú superior "**Configure**" cual de las interfaces de red disponibles en el equipo Windows queremos que utilice Cain para husmear la red (*sniffing*). En nuestro caso al sólo disponer de la interfaz configurada como adaptador puente, no hay muchas dudas.

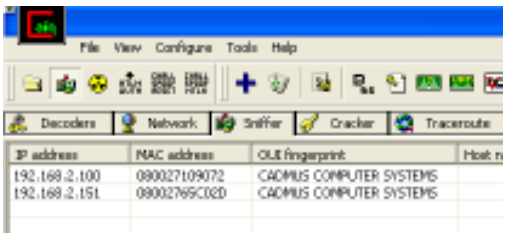


(4) A continuación iniciaremos la función de sniffer de Cain presionando sobre el segundo icono del menú superior "**Start/Stop Sniffer**":

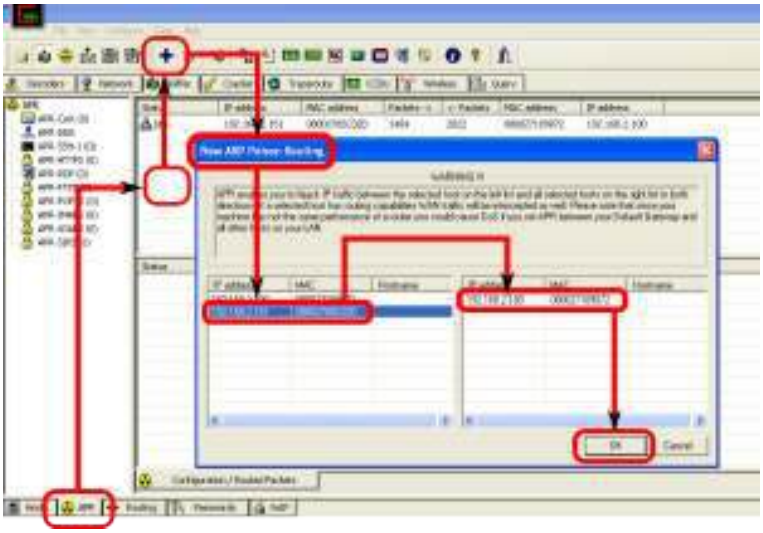
(5) A continuación inspeccionaremos la red para ver que equipos se encuentran activos, y por tanto, que equipos son susceptibles de ser atacados. Para ello nos situaremos sobre el menú inferior, en la pestaña "Hosts", y luego le daremos icono "+" del menú superior, lo que nos permitirá iniciar un escaneo de la red:



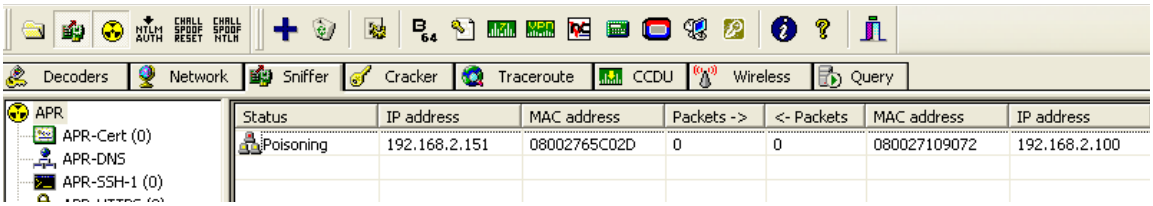
Hecho esto, Cain escaneará la red, detectando que los únicos equipos activos en la red lógica 192.168.2.0/24 son los esperados: el equipo cliente 192.168.2.151, y el equipo servidor Proxy 192.168.2.100:



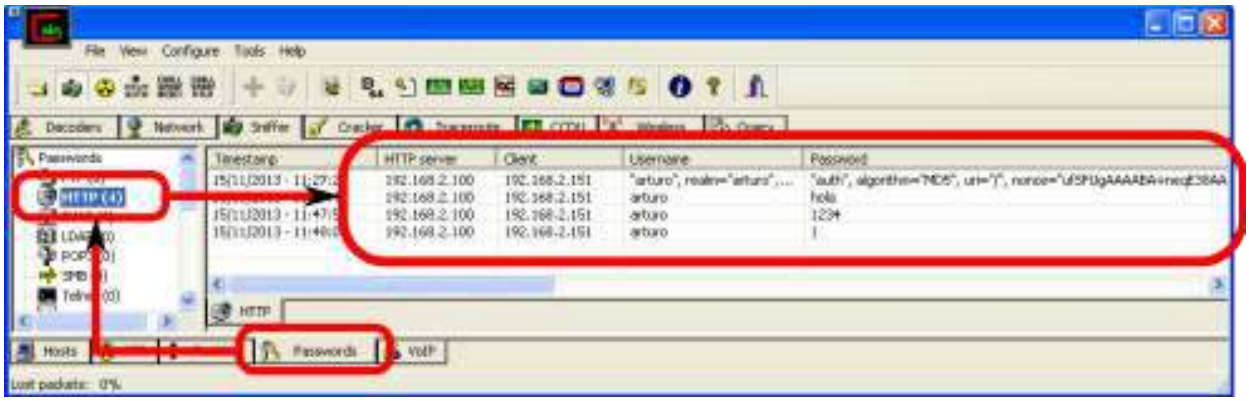
(6) Ahora ya estaremos en disposición de comenzar el ataque **Man in the Middle**. Para ello seleccionaremos la pestaña inferior "**APR**", y siguiendo las flechas que se muestran en la siguiente figura, configuraremos un nuevo ataque por envenenamiento ARP indicando entre que dos equipos nos situaremos: entre el cliente Web 192.168.2.151 y el servidor Proxy 192.168.2.100. A través de este envenenamiento de las tablas ARP de los equipos de la red Cain les engañará haciéndoles creer que él es el equipo con quien quieren comunicarse. Es decir, Cain hará una suplantación de identidad conocida en el mundo informático como **spoofing**.



(7) Para comenzar con el ataque, simplemente pulsaremos sobre el tercer icono superior referente a "**Start/Stop APR**".



(8) Por último, podremos observar si nos colocamos en la pestaña inferior "Passwords" como Cain va interceptando las autenticaciones realizadas entre el cliente y el servidor, mostrando la contraseña sin ningún tipo de problema en el caso que el tipo de autenticación sea **basic**, pero no así si es **digest**.



Aunque no hemos visto prácticamente nada de las posibilidades que nos ofrece el software Cain, tras esta pequeña práctica imagino que todo aquel que la haya realizado podrá advertir la gran potencia de ataque que tiene. Posteriormente, en el capítulo de ataques mostraremos alguna otra posibilidad.

Ej. Práctico 3.2.4: Proxy con Restricciones de Horarios para Usuarios

Tratando de completar a los ejercicios anteriores, en este caso configuraremos un Proxy donde tenga restringidas las franjas horarias en las que determinados usuarios pueden navegar a través de Internet. En concreto, trataremos de cumplir los requisitos resumidos en la siguiente tabla (respecto al *resto de requisitos generales, serán los mismos que en ejercicios anteriores*):

Tipo de Autenticación de Usuarios	Usuarios Permitidos	Franja Horaria de Acceso Restringido
digest	usuproxy1 usuproxy2	8:00-14:00 (*de lunes a viernes*)
	usuproxy3 usuproxy4	15:00-21:00 (*de lunes a viernes*)
	usuadmin1 usuadmin2	0:00-24:00 (*de lunes a domingo*)

Solución Ej. Pr. 3.2.4.I.- Configuración de Restricciones Horarias en Squid

Para mostrar la solución al ejercicio práctico propuesto nos basaremos en el fichero de configuración "squid.conf" del ejercicio anterior (*autenticación digest*), a la cual le añadiremos las ACLs y restricciones establecidas en la tabla anterior:

```
[root@mv1:/etc/squid3]# nano squid.conf
# Directivas globales
visible_hostname miproxy

http_port 3128
http_port 8088

cache_mem 32 MB
cache_dir ufs /var/spool/squid3 1000 16 256
maximum_object_size_in_memory 128 MB

access_log /var/log/squid3/access.log
cache_log /var/log/squid3/cache.log
cache_store_log /var/log/squid3/store.log

auth_param digest program /usr/lib/squid3/digest_pw_auth -c /etc/squid3/claves-digest.dat
auth_param digest children 5
auth_param digest realm proxy-digest

# Definimos las ACLs para definir los grupos de usuarios y franjas horarias que se usarán posteriormente para establecer restricciones de acceso:
acl grupo1 proxy_auth -i usuproxy1 usuproxy2
```

```
acl grupo2 proxy_auth -i usuproxy3 usuproxy4
acl grupo3 proxy_auth -i usuadmin1 usuadmin2
acl horario1 time M T W H F 8:00-14:00
acl horario2 time M T W H F 15:00-21:00
acl horario3 time M T W H F A S 0:00-24:00

acl cualquier-maquina src all
acl websprohibidas1 dstdomain "/etc/squid3/websprohibidas1.dat"
acl websprohibidas2 url_regex "/etc/squid3/websprohibidas2.dat"

# A los usuarios admin de la acl del grupo3 les permitimos el acceso desde cualquier equipo, cualquier día de la semana a cualquier hora
http_access allow cualquier-maquina grupo3 horario3 !websprohibidas1 !websprohibidas2
# Denegamos el acceso desde cualquier equipo (all) a las cuentas de usuario agrupadas en la ACL grupo1 en la franja horaria definida por la ACL horario2, y lo mismo para grupo2 en horario1:
http_access deny cualquier-maquina grupo1 horario2
http_access deny cualquier-maquina grupo2 horario1
# Permitimos el acceso desde cualquier equipo (all) a las cuentas de usuario agrupadas en la ACL grupo1 en la franja horaria definida por la ACL horario1, y lo mismo para grupo1 en horario2:
http_access allow cualquier-maquina grupo1 horario1 !websprohibidas1 !websprohibidas2
http_access allow cualquier-maquina grupo2 horario2 !websprohibidas1 !websprohibidas2
```

Para crear las cuentas de usuario necesarias para la autenticación digest, haremos uso de alguna de las dos estrategias mostradas en el ejercicio anterior. Después tan sólo nos quedará reiniciar el servicio para que surtan efecto los cambios en la configuración, aunque antes comprobaremos que no hayamos cometido algún error sintáctico o semántico (*"-k check" o "-k parse" nos proporcionan información equivalente*):

```
[root@mv1]# squid3 -k check
[root@mv1]# squid3 -k parse && /etc/init.d/squid3 restart
```

Por último, comprobaremos desde algún equipo cliente la correcta configuración del Proxy HTTP Caché.

3.3.- Control del Ancho de Banda mediante Squid

Si en los ejercicios prácticos anteriores hemos aprendido a implementar un Proxy HTTP Caché mediante Squid con la finalidad de poder controlar el acceso a los sitios Web de Internet, a continuación aprenderemos a controlar y limitar el ancho de banda disponible con Squid.

Teniendo en cuenta que todo el trafico generado por los clientes Web atraviesa al Proxy Squid para su control y análisis, no debería llamarnos excesivamente la atención la posibilidad que éste nos ofrece para decidir que cantidad de ancho de banda medido en bytes por segundo podrá acaparar cada cliente. En concreto, para la gestión del ancho de banda Squid dispone de unos limitadores denominados "**delay pools**" (*colas de retraso*), que podrían describirse como recipientes o almacenes de información donde se van acumulando los bytes que van generando las conexión Web de los clientes, y que se caracterizan por poder configurar la velocidad de salida.

Para todo ello, Squid dispone de un conjunto de directivas de configuración entre las cuales cabría destacar las siguientes:

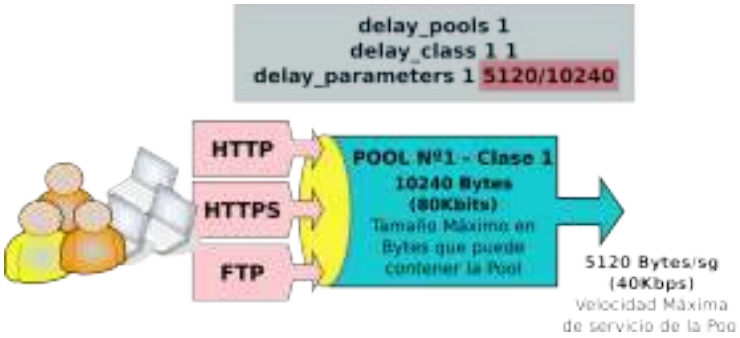
Directiva	Explicación / Ejemplo de Utilización
delay_pools	Nos permite definir el número de "delay pools" que usará Squid para la gestión del ancho de banda. Por defecto vale 0, que equivale a decir que por defecto Squid no establece ningún tipo de límite de ancho de banda a los clientes Web.
	delay_pools *número_de_pools* **delay_pools 4**
delay_class	Nos permite asignar a que clase pertenecerán cada una de las "delay pools" definidas mediante la directiva **delay_pools**. La asignación de la clase adecuada a cada una de las "delay pools" es vital para una correcta gestión el ancho de banda. Existen cinco clases diferentes las cuales se caracterizan por lo siguiente: **1.** La primera clase se caracteriza por no diferenciar entre equipos clientes ni usuarios, haciendo uso de un recipiente y límite de ancho de banda común para todos. Es decir, nos permite establecer un límite global en bytes por segundo el cual se repartirá, no necesariamente de manera equitativa, entre los clientes Web. **2.** La segunda clase esta pensada para controlar el ancho de banda que consume cada uno de los equipos de una red (*de clase C*) de manera independiente. Más concretamente, Squid analiza la dirección IPv4 (*32 bits, 4 octetos: a.b.c.d*) del equipo cliente y asigna un recipiente

	independiente con su correspondiente tamaño máximo de almacenaje en bytes, y su límite de velocidad máxima de salida en bytes por segundo, para cada una de las direcciones IPv4 que se diferencie en el último octeto (*d*). **3.** La tercera clase es similar a la anterior, pero pensada para limitar el ancho de banda de manera independiente entre diferentes redes (*de clase C*). Es decir, ante esta clase de "delay pool" Squid analiza las direcciones IPv4 de las conexiones Web que analiza, y asigna un recipiente independiente por cada dirección IPv4 que se diferencie en el tercer octeto (*c*). **4.** Complementa la clase anterior, pudiendo establecer un límite de ancho de banda por usuario autenticado, lo cual requiere la definición previa de las ACLs de tipo **auth_proxy** con sus correspondientes **http_access**. **5.** Permite establecer un límite de ancho de banda en base al etiquetado previo que se haya establecido a las conexiones Web.
	delay_class *número_de_pool número_de_clase* **delay_pools 3** **delay_class 1 1** **delay_class 2 2** **delay_class 3 4**
delay_access	Permite decidir a que equipos o usuarios se les aplicará los límites definidos en las "delay pools". Mediante el atributo allow de esta directiva confirmamos la aplicación del límite, y con deny, lo negamos.
	delay_access *número_de_pool allow/deny ACL* **acl red1 src 192.168.1.0/24** **delay_pools 1** **delay_class 1 2** **delay_access 1 allow red1** **delay_access 1 deny all**
delay_parameters	Nos permite establecer mediante sus parámetros los límites de ancho de banda a las "delay pools" definidas con las directivas **delay_pools** y **delay_class**. Para establecer estos límites será necesario especificar parejas de valores compuestas por la velocidad máxima de salida (*bytes por segundo*) y el tamaño máximo del recipiente (*bytes*). La número de parejas de valores a especificar dependerá de la clase de la Pool. En el caso de no querer poner un límite haremos uso de la pareja -1/-1.
	delay_parameters *número_de_pool pareja_valores_límites* **acl red1 src 192.168.1.0/24** **delay_pools 1** **delay_class 1 2** **delay_access 1 allow red1** **delay_access 1 deny all** **delay_parameters 1 -1/-1 128000/192000**

Para comprender mejor todo lo anterior, se proponen a continuación un conjunto de ejercicios prácticos.

Ej. Práctico 3.3.1: Control de Ancho de Banda de manera Global (Clase 1)

Configura Squid para evitar que el conjunto de todas las conexiones analizadas por él no supere 40Kbps (*5120 bytes/segundo*) del ancho de banda disponible. Además, el tamaño del recipiente que se encargará de ir almacenando los bytes de las conexiones Web será de 80Kbits (*10240 bytes*). Este límite tan restrictivo tan sólo tiene como finalidad poder percibir fácilmente su efecto por parte de los clientes Web a los que da servicio.



Solución Ej. Pr. 3.3.1.I.- Cómo Controlar el Ancho de Banda Global con Squid

Analizando el enunciado del ejercicio práctico propuesto podremos concluir que el comportamiento deseado para el Proxy Squid tan sólo implica el establecimiento de un límite de ancho de banda global, sin necesidad de tener que distinguir entre redes, equipos o usuarios, por lo que la elección de la clase de la Pool será de tipo 1. Según las directivas vistas previamente, la configuración de Squid podría quedar de la siguiente forma:

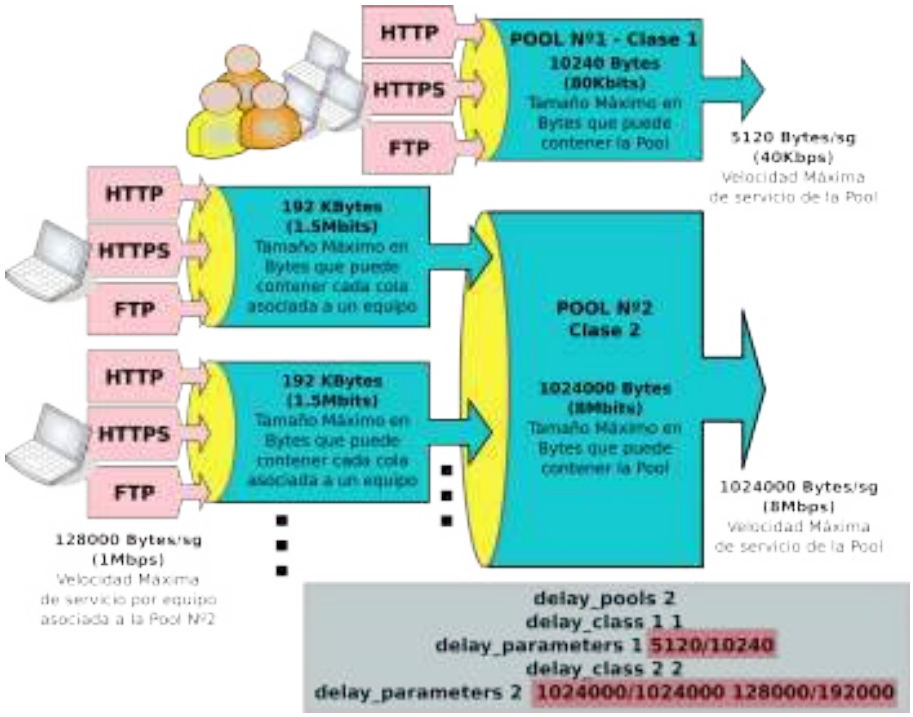
```
[root@mv1:/etc/squid3]# nano squid.conf
# Contenido del archivo de configuración del Proxy Squid
http_port 3128
cache_mem 128 MB
cache_dir ufs /var/spool/squid3 1000 16 256
maximum_object_size_in_memory 128 MB

access_log /var/log/squid3/access.log
cache_log /var/log/squid3/cache.log
cache_store_log /var/log/squid3/store.log

http_access allow all
delay_pools 1
delay_class 1 1
delay_access 1 allow all
delay_parameters 1 5120/10240
```

Por último, para surta efecto la configuración anterior, y así poder hacer las correspondientes comprobaciones desde algún cliente Web, deberemos chequear y reiniciar el servicio:

```
[root@mv1]# squid3 -k check
[root@mv1]# squid3 -k parse
[root@mv1]# /etc/init.d/squid3 restart
```

Ej. Práctico 3.3.2: Control de Ancho de Banda Global y por Equipo (Clase 1 y 2)

Suponiendo que Squid da servicio a dos redes diferentes, p.e. 192.168.100.0/24 y 192.168.200.0/24, configura Squid para limitarles el ancho de banda de tal forma que la primera red no pueda superar los 40Kbps (*5120 bytes/segundo*) del ancho de banda disponible, sin garantizar un reparto equitativo del ancho de banda entre los equipos que la forman, y que segunda red tenga un límite por equipo de 1Mbps (*128000 bytes/segundo*). Además, el tamaño del recipiente que se encargará de ir almacenando los bytes de las conexiones Web será de 80Kbits (*10240 bytes*) para la primera red, y de 1,5Mbits para la segunda red (*8Mbits en su conjunto*). La elección de estos límites tan restrictivos nos permitirán percibir fácilmente su efecto en los clientes Web a los que da servicio.



Solución Ej. Pr. 3.3.2.I.- Cómo Configurar dos Pools de clase 1 y 2

Para dar solución al ejercicio práctico se implementarán dos Pools independientes, una de clase 1, y otra de clase 2, ya que por un lado queremos establecer un límite global sin tener que distinguir entre equipos o usuarios para la primera de las redes, pero por otro lado queremos garantizar una determinada calidad de servicio (QoS) para cada uno de los equipos de la segunda red. De esta forma, las directivas de configuración de Squid podrían quedar de la siguiente manera:

```
[root@mv1:/etc/squid3]# nano squid.conf
# Contenido del archivo de configuración del Proxy Squid
http_port 3128
cache_mem 128 MB
cache_dir ufs /var/spool/squid3 1000 16 256
maximum_object_size_in_memory 128 MB
```

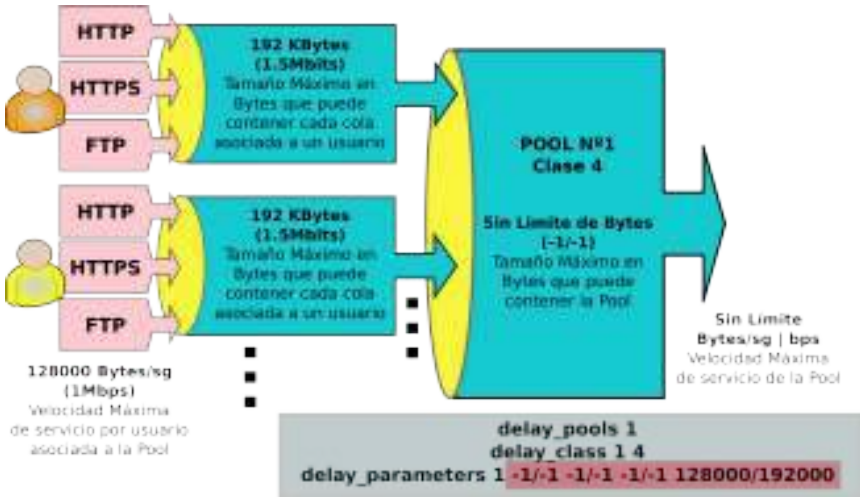
```
access_log /var/log/squid3/access.log
cache_log /var/log/squid3/cache.log
cache_store_log /var/log/squid3/store.log

acl red1 src 192.168.100.0/24
acl red2 src 192.168.200.0/24
http_access allow red1 red2
delay_pools 2
delay_class 1 1
delay_access 1 allow red1
delay_access 1 deny all
delay_class 2 2
delay_access 2 allow red2
delay_access 2 deny all
delay_parameters 1 5120/10240
delay_parameters 2 1024000/1024000 128000/192000
```

Por último, para surta efecto la configuración anterior, y así poder hacer las correspondientes comprobaciones desde algún cliente Web, deberemos chequear y reiniciar el servicio:

```
[root@mv1]# squid3 -k check
[root@mv1]# squid3 -k parse
[root@mv1]# /etc/init.d/squid3 restart
```

Ej. Práctico 3.3.3: Control de Ancho de Banda por Usuario (Clase 4)

Configurar Squid como Proxy no transparente con autenticación de usuarios, de tal forma que cada uno de los usuarios tenga un determinado máximo ancho de banda garantizado. Concretamente deberemos garantizar una calidad de servicio, QoS, correspondiente a 1Mbps (*128 Kbytes por segundo*), con un tamaño recipiente por usuario de 1,5Mbits (*192 Kbytes*). La elección de estos límites tan restrictivos nos permitirán percibir fácilmente su efecto en los clientes Web a los que da servicio Squid.



Solución Ej. Pr. 3.3.3.I.- Cómo Garantizar un Ancho de Banda por Usuario (QoS)

Para dar solución al ejercicio práctico se implementará una única Pool de clase 4 para poder limitar el ancho de banda a los usuarios que se autentifiquen. Al tratarse de una clase 4 será necesario especificar en la directiva **delay_parameters** tanto los límites por usuario, como los límites por máquina y red, y un límite a nivel global, los cuales se dejarán a un valor indefinido, -1/-1, al estar interesados únicamente en establecer límites por usuario. De esta forma, las directivas de configuración de Squid podrían quedar de la siguiente manera:

```
[root@mv1:/etc/squid3]# nano squid.conf
# Contenido del archivo de configuración del Proxy Squid
http_port 3128
cache_mem 128 MB
cache_dir ufs /var/spool/squid3 1000 16 256
maximum_object_size_in_memory 128 MB

access_log /var/log/squid3/access.log
cache_log /var/log/squid3/cache.log
cache_store_log /var/log/squid3/store.log

auth_param digest program /usr/lib/squid3/digest_pw_auth -c /etc/squid3/usuarios.digest
auth_param digest children 5
auth_param digest realm acceso_restringida
```

```
 acl usuarios proxy_auth REQUIRED
acl mired src 192.168.100.0/24

http_access allow usuarios mired
http_access deny all
delay_pools 1
delay_class 1 4
delay_access 1 allow usuarios mired
delay_access 1 deny all
delay_parameters 1 -1/-1 -1/-1 -1/-1 128000/192000
```

Por último, para surta efecto la configuración anterior, y así poder hacer las correspondientes comprobaciones desde algún cliente Web, deberemos chequear y reiniciar el servicio:

```
[root@mv1]# squid3 -k check
[root@mv1]# squid3 -k parse
[root@mv1]# /etc/init.d/squid3 restart
```

3.4.- Squid como Proxy Transparente

Hasta ahora hemos visto diferentes posibilidades de configuración del Proxy Squid como "no trasparente", lo cual nos ha permitido explotar diversas características interesantes del Proxy como pueden ser la autenticación de usuarios o el filtrado de URLs independientemente de si el protocolo usado para navegar es HTTP tcp/80 o HTTPS tcp/443. No obstante, este tipo de configuración también presenta algunos inconvenientes, entre los cuales, como principal inconveniente cabría señalar el tener que configurar el Proxy en todo aquel equipo cliente Web que desee navegar.

A continuación, con la finalidad de no tener que configurar los equipos clientes, ya sea por el engorro que ello supone al administrador de la red, o porque no existe la posibilidad de ello (*p.e. la red de un hotel o de una biblioteca*), se mostrará como configurar Squid para que sea transparente para todo cliente Web, con y sin portal cautivo.

Ej. Práctico 3.4.1: Proxy Transparente

Si en los ejercicios anteriores hemos configurado nuestro Proxy como No Transparente, a continuación diseñaremos un servicio **Proxy Trasparente**. Esto evitará que el usuario tenga que configurar su cliente Web para poder navegar a través del Proxy. Es decir, desde punto de vista del usuario final, navegar por Internet a través del Proxy le será totalmente transparente, sin tener que configurar nada en su equipo cliente. Para ello, en la configuración del servicio lo indicaremos explícitamente al indicar el puerto de servicio añadiendo "**transparent**", "**http_port 3128 transparent**", además de tener que redirigir mediante "**iptables**" los paquetes recibidos por los clientes que vayan destinados a la navegación por Internet (*p.e. HTTP/80*), al puerto configurado en el Proxy (*p.e. Proxy HTTP/3128*).

¡¡Importante!! Al configurar nuestro Proxy como transparente, evitamos tener que configurar en los equipos clientes el Proxy en sus navegadores Web, pero por contra perdemos el control en dos aspectos sumamente importantes: (1) **No podremos realizar autenticación de usuarios** como realizabamos con un Proxy no transparente, y (2) **no podremos filtrar las URLs ante el protocolo HTTPS**. El primero de los inconvenientes, si queremos forzar a una autenticación previa al usuario, lo podremos sortear configurando un portal cautivo en el servidor Proxy que obligue a la autenticación, tal como veremos en el siguiente ejercicio práctico. Pero el segundo problema no podremos solventarlo a no ser que convirtamos nuestro Proxy en un *Man In The Middle*, lo cual vulneraría la complicidad de los clientes. Es decir, mientras el Proxy no transparente hacía de intermediario entre el cliente y el servidor, redireccionando el tráfico HTTPS hacía el puerto 443, sin poder cachear nada al tratarse de una comunicación cifrada en modo seguro, en la configuración en modo transparente para camuflarse debe suplantar al cliente, de tal forma que el certificado de autenticación del sitio HTTPS recibido por el cliente es el del Proxy y no el del servidor, y a su vez el Proxy hace de cliente hacía el servidor HTTPS, provocando de esta forma que la comunicación sea totalmente insegura para el cliente: datos bancarios, contraseñas, etc.

Siguiendo con las especificaciones solicitadas en ejercicios anteriores, en la siguiente tabla se resumen las características del Proxy a configurar:

Puertos de Servicio **Cantidad Memoria RAM**	**Directorio Caché** **Tamaño Caché** **Nº Directorios y Subdirectorios** **Tamaño máximo de archivo a cachear**	**Webs no permitidas** **Auditoria Caché** **Auditoria Almanamiento**	**Autenticación** **Usuarios permitidos** **Equipos permitidos y no permitidos** **Auditoria acceso**
3128 transparent 32 MB	/var/spool/squid3 1GB 16/256 128MB	www.marca.com www.sport.es www.as.com **cache_log /var/log/squid3/cache.log** **cache_store_log /var/log/squid3/store.log**	No **Todos (acceso anónimo)** Todos (192.168.2.0/24) **access_log /var/log/squid3/access.log**

Solución Ej. Pr. 3.4.1.I.- Configuración de un Proxy Transparente

Para dar solución al problema planteado, al igual que se hizo en el ejercicio anterior, se modificará el correspondiente fichero de configuración asociado al servicio squid:

```
[root@mv1]# nano squid.conf
# Directivas globales
# Nombre del servidor proxy visible en los mensajes informativos enviados al cliente Web:
visible_hostname transproxy
# Puertos de servicio del proxy. Indicamos explícitamente que queremos que se comporte de manera transparente
http_port 3128 transparent
# Datos de la Cache:
## Ruta absoluta del directorio donde se cachearán los sitios Web visitados, su tamaño 1000MB, y el número de directorios (16) y subdirectorios (256)
cache_dir ufs /var/spool/squid3 1000 16 256
## Cantidad de memoria RAM que reservamos para este servicio proxy HTTP (p.e.): 32 MB
cache_mem 32 MB # Las unidades no pueden estar juntas a la cantidad 256MB!!!
## Máximo tamaño del archivo que una vez descargado será cacheado (p.e.): 128 MB
maximum_object_size_in_memory 128 MB
# Archivos de auditoría en relación con los accesos, contenidos visitados y cacheados:
access_log /var/log/squid3/access.log
cache_log /var/log/squid3/cache.log
cache_store_log /var/log/squid3/store.log
## Definición de ACLs para controlar el acceso a máquinas
acl cualquier-maquina src all
#acl acceso src 192.168.100.0/24
```

```
## Definición de ACLs para filtrar URLs de determinados sitios Web:
acl websprohibidas1 dstdomain "/etc/squid3/websprohibidas1.dat"
# Filtrado en base a las ACLs definidas anteriormente:
http_access allow cualquier-maquina !websprohibidas1
```

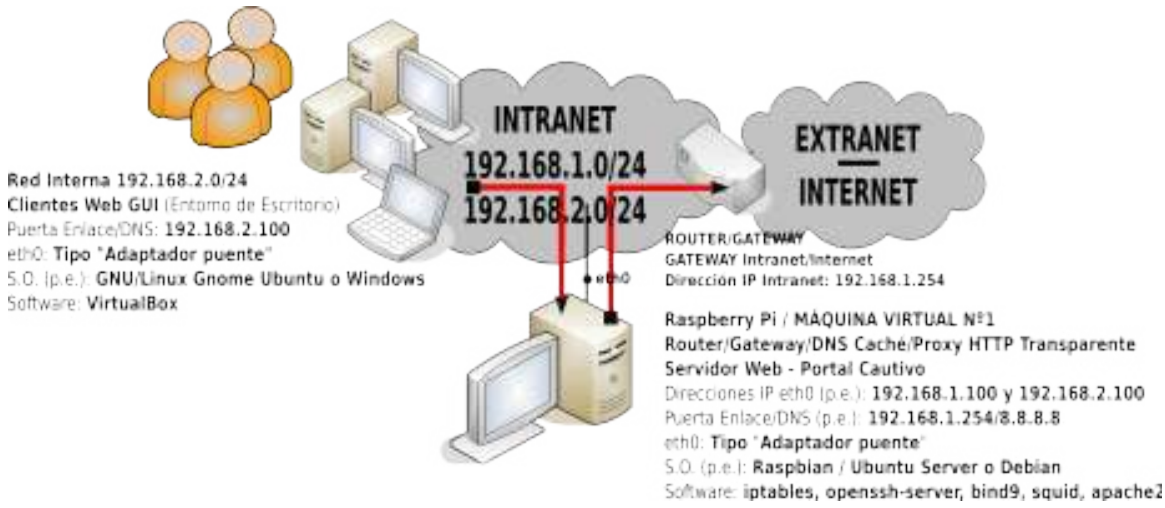
Por último, tan sólo tendremos que reiniciar el servicio Proxy para que surtan efecto los cambios anteriores realizados en la configuración y redirigir (*NAT PREROUTING*) al puerto del servicio Proxy 3128 todos aquellos paquetes TCP/IP que reciba el equipo servidor (*es el gateway de la Intranet*) y que vayan dirigidos al puerto 80 (*-p tcp/udp --dport 80*). Después tan sólo nos quedará comprobar que desde el cliente se puede navegar sin necesidad de una configuración previa:

```
[root@mv1]# iptables -t nat -A PREROUTING -s 192.168.2.0/24 -p tcp --dport 80 \
                        -j REDIRECT --to-port 3128
[root@mv1]# iptables -t nat -A PREROUTING -s 192.168.2.0/24 -p udp --dport 80 \
                        -j REDIRECT --to-port 3128
[root@mv1]# squid3 -k check
[root@mv1]# squid3 -k parse && /etc/init.d/squid3 restart
```

Ej. Práctico 3.4.2: Proxy Transparente con Portal Cautivo

Con la finalidad de cumplimentar el ejercicio práctico anterior, en el presente ejercicio práctico mostraremos como implementar un **Proxy Transparente** pero sin renunciar a la autenticación de los usuarios a través de un **portal cautivo**. Por compatibilidad con cosas vistas en ejercicios anteriores, los usuarios que podrán autenticarse a través del portal cautivo serán usuarios que previamente existan en un archivo creado mediante la herramienta **htdigest**.

Un portal cautivo, no es más que un sitio Web local al Proxy donde a través de un formulario se le solicitan credenciales al cliente, y en caso de una autenticación exitosa, se le permitirá navegar hacía Internet. Aunque existen soluciones integrales a la implementación de un portal cautivo, aquí haremos uso de una solución particular al no presentar gran dificultad su diseño y al guardar una estrecha relación con lo ya visto en las prácticas de diseño de firewalls.



Solución Ej. Pr. 3.4.2.I.- Proxy Transparente con Portal Cautivo

Como el diseño de un Proxy transparente ya se mostró en el ejercicio anterior, aquí únicamente nos centraremos en la implementación de un portal cautivo. Para ello, en primer lugar, forzaremos a los clientes que desean navegar hacía internet vía HTTP tcp/80 o HTTPS tcp/443 a tener que autenticarse a través de un formulario que tendrá la página de inicio "index.php" del sitio Web local servido por el propio Proxy. Esto lo conseguiremos simplemente redireccionando toda solicitud HTTP tcp/80 o HTTPS tcp/443 hacía el propio servidor Proxy:

```
[root@mv1]# iptables -t nat -A PREROUTING -s 192.168.2.0/24 -p tcp --dport 80 \
             -j DNAT --to 192.168.2.100:80
[root@mv1]# iptables -t nat -A PREROUTING -s 192.168.2.0/24 -p tcp --dport 443 \
             -j DNAT --to 192.168.2.100:443
```

¡¡Importante!! Para evitar el tener que ejecutar los comandos anteriores cada vez que queramos hacer esta práctica, lo más cómodo sería modificar el script **conf-red** (*o hacer una copia de él*) que ya creamos durante el capítulo de iptables, y que hacíamos que se ejecutase al iniciarse la máquina, lo que nos permitirá ejecutarlo cuando deseemos.

```
[root@mv1]# nano /etc/init.d/conf-red
#!/bin/bash
/etc/init.d/networking stop
ifconfig eth0 down
ifconfig eth0 192.168.1.100
route add default gw 192.168.1.254
ifconfig eth0:alias1 192.168.2.100
# Si el equipo servidor lo tenemos configurado como Servidor DNS Caché (si no 8.8.8.8):
echo "nameserver 127.0.0.1" > /etc/resolv.conf
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -t nat -F
iptables -t nat -A POSTROUTING -j MASQUERADE
iptables -t nat -A PREROUTING -s 192.168.2.0/24 -p tcp --dport 80 \
-j DNAT --to 192.168.2.100:80
iptables -t nat -A PREROUTING -s 192.168.2.0/24 -p tcp --dport 443 \
-j DNAT --to 192.168.2.100:443
iptables -t filter -F
iptables -P FORWARD DROP
iptables -A FORWARD -p udp --dport 53 -j ACCEPT
iptables -A FORWARD -p icmp -s 192.168.2.0/24 -j ACCEPT
iptables -A FORWARD -p tcp --dport 3128 -s 192.168.2.0/24 -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
[root@mv1]# chmod +x /etc/init.d/conf-red
[root@mv1]# echo "bash /etc/init.d/conf-red" >> /etc/init.d/rc.local
```

De esta forma, los usuarios que desde sus equipos clientes intenten navegar a través de Internet, verán en su navegador el sitio Web servido por el propio Proxy. Para ello, por ejemplo, haremos uso de Apache (*apache2 + PHP*):

```
[root@mv1]# apt-get install apache2 libapache2-mod-php5
```

Para evitar configuraciones innecesarias, nos aprovecharemos de los sitios Web que Apache dispone por defecto en **/etc/apache2/sites-available**. Concretamente, Apache tiene habilitado un servicio HTTP tcp/80 por defecto llamado "**000-default**" en **/etc/apache2/sites-enabled**, pero aunque disponible, no se encuentra habilitado por defecto su servicio HTTPS tcp/443. Para ello, haciendo uso de los comandos **a2ensite** (*apache2 enable site*) y **a2enmod** (*apache2 enable module*) habilitaremos el sitio "**default-ssl**" y el módulo que le da soporte "**ssl**":

```
[root@mv1]# a2ensite default-ssl
[root@mv1]# a2enmod ssl
[root@mv1]# ls /etc/apache2/sites-enabled/
000-default  default-ssl
```

Por último, antes de reiniciar el servicio Apache, personalizaremos las páginas de inicio de los sitios Web servidos por defecto, "**/var/www/index.html**" y "**/var/www/index.php**". En concreto, el archivo **index.html** contendrá únicamente un redireccionamiento HTML hacía la página **https://*direccion_ip_proxy*/index.php** con la finalidad de garantizar una comunicación

segura entre el cliente y servidor del portal cautivo, y el **index.php** contendrá un formulario HTML que enviará el login y password introducidos para la autenticación de los usuarios a través del portal cautivo, y su validación.

Para más detalle, aunque cada uno podrá diseñarlo como desee, la nueva página de inicio **index.php** a servir constará de un formulario HTML donde el usuario introducirá sus credenciales, las cuales serán enviadas al servidor para que este las compruebe vía HTTPS. En caso de éxito, se ejecutará un comando **iptables** mediante la función PHP **exec($comando,$salida,$estado_salida)** que dará acceso a la máquina cliente a navegar hacía Internet, y de esta forma dejar de estar cautivo:

¡¡Aclaración!! Por cuestiones de compatibilidad con lo visto hasta ahora, la **validación del login y password** que serán introducidos por el usuario a través del formulario HTML (*método de envio POST, method=post, $_POST['nombre'] y $_POST['pass']*) que incluye el index.php, se basará en la **autenticación digest** llevada a cabo en los últimos ejercicios prácticos. Es decir, los usuarios que se validarán a través del portal cautivo habrán tenido que haberse creado previamente mediante la herramienta **htdigest**, tal como se ha realizado hasta ahora (*repasar el ejercicio 3.1.2*). Por ejemplo:

[root@mv1]# **htdigest** [-c] [ruta_archivo_de_autenticación] [realm] [nombre_usuario]
[root@mv1]# **htdigest -c /etc/squid3/claves-digest.dat portal-cautivo usuario1**

Una vez creadas las cuentas de usuario que podrán autenticarse a través del portal cautivo será necesario introducir el código PHP en el index.php para que corrobore el login y password introducidos. Para ello, mediante la función PHP **md5** (*repasar el ejercicio 3.1.2*) comprobaremos si el login y password introducidos se corresponden con los registrados en el fichero de cuentas de usuario digest:

```
$ruta_digest = "/etc/squid3/claves-digest.dat";
$fichero = fopen ($ruta_digest,"r");
$verificado = 0;
while ( preg_match("/:/", $linea = fgets($fichero) ) )
{
        $linea = rtrim ($linea);
        $campos = explode (':',$linea);
        $realm = "portal-cautivo";
        $posible_pass = md5($_POST['nombre'].":".$realm.":".$_POST['pass']);
        if ( $_POST['nombre'] == $campos[0] && $posible_pass == $campos[2] )
        {
                $verificado = 1;
        }
}
```

Tal como se diseñan los sitios Web actuales sería más preferible hacer la verificación a través de los usuarios y contraseñas introducidos previamente en una base de datos MySQL, pero el caso práctico que se trata de mostrar aquí se complicaría en exceso.

[root@mv1]# **mv /var/www/index.html /var/www/index.php**
[root@mv1]# **nano /var/www/index.php**

```
<? session_start(); ?>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8">
<style type="text/css">
	div.div1 { width: 100%; margin-top: 100px; text-align: center; }
	fieldset.f1 { width: 50%; margin-top: 100px; margin-left: auto;
	margin-right: auto; background-color: yellow; text-align: center; }
	fieldset.f2 { width: 50%; margin-left: auto; margin-right: auto;
	background-color: black; color: white; text-align: center; }
</style>
<div class="div1">
	<fieldset class="f1">
	<legend>Control de Acceso</legend>
	<form name="form1" method="post" action="">
		Nombre: <input type="text" name="nombre" size="20"><br><br>
		Password: <input type="password" name="pass" size="20"> <br><br>
		<input type="submit" name="boton" value="Enviar">
	</form>
	</fieldset>
</div>
<?
if ( isset($_POST['boton']) )
{
	$ruta_digest = "/etc/squid3/claves-digest.dat";
	$fichero = fopen ($ruta_digest,"r");
	$verificado = 0;
	while ( preg_match("/:/", $linea = fgets($fichero) ) )
	{
		$linea = rtrim ($linea);
		$campos = explode (':',$linea);
		$realm = "portal-cautivo";
		$posible_pass = md5($_POST['nombre'].":".$realm.":".$_POST['pass']);
		if ( $_POST['nombre'] == $campos[0] && $posible_pass == $campos[2] )
		{
			$verificado = 1;
		}
	}
	if ( $verificado == 1 )
	{
		$_SESSION['comprobacion'] = "ok";
		$_SESSION['nombreusu'] = $_POST['nombre'];
		$_SESSION['ip'] = $_SERVER['REMOTE_ADDR'];
		$comando1 = "sudo iptables -t nat -I PREROUTING 1 -s ".
			$_SESSION['ip']." -p tcp --dport 80 -j REDIRECT --to-port 3128";
		$comando2 = "sudo iptables -t nat -I PREROUTING 1 -s ".
			$_SESSION['ip']." -p tcp --dport 443 -j ACCEPT";
		$comando3 = "sudo iptables -I FORWARD 1 -s ".
			$_SESSION['ip']." -p tcp --dport 443 -j ACCEPT";
```

```
                exec($comando1,$salida1,$estado_salida1);
                exec($comando2,$salida2,$estado_salida2);
                exec($comando3,$salida3,$estado_salida3);
                if ( ( $estado_salida1 == 0 ) && ( $estado_salida2 == 0 ) &&
                        ($estado_salida3 == 0) )
                {
                        header('Location: https://192.168.2.100/acceso.php');
                }
                else
                {
                        header('Location: https://192.168.2.100/error.php');
                }
        }
        else
        {
                $_SESSION['comprobacion'] = "error";
                ?>
                <br>
                <div class="div1">
                <fieldset class="f2">
                <legend style="color: red;">Respuesta ante la autenticación</legend>
                Lo sentimos ... Tu login o password no son correctos!!!
                <br>
                Tu máquina con dirección IP <? echo $_SERVER['REMOTE_ADDR']; ?>
                no tendrá acceso a Internet mientras no te autentiques correctamente.
                </fieldset>
                </div>
                <?
        }
unset($_POST['boton']);
}
?>
```

¡¡Aclaración!! El **$comando3** solamente será necesario que se ejecute si la política por defecto de FORWARD es DROP o está denegado explícitamente el FORWARD para el puerto **tcp/443**. Obviamente, la dirección IP 192.168.2.100 habrá que sustituirla por la que tenga el equipo servidor Proxy durante la práctica.

Es importante señalar que al tratarse de un **Proxy transparente** el tráfico **tcp/443** no puede pasar a través de squid a no ser que se configure el Proxy como un Man in the Middle, de hay que tal como se puede advertir a través del siguiente código PHP las conexiones tcp/80 si son redireccionadas hacía el puerto tcp/3128 del Proxy en el caso de una correcta autenticación, pero las tcp/443 no se redireccionan y son sencillamente aceptadas.

```
$comando2 = "sudo iptables -t nat -I PREROUTING 1
                -s ".$_SESSION['ip']." -p tcp --dport 443 -j ACCEPT";
$comando3 = "sudo iptables -I FORWARD 1
```

```
-s ".$_SESSION['ip']." -p tcp --dport 44 3 -j ACCEPT";
```

Por otro lado, es importante comprender la importancia del comando PHP **exec()**, el cual nos permite ejecutar comandos del sistema. En concreto, tiene el siguiente formato:

exec($comando [, $salida, [$estado_salida]]);

El primer argumento, **$comando**, hace referencia al comando del sistema que deseamos ejecutar.

El segundo argumento, **$salida** (*opcional*), es un array o vector donde en cada una de sus posiciones se almacenan las diferentes líneas que se producen como respuesta. Por ejmplo, $salida[0] se correspondería con la primera línea de los resultados mostrados por consola al ejecutar el comando indicado.

El tercer argumento, **$estado_salida** (*opcional*), nos informa de si tuvo éxito o no la ejecución del comando indicado. En concreto, adopta el valor "**0**" en caso de éxito, y un valor distinto de cero, como código de error.

En el caso de que la autenticación sea válida se mostrará en una segunda página PHP compuesta por un simple formulario de Google para indicar una posible búsqueda:

```
[root@mv1]# nano /var/www/acceso.php
<? session_start(); ?>
<?
if ( $_SESSION['comprobacion'] == "ok" )
{ ?>
        <h1 style="text-align: center; color:blue;">
        <? echo "Enhorabuena ".$_SESSION['nombreusu']."!! ... ya puedes Navegar desde tu
        equipo cliente c on IP ".$_SESSION['ip']; ?>
        </h1>
        <div style="width: 100%; text-align: center;">
        <form method="get" action="http://www.google.com/custom">
        <table bgcolor="#FFFFFF" cellspacing="0" cellpadding="0" border="0">
        <tr>
        <td><a href="http://www.google.com/custom/"> </a> </td>
        <td> <input type="text" name="q" size="25" maxlength="255" value=""> </td>
        <td> <input type="submit" name="btnG" value="Buscar en Google"> </td>
        </tr>
        </table>
        </form>
        </div>
<?
// Destruimos las variables de sesión
$_SESSION = array();
// Para borra la sesión completamente, borraremos también la cookie de sesión.
if ( ini_get("session.use_cookies") ) {
        $params = session_get_cookie_params();
        setcookie (session_name(), '', time() - 42000,
```

```
                $params["path"], $params["domain"],
                $params["secure"], $params["httponly"]
        );
}
// Destruimos la sesión abierta
session_destroy();
}
else { ?>
<fieldset style="width: 100%; margin-top: 100px; text-align: center;
        color: blue; background-color: red;">
<legend>Problema de Seguridad</legend>
No deberías tratar de acceder por aquí. Debes autenticarte primero. <br><br> Gracias.
</fieldset>
<?  } ?>
```

Por último, tan sólo tendremos que permitir al servicio Apache que pueda modificar el servicio firewall /sbin/**iptables** mediante los comandos **exec($comando)**, ya que por defecto, se requieren privilegios de **root**. Para ello, añadiremos los suficientes privilegios en **sudo**:

¡¡Aclaración!! Existen distribuciones GNU/Linux (*p.e. Debian*) donde el software de delegación de privilegios a usuarios **sudo** no viene instalado por defecto. En esos casos, será necesario instalarlo previamente.

```
[root@mv1]# apt-get install sudo
[root@mv1]# visudo

# User privilege specification
root ALL=(ALL) ALL
# Añadir la siguiente línea: Permitirá a Apache poder modificar la configuración de iptables
www-data ALL = (ALL) NOPASSWD: /sbin/iptables
...
```

Ahora ya podremos reiniciar Apache y comprobar el correcto funcionamiento del Proxy con portal cautivo:

```
[root@mv1]# /etc/init.d/apache2 restart
```

¡¡Importante!! Un aspecto inseguro a tener en cuenta es que las credenciales introducidas por el usuario a través del formulario del portal cautivo pueden viajar vía HTTP de manera insegura hacia el servidor pudiendo ser interceptadas por cualquier *sniffer*. Para evitar esta situación, una solución sería forzar a que el portal cautivo fuese servido vía HTTPS. Para conseguirlo existen varias posibilidades de las cuales aquí se destacarán dos:

(1ª opción) La opción más sencilla, sin lugar a dudas, es que la página de inicio (*p.e. index.html*) servida por el portal cautivo vía HTTP sea un documento que contenga un simple redireccionamiento HTML, **<meta http-equiv="refresh" content="*retardo en sg*;URL=*destino_redireccionamiento*">** (*en realidad su finalidad es para refrescar los contenidos*

HTML que se visualizan), que de manera automática redireccione al cliente Web hacía el sitio Web HTTPS en modo seguro (*portal cautivo en modo seguro*):

```
[root@mv1]# more /var/www/index.html
<html>
        <head>
                <meta http-equiv="refresh" content="0;URL=https://192.168.2.100">
        <head>
        <body> </body>
</html>
```

Analizando el código HTML anterior, podrá advertirse que la página de inicio del portal cautivo HTTP se refrescará al cabo de 0 segundos redirigiendo la petición hacía el portal cautivo HTTPS del Proxy (*p.e. 192.168.2.100*).

Teniendo en cuenta que el directorio **DocumentRoot /var/www** donde se encuentran los documentos servidos por los sitios Web que por defecto están implementados en Apache son los mismos tanto vía HTTP tcp/80 como vía HTTPS tcp/443, será necesario indicar explícitamente que sus páginas de inicio son diferentes mediante el uso de la directiva **DirectoryIndex**: **index.html** para el servicio HTTP tcp/80 e **index.php** el servicio HTTPS tcp/443.

```
[root@mv1]# nano sites-enabled/000-default
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        DirectoryIndex index.html
        DocumentRoot /var/www
…
</VirtualHost>
```

```
[root@mv1]# nano sites-enabled/default-ssl
<IfModule mod_ssl.c>
<VirtualHost _default_:443>
        ServerAdmin webmaster@localhost
        DirectoryIndex index.php
        DocumentRoot /var/www
…
</VirtualHost>
</IfModule>
```

(2ª opción) Otra posibilidad sería utilizando el módulo **rewrite** de Apache, el cual habrá que habilitarlo previamente, y que nos permitirá formatear la URL devuelta del servidor al cliente. En concreto, haciendo uso de las directivas de este módulo redireccionaremos las peticiones recibidas por el servidor vía HTTP tcp/80 hacia el servicio HTTPS tcp/443, para lo cual modificaremos por completo (*se aconseja hacer una copia previamente*) el fichero de configuración de Apache asociada al servicio tcp/80 introduciendo las siguientes líneas:

```
[root@mv1]# a2enmod rewrite
```

```
[root@mv1]# nano sites-enabled/000-default
<VirtualHost *:80>
        RewriteEngine on
        ReWriteCond %{SERVER_PORT} !^443$
        RewriteRule ^/(.*) https://192.168.2.100/index.php
</VirtualHost>
[root@mv1]# ls sites-enabled/
000-default  default-ssl
[root@mv1]# /etc/init.d/apache2 restart
```

Para terminar crearemos un script que se ejecutará automáticamente cada 5 minutos mediante el servicio de automatización de tareas **cron**, con la finalidad de dar de baja y cancelar el acceso en el firewall a aquellos equipos clientes que ya no estén activos.

¡¡Aclaración!! El siguiente **script**, llamado **dar-de-baja-desconectados.sh**, es una de las posibles soluciones que podrían aplicarse para dar de baja a aquellos equipos de la Intranet que ya se han desconectado, y que previamente se les dio acceso a Internet a través del portal cautivo.
Para ello, tal como se puede advertir consultando el script, se ha hecho uso entre otros muchos de los siguientes programas/comandos que merecen una especial mención:

(a) nmap: potente herramienta pensada para la exploración de redes y el sondeo de la seguridad de los puertos de un equipo. Dentro de las muchas opciones que nos permite esta aplicación (*man nmap*), aquí haremos uso de "**-sP**", **sondeo ping**, que permite determinar que equipos están vivos en nuestra red. La opción "**-T5**" selecciona la plantilla de temporizado y rendimiento, indicándole que invierta el menor tiempo posible.

nmap -sP -T5 192.168.2.0/24

(b) comm: permite comparar el contenido de dos ficheros ordenados, suministrando como resultado tres columnas (*man comm*). La 1ª columna nos informa de las filas que tiene el primer archivo y que no tiene el segundo. Es decir, en que difiere el primer archivo del segundo. La 2ª columna nos informa de las filas que tiene el segundo fichero y que no tiene el primero. Es decir, en que difiere el segundo archivo del primero. La 3ª y última columna nos informa de las filas que tienen en común ambos archivos. En concreto, en el script compararemos mediante **comm** dos archivos, **ultimos-eq.activos** y **eq.activos**, los cuales contienen respectivamente las direcciones IP de los equipos que estaban activos hace 5 minutos, y las direcciones IP de los equipos que se encuentran activos actualmente. Por tanto, la comparación nos informará de (*1ª columna*) equipos que estaban activos hace 5 minutos, y que ahora ya no lo están, (*2ª columna*) equipos que se han activado recientemente y que antes no estaban, y (*3ª columna*) los equipos que hace 5 minutos estaban activos, y lo siguen estando ahora. Como nosotros tan solo estamos interesados en los equipos que han dejado de estar activos (*inactivos*) para luego eliminar sus respectivas reglas del firewall, sólo nos interesaremos por la 1ª columna, y descartaremos la 2ª y 3ª columna. De hay los parámetros "**-2**" y "**-3**":

comm -2 -3 /etc/squid3/**ultimos-eq.activos** /etc/squid3/**eq.activos** > /etc/squid3/**eq.inactivos**

Por último, aclarar que las comillas graves `*comando*`, o comillas francesas, nos permiten

hacer referencia al resultado de evaluar un comando, y es equivalente a $(*comando*):

CONTADOR=`expr $CONTADOR + 1`
CONTADOR=$(expr $CONTADOR + 1)

[root@mv1]# **nano /etc/squid3/dar-de-baja-desconectados.sh**

```
#!/bin/bash
# Mediante nmap inspeccionamos que equipos clientes hay activos en nuestra red.  Sus direcciones ip se almacenan de manera ordenada (sort) en un fichero de texto llamado eq.activos:
nmap -sP -T5 192.168.2.0/24 | grep "192.168.2" | cut -d" " -f5 | sort > /etc/squid3/eq.activos
# Comparamos los equipos activos detectados con los que hubo hace 5 minutos.  De esta forma detectamos que equipos ya no están activos. Se almacenan en una fichero llamado eq.inactivos
comm -2 -3 /etc/squid3/ultimos-eq.activos /etc/squid3/eq.activos > /etc/squid3/eq.inactivos
comm -2 -3 /etc/squid3/ultimos-eq.activos /etc/squid3/eq.activos >> /etc/squid3/eq .inactivos.bkup
# Guardamos las reglas de iptables actuales en archivos para su consulta más rápida
/sbin/iptables -t nat -L PREROUTING --line-numbers > /etc/squid3/firewall-nat-pre
/sbin/iptables -L FORWARD --line-numbers > /etc/squid3/firewall-forward
# Contamos cuantos equipos inactivos se han encontrado, para a continuación mediante un bucle buscarlos en las reglas de iptables y eliminar el posible acceso a Internet que tuvieran
NUM=`cat /etc/squid3/eq.inactivos | wc -l`
echo "Hay $NUM equipos inactivos ... Se van a borrar de iptables!!"
CONTADOR1=1
while test $CONTADOR1 -le $NUM
do
        IP=`cat /etc/squid3/eq.inactivos | head -$CONTADOR1 | tail -1`
        # Comenzaremos borrando las reglas asociadas a la IP de NAT y luego de FORWARD
        NUM2=`cat /etc/squid3/firewall-nat-pre | grep "$IP" | wc -l`
        echo "La ip del equipo inactivo es $IP y tiene $NUM2 reglas de NAT para borrar ..."
        CONTADOR2=1
        while test $CONTADOR2 -le $NUM2
        do
                # Para eliminar las reglas de iptables averiguamos que posición ocupan
                NUMREGLA=`cat /etc/squid3/firewall-nat-pre | grep "$IP" | cut -d" " -f1 | head -1`
                # Con la opción -D (delete) eliminamos el número de regla deseada
                if /sbin/iptables -t nat -D PREROUTING $NUMREGLA
                then
                        echo "Se ha borrado la regla NAT $NUMREGLA asociada a $IP !!"
                else
                        echo "Se ha detectado algún error al borrar $IP de iptables NAT ..."
                fi
                CONTADOR2=`expr $CONTADOR2 + 1`
        done
        # Si el eq. inactivo tuviera reglas de FORWARD asociadas deben eliminarse (p.e. tcp/443)

        NUM2=`cat /etc/squid3/firewall-forward | grep "$IP" | wc -l`
        echo "La ip del equipo es $IP y tiene $NUM2 reglas de FORWARD para borrar ..."
        CONTADOR2=1
```

```
        while test $CONTADOR2 -le $NUM2
        do
                NUMREGLA=`cat /etc/squid3/firewall-forward | grep "$IP" | cut -d" " -f1 | head -1`
                if /sbin/iptables -D FORWARD $NUMREGLA
                then
                        echo "Se ha borrado la regla FORWARD $NUMREGLA asociada a $IP !!"
                else
                        echo "Se ha detectado algún error al borrar $IP de iptables FORWARD ..."
                fi
        CONTADOR2=`expr $CONTADOR2 + 1`
        done
        CONTADOR1=`expr $CONTADOR1 + 1`
done
# Por último, registramos cuales han sido los últimos equipos activos detectados, para compararlos dentro de cinco minutos con los que haya en ese momento:
cat /etc/squid3/eq.activos > /etc/squid3/ultimos-eq.activos
```

Después tan sólo tendremos que configurar el **cron** para garantizar que el script anterior se ejecute periódicamente cada 5 minutos:

```
[root@mv1]# chmod +x /etc/squid3/dar-de-baja-desconectados.sh
[root@mv1]# crontab -e
# Añadir la siguiente línea: Ejecutará el script anterior cada cinco minutos */5
*/5 * * * * bash /etc/squid3/dar-de-baja-desconectados.sh
```

Ej. Práctico 3.4.3: Evitar un Portal Cautivo mediante el Encapsulamiento HTTP/HTTPS bajo un Túnel DNS

En este ejercicio práctico se propone la implementación de un Túnel **DNS** con la finalidad de detectar alguna de las vulnerabilidades que ofrece un portal cautivo. Es decir, a lo largo del ejercicio se explicará dicha vulnerabilidad, y se detallarán los pasos necesarios para implementar un Túnel **DNS** mediante el software **iodine**, lo que nos permitirá encapsular sobre dicho túnel las tramas HTTP/HTTPS que generamos cuando navegamos.

Solución Ej. Pr. 3.4.3.I.- Cómo Saltarse un Portal Cautivo mediante un Túnel DNS

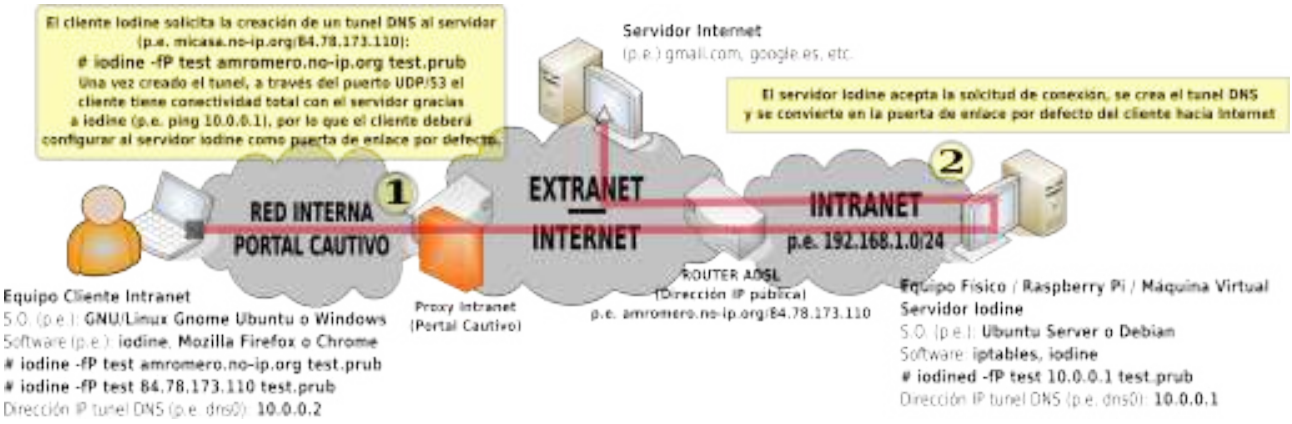
Una vez implementado el portal cautivo, si repasamos el protocolo de actuación entre el cliente y el servidor, podremos detectar que tiene un pequeño resquicio de vulnerabilidad que podremos aprovechar para saltárnoslo, y evitar tener que autenticarnos. Esto nos permitiría saltarnos esta herramienta de control de acceso en la mayor parte de los hoteles, bibliotecas, centros comerciales, o aeropuertos que la implementan. Es decir, cuando un cliente quiere intentar navegar en un entorno donde hay implementado un portal cautivo, a grandes rasgos, sucede lo siguiente:

(1) El cliente, normalmente de manera inalámbrica, se conecta a la red WIFI abierta del entorno en que se encuentra.
(2) A continuación, el cliente inicia un cliente Web (p.e. Mozilla Firefox o Chrome) con la intención de navegar por Internet. Para ello coloca en la barra de direcciones del navegador "http://" seguido del nombre de dominio correspondiente al servidor HTTP/HTTPS al que se desea conectar.
(3) Al darle al botón de "Ir" o "Navegar", siendo transparente para el usuario, en primer lugar el equipo cliente hace una solicitud de resolución DNS para conocer cual es la dirección IP pública correspondiente al nombre de dominio del servidor al que se desea conectar. Para ello, a no ser que la resolución ya se encuentre cacheada en algún servidor DNS Caché de la Intranet, la solicitud DNS del cliente sale a Internet para averiguarlo.
(4) Una vez que el equipo cliente recibe el resultado de la resolución es cuando por fin intenta realizar una conexión HTTP/HTTPS (tcp 80/443) a la dirección IP asociada al servidor Web.
(5) El equipo Gateway/Proxy/Firewall detecta que el cliente quiere navegar vía tcp 80/443, y mediante una NAT PREROUTING redirige la petición hacia el portal cautivo solicitándonos una autenticación.

Según todo lo anterior, podremos advertir que en el paso **(3)** la solicitud DNS generada por nuestro equipo navega por Internet sin tener que pasar por el control de acceso del portal cautivo, aspecto que muchos desarrolladores de software han tenido en cuenta para crear herramientas (*p.e. iodine*) que permitan navegar al usuario vía protocolo DNS udp/53, mediante la creación previa de lo que se denomina un **tunel DNS**, consistente en encapsular tramas del protocolo HTTP/HTTPS dentro de las tramas DNS. No obstante, cabe resaltar que existen de igual forma otras herramientas software de detección que tras analizar las tramas DNS pueden detectar nuestra intención y descartarla.

En definitiva, tal como vamos a poder comprobar mediante la implementación del siguiente ejercicio práctico, se va a implementar una "especie de VPN" mediante un túnel DNS udp/53, que nos permitirá encapsular todo el trafico que genere el cliente hacia el servidor, para que este a su

vez lo redireccione hacia quien corresponda (*p.e. servidor Web www.google.es*).



¡¡Advertencia!! Antes de empezar a implementar la práctica deberíamos tener en cuenta que un túnel DNS puede ser una vía de escape para poder navegar a través de una red controlada mediante un portal cautivo sin necesidad de autenticarnos, pero teniendo presente las limitaciones de ancho de banda que presenta a consecuencia de querer tunelizar tráfico HTTP/HTTPS sobre el protocolo DNS udp/53, el cual esta pensado originalmente para un trasiego de tramas de tamaño reducido y con necesidad de ancho de banda muy reducidas. Los desarrolladores de **iodine** advierten que el ancho de banda resultante que un equipo cliente puede llegar a obtener es de **680 Kbit/s** de subida y **2,3 Mbit/s** de bajada en una Intranet cableada, y de **50 Kbit/s** de subida y **200 Kbit/s** de bajada en una Intranet inalámbrica. Por este motivo, el uso de un túnel DNS puede ser interesante cuando queremos gestionar un servidor remoto mediante ssh, o queremos consultar un sitio Web con poca carga de contenidos, pero no en casos donde se requiera un importante ancho de banda.

Por esta razón, al final de la práctica se sugiere medir la velocidad de la conexión (*p.e. mediante speedtest*), y así advertir que tipo de conexiones abrir a través del túnel DNS.

En concreto para su implementación seguiremos los siguientes pasos, empezando en primer lugar por la configuración el servidor, y después con la del cliente:

Paso Nº1.- Comenzaremos instalando el software servidor **iodine** en el servidor e iniciando su servicio **iodined** (*demonio o servicio asociado a iodine*). Señalar que las opciones de inicialización del servicio **iodined** son muy variadas (*man iodine*), pero destacaremos las siguientes:

iodined [-f] [-P password] *dirección_IP*[/máscara] *nombre_dominio*

- **-f**, parámetro opcional que provoca que el servicio **iodined** se ponga a la escucha en modo foreground, evitando de esta forma que lo haga en background, lo cual nos permitirá desactivarlo en cualquier momento tecleando un CTRL+C.
- **-P**, parámetro opcional que nos permitirá indicar una password que será necesaria para autenticarse. En caso de no indicarla se solicitará al ejecutar el comando **iodined**.
- **dirección_IP**[/máscara], parámetro obligatorio que informará al servicio **iodined** de cual será la dirección IP que recibirá el extremo servidor del túnel DNS. En concreto, **iodined** creará una interfaz de red virtual en el equipo servidor, p.e. **dns0**, la cual recibirá la dirección IP indicada, modificando en consecuencia su tabla de enrutamiento, y que usará para comunicarse con el otro extremo del túnel (*equipo cliente*). Por tanto, la elección de la

dirección IP establecerá el rango de direcciones IP privadas que se usarán en la comunicación entre cliente y servidor, motivo por el cual es obligatorio que dicho rango de direcciones IP no se este utilizando simultáneamente en las redes privadas o Intranet en las cuales se encuentran cliente y servidor, ya que en ese caso la tabla de enrutamiento ocasionaría una ambigüedad al sistema al tomar la decisión de como enrutar los paquetes hacia el destino. En el caso práctico que aquí se resuelve se usa el rango de

- **nombre_dominio**, parámetro obligatorio que establecerá bajo que resolución de nombres de dominio se encapsulará el trafico que genere el cliente hacia el servidor.

```
[root@servidor]# apt-get install iodine
[root@servidor]# iodined -f -P mipass 10.0.0.1/24 test.prub
```

Una vez iniciado el servicio podremos comprobar el aspecto que tiene su nueva tabla de enrutamiento abriendo una nueva terminal (*iodined se inicia en modo foreground dejando inoperativa la terminal*). En concreto, deberemos fijarnos en que se ha creado una nueva interfaz de red virtual, p.e. dns0, a través de la cual se enrutará todo el trafico dirigido hacia equipos de la red 10.0.0.0/24 (*se supone que la dirección de red de la Intranet donde se encuentra el servidor iodined es la 192.168.1.0/24, y que su interfaz de red física es inalámbrica, wlan0*).

```
[root@servidor]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG    0      wlan0
10.0.0.0        0.0.0.0         255.255.255.0   U     0      dns0
192.168.1.0     0.0.0.0         255.255.255.0   U     0      wlan0
```

Por último, para terminar de configurar el servidor **iodined**, deberemos activar el ip_forward y el enmascaramiento de los paquetes (*repasar prácticas del firewall*), con la finalidad de que el servidor pueda hacer las veces de puerta de enlace para los equipos clientes que establezcan el túnel DNS.

```
[root@servidor]# echo 1 > /proc/sys/net/ipv4/ip_forward
[root@servidor]# iptables -t nat -A POSTROUTING -j MASQUERADE
```

¡¡Observación Importante!! Para comprobar que el servicio **iodined** está a la escucha, puede hacerse una exploración de puertos mediante los comandos **netstat** o **nmap** (*nmapfe o zenmap son herramientas basadas en nmap con entorno gráfico*).

En el caso de usar **netstat**, es conveniente usar los siguientes parámetros para no obtener demasiada información: (1) **-l**, para filtrar los servicios que están a la escucha (*listening*), (2) **-u**, para indicar que en el resultado del escáner solo muestre información sobre los servicios UDP (*el servicio DNS al que suplanta iodined trabaja por defecto bajo udp/53*), (3) **-p**, para poder identificar el PID del proceso que da dicho servicio, y (4) **-n**, para informar mediante direcciones IP en lugar de nombres de dominio asociados. No obstante, también podemos filtrar al máximo el resultado apoyándonos en el comando **grep**.

```
[root@servidor]# netstat -l -u -p -n
```

```
[root@servidor]# netstat -lupn | grep ":53 "
[root@servidor]# netstat -lupn | grep iodined
udp        0      0 0.0.0.0:53              0.0.0.0:*                           32502/iodined
```

En el caso de usar **nmap**, sabiendo que por defecto **iodined** ofrece el servicio vía UDP por el puerto 53 (*suplanta a un servidor DNS*), le diremos que analice ese puerto directamente para evitar tiempo de computo innecesario. Las opciones **-sU** y **-p** nos permiten indicarle a **nmap** que únicamente analice protocolos UDP en el rango de puertos que le especifiquemos:

```
[root@servidor|cliente|equipo_red]# nmap -sU -p 53 dirección_IP_Servidor_iodined
PORT      STATE          SERVICE
53/udp    open|filtered  domain
```

Según lo anterior deberemos tener en cuenta que en el equipo servidor donde iniciemos **iodined** no debe estar iniciado simultáneamente ningún otro software servidor que de servicio por el mismo puerto **udp/53** (*algún servicio DNS*), como podría ser **bind9**. En caso contrario, se produciría un conflicto entre ambos servicios, no pudiendo establecerse el túnel.

Paso Nº2.- Tras configurar el servidor, continuaremos instalando **iodine** en el equipo cliente para posteriormente establecer el túnel DNS de comunicación. Respecto a los parámetros que podrían usarse en la aplicación cliente **iodine**, podríamos destacar los siguientes:

iodine [-f] [-P password] [dirección_IP_Servidor] *nombre_dominio*

- **-f**, parámetro opcional, que al igual que en el servidor, provoca que **iodine** establezca el túnel de comunicación con el servidor en modo foreground, evitando de esta forma que lo haga en background, lo cual nos permitirá desactivarlo en cualquier momento tecleando un CTRL+C.
- **-P**, parámetro opcional que nos permitirá indicar una password que será necesaria en la autenticación contra el servidor. Obviamente indicaremos la que se asigno en el servidor al iniciar el servicio **iodined**. En caso de no indicarla se solicitará al ejecutar el comando **iodine**.
- **dirección_IP_Servidor**, parámetro opcional que informará a **iodine** de cual será la dirección IP o nombre de dominio del servidor DNS que será usado para salir a Internet y conocer de esta forma la dirección IP asociada al servidor **iodined**. En concreto, para que la solicitud quede totalmente camuflada, debería adquirirse un nombre de dominio de segundo nivel (*2LD, second Level Domain*) con gestión propia, p.e. amromero.es, de tal forma que podamos asociar a su **registro NS** (*NameServer*) la dirección IP del servicio **iodined**. No obstante, si lo único que pretendemos de momento es realizar pruebas de funcionamiento, podremos poner directamente la dirección IP del servidor **iodined**, que es lo que se ha hecho en la resolución de la práctica.
- **nombre_dominio**, relacionado con lo indicado en el punto anterior, este parámetro obligatorio indicará el nombre del dominio que cuya resolución nos llevará a conocer la dirección IP del servidor **iodined**. Éste deberá estar en consonancia con el dominio indicado al iniciar el servidor **iodined**. En el caso práctico que aquí se muestra, al indicar como dirección IP del servidor, la del servidor **iodined**, el nombre de dominio puede ser ficticio (*p.e. test.prub*).

```
[root@cliente]# apt-get install iodine
[root@cliente]# iodine -f -P mipass dirección_IP_Servidor_iodined test.prub
...
Setting IP of dns0 to 10.0.0.2
Setting MTU of dns0 to 1130
Server tunnel IP is 10.0.0.1
Testing raw UDP data to the server (skip with -r)
Server is at dirección_ip_Servidor_iodined, trying raw login: OK
Sending raw traffic directly to dirección_ip_Servidor_iodined
Connection setup complete, transmitting data.
```

Tras lanzar el comando anterior y establecerse exitosamente el túnel DNS, debería haberse creado en el cliente, al igual que ocurrió en el servidor, una interfaz de red virtual **dns0** con una dirección IP del mismo rango reservado por el servidor **iodined 10.0.0.0/24**. En concreto, si es el primer cliente que se conecta recibirá la dirección IP siguiente en el rango a la que se asigno al servidor **10.0.0.2**. Es decir, que el servidor **iodined** actúa como un servidor DHCP asignando a los clientes una dirección IP del rango reservado.

Por lo tanto, si visualizamos la tabla de enrutamiento del equipo cliente deberíamos ver algo parecido a lo siguiente (*se supone que el equipo cliente se encuentra en una Intranet con dirección de red 192.168.123.0/24, y que su interfaz de red física es inalámbrica, wlan0*):

```
[root@cliente]# route -n
```

Destino	Pasarela	Genmask	Indic	Métric	Interfaz
0.0.0.0	192.168.123.254	0.0.0.0	UG	0	wlan0
10.0.0.0	0.0.0.0	255.255.255.0	U	0	**dns0**
192.168.123.0	0.0.0.0	255.255.255.0	U	9	wlan0

Por último, para aprovecharnos del túnel DNS creado, mediante el uso de reglas de enrutamiento estáticas redireccionaremos el trafico de Internet hacia el servidor iodined, sin olvidarnos de que para mantener el túnel DNS activo, deberemos introducir las reglas correspondientes de acceso al servidor a través de la puerta por defecto de la Intranet (*suponiendo que el servidor DNS preferente configurado en el cliente es el 8.8.8.8*):

```
[root@cliente]# route add -host 8.8.8.8 gw 192.168.123.254
[root@cliente]# route add -host dirección_ip_Servidor_iodined gw 192.168.123.254
[root@cliente]# route del default gw 192.168.123.254
[root@cliente]# route add default gw 10.0.0.1
```

De esta forma la tabla de enrutamiento resultante debería quedar así:

```
[root@cliente]# route -n
```

Destino	Pasarela	Genmask	Indic	Métric	Interfaz
192.168.123.0	0.0.0.0	255.255.255.0	U	9	wlan0
ip_Servidor_iodined	192.168.123.254	255.255.255.255	**UH**	0	wlan0
8.8.8.8	192.168.123.254	255.255.255.255	**UH**	0	wlan0
10.0.0.0	0.0.0.0	255.255.255.0	U	0	**dns0**

```
0.0.0.0          10.0.0.1        0.0.0.0         UG    0      dns0
```

¡¡Observación!! Al indicarse en el comando iodine directamente la dirección IP del servidor iodined, en lugar de la dirección del servidor DNS permitido en la Intranet, es posible que en un entorno real no se pueda crear el túnel DNS, en función de como haya sido filtrado el servicio udp/53. Por ejemplo, si las consultas hacía el servicio DNS dentro de una Intranet protegida con portal cautivo han sido filtradas han sido filtradas y se permite únicamente consultar por ejemplo al servidor DNS 8.8.8.8 (*repasar el capitulo de prácticas asociado al firewall iptables*), no será posible establecer la conexión de la forma anterior.

```
[root@firewall_Intranet] iptables -t filter -A FORWARD -p udp --dport 53 -d 8.8.8.8 -j ACCEPT
[root@firewall_Intranet] iptables -t filter -A FORWARD -p udp --dport 53 -j DROP
[root@firewall_Intranet] iptables -t filter -A FORWARD -p udp --dport 53 \
-m state --state ESTABLISHED,RELATED -j ACCEPT
```

En ese caso, al no poder hacer consultas DNS directamente tal como se ha indicado anteriormente será necesario registrar un nombre de dominio público, p.e. amromero.es, con la posibilidad de gestión de los registros DNS, e indicar que el servidor NS (*NameServer*) que el equipo encargado de resolver nombres asociados a dicho dominio es el servidor **iodined**. Es decir, podría ser algo así:

```
amromero.es.          IN    NS    iodined.amromero.es.
iodined.amromero.es.  IN    A     dirección_IP_pública_Servidor_iodined
```

Para terminar con la práctica, sería conveniente hacer un test de velocidad y así comprobar el ancho de banda que nos ofrece un túnel DNS, de tal forma que seamos conscientes de que tipo de aplicaciones podemos tunelizar: conexiones SSH, tráfico HTTP/HTTPS con pequeña carga en sus contenidos (texto, imagenes, etc.), transferencia de archivos de reducido tamaño, etc. Para la medición podemos descargarnos la aplicación python **speedtest**:

```
[usuario@cliente]$ wget https://raw.github.com/sivel/speedtest-cli/master/speedtest_cli.py
[usuario@cliente]$ chmod +x speedtest_cli.py
[usuario@cliente]$ ./speedtest_cli.py
Retrieving speedtest.net configuration...
Retrieving speedtest.net server list...
Testing from Orange Espana (dirección_IP_pública_asociada_Servidor_iodined)...
Selecting best server based on ping...
Hosted by iA Soft Aragon (Zaragoza) [0.54 km]: 29.643 ms
Testing download speed.....................................
Download: 0.08 Mbits/s
Testing upload speed.....................................................
Upload: 0.10 Mbits/s
```

Ej. Práctico 3.4.4: Proxy Transparente y No Transparente - Cron

Para terminar con el servicio Proxy HTTP Caché **squid**, a continuación se propone una última configuración donde mediante la ayuda del servicio **cron** (*programación de tareas automáticas*), dependiendo de la franja horaria en la que nos encontremos, nuestro Proxy se comportará de una manera u otra según las especificaciones de autenticación y filtrado de contenidos que se especifican en la siguiente tabla. En relación al valor que adoptarán el resto de parámetros generales de configuración del Proxy, **visible_hostname, http_port, cache_mem, cache_dir ufs, maximum_object_size_in_memory, access_log, cache_log, cache_store_log**, serán los ya establecidos a modo de ejemplo en los ejercicios anteriores, aunque pueden modificarse según se desee:

Franja Horaria *lunes a viernes*	Tipo Proxy Autenticación y Realm	Usuarios Permitidos	URLs Prohibidas
8:00 – 14:00	No Transparente Digest - Zona_Restringida	empleado1, empleado2	sport, facebook, twitter, sex, porn
		adm1, adm2	Ninguna
14:00 – 16:00	Proxy Transparente con Portal Cautivo	Todos (*adm1*, *adm2*, ...)	sex, porn
16:00 – 22:00	No Transparente Digest - Zona_Restringida	empleado3, empleado4	sport, facebook, twitter, sex, porn
		adm3	sex, porn

Solución Ej. Pr. 3.4.4.I.- *Configuración del Proxy mediante Cron*

La solución que se muestra a continuación consta de las siguientes partes:

(0) Configuración previa del entorno de red y el firewall del servidor Proxy (*p.e. script* ***conf-red***).
(1) Creación de los archivos de configuración del servicio **squid** para su comportamiento como transparente y no transparente, según las especificaciones indicadas en la tabla anterior: **squid-transparente.conf** y **squid-no-transparente.conf**
(2) Creación del fichero plano **usuarios.digest** de cuentas de usuario **digest** mediante **htdigest**.
(3) Creación de los ficheros de texto que contendrán las palabras clave con las que serán filtradas las URLs: **palabras-clave.filtro1** (*sport, facebook, twitter, sex, porn*) y **palabres-clave.filtro2** (*sex, porn*).
(4) Configuración del servicio **cron** para que que cargue una u otra configuración en función de la franja horaria en la que nos encontremos. Para ello, tan sólo será necesario reflejar en el fichero general de configuración del Proxy **squid.conf** uno de los dos archivos anteriores, **squid-transparente.conf** o **squid-no-transparente.conf**, a las horas establecidas en la tabla.
(5) Configuración de **apache2** para dar servicio **tcp/80** y **tcp/443** mediante sus sitios Web por defecto (*consultar el ejercicio 3.1.6*).
(6) Creación de las páginas de inicio de los sitios Web servidos por apache2 del portal cautivo: **index.html** (*consultar el ejercicio 3.1.6*) e **index.php**.

(7) Concesión de privilegios mediante **sudo** a apache2 para ejecutar comandos del sistema, **iptables**, a través de su cuenta de usuario del sistema **www-data**.
(8) Creación del script que dará de baja aquellas máquinas que estén inactivas, y que se les haya dado acceso previamente a través del portal cautivo (*consultar el script* **dar-de-baja-desconectados.sh** *del ejercicio 3.1.6*).

A continuación se detallarán cada uno de los pasos anteriores:

(0) Al igual que hacíamos en la resolución de los ejercicios prácticos del capítulo de diseño de firewalls, crearemos un **script** que al iniciarse el sistema, o cuando nosotros requiramos, será ejecutado, y se encargará de configurar tanto la red como las reglas del firewall:

```
[root@mv1]# nano /etc/init.d/conf-red
#!/bin/bash
/etc/init.d/networking stop
ifconfig eth0 down
ifconfig eth0 192.168.1.100
route add default gw 192.168.1.254
ifconfig eth0:alias1 192.168.2.100
# Si el equipo servidor lo tenemos configurado como Servidor DNS Caché (si no 8.8.8.8):
echo "nameserver 127.0.0.1" > /etc/resolv.conf
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -t nat -F
iptables -t nat -A POSTROUTING -j MASQUERADE
iptables -t nat -A PREROUTING -s 192.168.2.0/24 -p tcp --dport 80 \
-j DNAT --to 192.168.2.100:80
iptables -t nat -A PREROUTING -s 192.168.2.0/24 -p tcp --dport 443 \
-j DNAT --to 192.168.2.100:443
iptables -t filter -F
iptables -P FORWARD DROP
iptables -A FORWARD -p udp --dport 53 -j ACCEPT
iptables -A FORWARD -p icmp -s 192.168.2.0/24 -j ACCEPT
iptables -A FORWARD -p tcp --dport 3128 -s 192.168.2.0/24 -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

```
[root@mv1]# chmod +x /etc/init.d/conf-red
[root@mv1]# echo "bash /etc/init.d/conf-red" >> /etc/init.d/rc.local
[root@mv1]# bash /etc/init.d/conf-red
```

(1) Ficheros de configuración **squid-no-transparente.conf** y **squid-transparente.conf**:

```
[root@mv1]# nano /etc/squid3/squid-no-transparente.conf
# Directivas globales
visible_hostname miproxy
http_port 3128

cache_mem 32 MB
```

```
cache_dir ufs /var/spool/squid3 1000 16 256
maximum_object_size_in_memory 128 MB

access_log /var/log/squid3/access.log
cache_log /var/log/squid3/cache.log
cache_store_log /var/log/squid3/store.log

auth_param digest program /usr/lib/squid3/digest_pw_auth -c /etc/squid3/usuarios.digest
auth_param digest children 5
auth_param digest realm zona_restringida

acl empleados_manana proxy_auth -i empleado1 empleado2
acl empleados_tarde proxy_auth -i empleado3 empleado4
acl administradores_manana proxy_auth -i adm1 adm2
acl administradores_tarde proxy_auth -i adm3
acl horario_manana time M T W H F 8:00-14:00
acl horario_tarde time M T W H F 16:00-22:00
acl filtro1 url_regex "/etc/squid3/palabras-clave.filtro1"
acl filtro2 url_regex "/etc/squid3/palabras-clave.filtro2"
acl cualquier-maquina src all

http_access allow cualquier-maquina administradores_manana horario_manana
http_access allow cualquier-maquina administradores_tarde horario_tarde !filtro2
http_access allow cualquier-maquina empleados_manana horario_manana !filtro1
http_access allow cualquier-maquina empleados_tarde horario_tarde !filtro1
http_access deny cualquier-maquina
```

```
[root@mv1]# nano /etc/squid3/squid-transparente.conf
# Directivas globales
visible_hostname miproxy
http_port 3128 transparent

cache_mem 32 MB
cache_dir ufs /var/spool/squid3 1000 16 256
maximum_object_size_in_memory 128 MB

access_log /var/log/squid3/access.log
cache_log /var/log/squid3/cache.log
cache_store_log /var/log/squid3/store.log

acl filtro2 url_regex "/etc/squid3/palabras-clave.filtro2"
acl cualquier-maquina src all

http_access allow cualquier-maquina !filtro2
```

(2) Continuaremos con la creación del fichero plano que contendrá las cuentas de usuario para la autenticación del Proxy no transparente **/etc/squid3/ usuarios.digest**:

```
[root@mv1]# htdigest -c /etc/squid3/usuarios.digest zona_restringida  empleado1
[root@mv1]# htdigest /etc/squid3/usuarios.digest zona_restringida  empleado2
[root@mv1]# htdigest /etc/squid3/usuarios.digest zona_restringida  empleado3
[root@mv1]# htdigest /etc/squid3/usuarios.digest zona_restringida  empleado4
[root@mv1]# htdigest /etc/squid3/usuarios.digest zona_restringida  adm1
[root@mv1]# htdigest /etc/squid3/usuarios.digest zona_restringida  adm2
[root@mv1]# htdigest /etc/squid3/usuarios.digest zona_restringida  adm3
```

(3) Pasaremos a crear los ficheros de texto que contendrán las palabras clave con las que serán filtradas las URLs: **palabras-clave.filtro1** (*sport, facebook, twitter, sex, porn*) y **palabres-clave.filtro2** (*sex, porn*). Para ello simplemente crearemos ambos ficheros de texto, donde en cada línea aprezca una de las palabras clave a filtrar:

```
[root@mv1]# nano /etc/squid3/palabras-clave.filtro1
sport
facebook
twitter
sex
porn
[root@mv1]# nano /etc/squid3/palabras-clave.filtro2
sex
porn
```

(4) Por último el el usuario **root** administrador del equipo servidor **mv1** configurará el servicio **cron** para que que cargue una u otra configuración en función de la franja horaria en la que nos encontremos. En concreto, como se podrá advertir a continuación, lo único que se hace es copiar en el archivo de configuración del Proxy **squid.conf** uno de los dos archivos de configuración anteriores, **squid-transparente.conf** o **squid-no-transparente.conf**, y reiniciar el servicio **squid** a las horas establecidas en la tabla del enunciado.

```
[root@mv1]# crontab -e
#Añadimos las siguientes líneas
0 8 * * * cp /etc/squid3/squid-no-transparente.conf /etc/squid3/squid.conf && /etc/init.d/squid3 restart
0 14 * * * cp /etc/squid3/squid-transparente.conf /etc/squid3/squid.conf && /etc/init.d/squid3 restart
0 16 * * * cp /etc/squid3/squid-no-transparente.conf /etc/squid3/squid.conf && /etc/init.d/squid3 restart
0 22 * * * /etc/init.d/squid3 stop
```

(5) Configuraremos **apache2** para dar servicio **tcp/80** y **tcp/443** mediante sus sitios Web por defecto (*consultar el ejercicio 3.1.6*).

(6) Crearemos de las páginas de inicio de los sitios Web servidos por apache2 del portal cautivo: **index.html** (*consultar el ejercicio 3.1.6*) e **index.php**. El index.html servido por apache2 vía tcp/80 tan sólo contendrá un enlace al index.php del servicio tcp/443 de apache2, con la finalidad de que los datos introducidos por los usuarios a través del portal cautivo, login y password, viajen por un canal seguro (*HTTPS*).

En relación al **index.php**, basándonos en el ya mostrado en el ejercicio 3.1.6, su aspecto

podría ser el siguiente (*la autenticación se realiza a través del mismo fichero de usuarios que se creo mediante htdigest para el Proxy no transparente*):

¡¡Aclaración!! El comando PHP **exec()** que nos permite ejecutar comandos del sistema, tiene el siguiente formato:

exec($comando [, $salida, [$estado_salida]]);

- El primer argumento, **$comando**, hace referencia al comando del sistema que deseamos ejecutar.
- El segundo argumento, **$salida** (*opcional*), es un array o vector donde en cada una de sus posiciones se almacenan las diferentes líneas que se producen como respuesta. Por ejemplo, $salida[0] se correspondería con la primera línea de los resultados mostrados por consola al ejecutar el comando indicado.
- El tercer argumento, **$estado_salida** (*opcional*), nos informa de si tuvo éxito o no la ejecución del comando indicado. En concreto, adopta el valor "**0**" en caso de éxito, y un valor distinto de cero, como código de error.

```
[root@mv1]# nano /var/www/index.php
<meta http-equiv="Content-Type" content="text/html;charset=utf-8">
<style type="text/css">
div.div1 { width: 100%; margin-top: 40px; text-align: center; }
div.div2 { width: 100%; margin-top: 10px; text-align: center; }
fieldset.f1 { width: 50%; margin-left: auto; margin-right: auto;
background-color: yellow; text-align: center; }
fieldset.f2 { width: 50%; margin-left: auto; margin-right: auto;
background-color: black; color: white; text-align: center; }
fieldset.f3 { width: 50%; margin-left: auto; margin-right: auto;
background-color: pink; color: brown; font-weight: bold; text-align: justify; padding: 10px }
</style>
<div class="div1">
<fieldset class="f1">
<legend>Control de Acceso - Portal Cautivo</legend>
<form name="form1" method="post" action="">
Nombre: <input type="text" name="nombre" size="20"><br><br>
Password: <input type="password" name="pass" size="20"> <br><br>
<input type="submit" name="boton" value="Enviar">
</form>
</fieldset>
</div>
<?
if ( isset($_POST['boton']) )
{
// Mediante el comando del sistema "date" podemos conocer la hora del sistema:
// El día de la semana devuelto por %u es: 1 - lunes, 2 - martes ...
$comando_dia_semana = "date +%u";
// La hora del día devuelto por %H es formato 24h: 00 .. 23
$comando_hora = "date +%H";
```

```
exec($comando_dia_semana,$dia);
exec($comando_hora,$hora);
if ( $dia[0] != 6 && $dia[0] != 7 && $hora[0] >= 14 && $hora[0] <= 16 )
{
        $ruta_digest = "/etc/squid3/usuarios.digest";
        $fichero = fopen ($ruta_digest,"r");
        $verificado = 0;
        // Mediante un bucle comprobamos si los datos, login y password, son correctos
        while ( preg_match("/:/", $linea = fgets($fichero) ) )
        {
                $linea = rtrim ($linea);
                $campos = explode (':',$linea);
                $realm = "zona_restringida";
                $posible_pass = md5($_POST['nombre'].":".$realm.":".$_POST['pass']);
                if ( $_POST['nombre'] == $campos[0] && $posible_pass == $campos[2] )
                {
                        $verificado = 1;
                }
        }
        if ( $verificado == 1 )
        {
                $ip = $_SERVER['REMOTE_ADDR'];
                $comando1 = "sudo iptables -t nat -I PREROUTING 1 -s ".
                                $ip." -p tcp --dport 80 -j REDIRECT --to-port 3128";
                $comando2 = "sudo iptables -t nat -I PREROUTING 1 -s ".
                                $ip." -p tcp --dport 443 -j ACCEPT";
                $comando3 = "sudo iptables -I FORWARD 1 -s ".
                                $ip." -p tcp --dport 443 -j ACCEPT" ;
                exec($comando1,$salida1,$estado_salida1);
                exec($comando2,$salida2,$estado_salida2);
                exec($comando3,$salida3,$estado_salida3);
                if ( ( $estado_salida1 == 0 ) && ( $estado_salida2 == 0 ) &&
                        ($estado_salida3 == 0) )
                {
                        header('Location: http://www.google.es/nwshp');
                }
                else
                {
                        header('Location: https://192.168.2.100/error.php');
                }
        }
        else
        {
                ?>
                <br>
                <div class="div1"><fieldset class="f2">
                <legend style="color: red;">Respuesta ante la autenticación</legend>
```

```
                Lo sentimos ... Tu login o password no son correctos!!!<br>
                Tu máquina con ip <? echo $ip; ?> no tendrá acceso
                a Internet mientras no te autentiques correctamente.
                </fieldset></div>
                <?
        }
}
else
{
?>
        <br>
        <div class="div2"><fieldset class="f3">
        <legend style="color: red;">Respuesta ante la autenticación</legend>
        Lo sentimos ... Estas fuera del horario permitido para el uso del
        Proxy Transparente (lunes-viernes, 14h-16h).
        Para poder navegar a través de Internet desde tu máquina con ip
        <? echo $_SERVER['REMOTE_ADDR']; ?> deberás configurar el Proxy:
        <p style="text-align: center">
        ip proxy: 192.168.2.100<br> puerto proxy: 3128
        </p>
        </fieldset></div>
<?
}
unset($_POST['boton']);
} ?>
```

¡¡Aclaración!! Repasando el código PHP anterior podrá advertirse, que con la finalidad de cumplir la restricción horaria indicada en el enunciado del ejercicio práctico, referente a que los usuarios tan sólo deben poder acceder a través del portal cautivo a Internet y navegar a través del Proxy transparente en la **franja horaria de 14h a 16h de lunes a viernes**, se ha hecho uso del comando del sistema **date** (*man date*). Este comando nos permite conocer de una manera muy sencilla el día y la hora del servidor sin necesidad de recurrir a funciones PHP, de tal forma que mediante una estructura condicional **if**, condicionamos la validación de la autenticación del usuario y el permiso de acceso a navegar:

```
// Mediante el comando del sistema "date" podemos conocer la hora del sistema:
// El día de la semana devuelto por %u es: 1 - lunes, 2 - martes ...
$comando_dia_semana = "date +%u";
// La hora del día devuelto por %H es formato 24h: 00 .. 23
$comando_hora = "date +%H";
exec($comando_dia_semana,$dia);
exec($comando_hora,$hora);
if ( $dia[0] != 6 && $dia[0] != 7 && $hora[0] >= 14 && $hora[0] <= 16 )
{
        Instrucciones PHP;
}
```

(7) Concesión de privilegios mediante **sudo** a apache2 para ejecutar comandos del sistema a través de su cuenta de usuario del sistema **www-data**. Esto permitirá que sean ejecutados de manera exitosa los comandos **iptables** indicados en el **index.php**.

```
[root@mv1]# visudo
# Añadimos la siguiente línea:
www-data ALL=(ALL:ALL) NOPASSWD: /sbin/iptables
```

(8) Creación del script que dará de baja aquellas máquinas que estén inactivas, y que se les haya dado acceso previamente a través del portal cautivo (*consultar el script* **dar-de-baja-desconectados.sh** *del ejercicio 3.1.6*).

Ahora tan sólo queda comprobar su correcto funcionamiento. Mediante el comando date, podemos modificar la hora del sistema con la finalidad de comprobar el funcionamiento del Proxy como transparente o no transparente dependiendo de la franja horaria (*man date*):

```
[root@mv1]# date --set "2013-12-8 15:00:00"
```

Ej. Práctico 3.4.5: Monitorización del Estado del Proxy

El último ejercicio práctico que se propone en relación al Proxy squid consiste en instalar en el equipo servidor Proxy un sistema de monitorización vía Web que permita consultar los accesos que se han realizado por parte de los clientes. Para ello, se presentan a continuación dos posibilidades muy similares: **sarg** y **lightsquid**.

Solución Ej. Pr. 3.4.5.I.- Monitorización del Proxy Squid: sarg o lightsquid

Al igual que al final del capítulo anterior asociado a los firewall y la seguridad perimetral se propuso un ejercicio práctico relacionado con la monitorización del tráfico de nuestra red, en este capítulo se propone monitorizar los accesos realizados por los equipos clientes a los cuales les da servicio el Proxy. Esta monitorización nos facilitará la lectura y comprensión del archivo de auditoría **access.log**, lo cual es **sumamente útil sobre todo cuando el Proxy esta configurado como no transparente**, ya que al tener que autenticarse previamente en ese tipo de configuración, a través de la auditoría se advierte perfectamente cada usuario por que sitios navego, cuantos datos se descargo, etc.

(1ª opción) Instalar **sarg**. Esta es la opción más recomendada por la sencillez en su instalación y post-configuración. En concreto, se trata de un servicio de generación de informes en base a la información encontrada en **/var/log/squid3/access.log**, cuyos resultados son mostrados vía Web mediante tablas HTML, y por tanto, de manera dependiente del servidor Web que haya instalado en el equipo Proxy (*p.e. apache2*). Por tanto, para ofrecer este servicio de monitorización será necesario instalar por ejemplo **apache2** y **sarg**.

```
[root@mv1]# apt-get install apache2 sarg
```

A continuación configuraremos **apache2** para dar servicio al sitio Web de **sarg** ubicado en **/var/lib/sarg** (*su ubicación y otros muchos parámetros pueden modificarse editando el archivo de configuración /etc/sarg/sarg.conf*) que contiene los datos extraídos del archivo de autoría **access.log** del servicio **squid**. Para ello, editaremos un nuevo archivo de configuración en Apache, para que a través del alias **/sarg** pueda accederse de una manera sencilla a los contenidos de ese directorio:

http://dirección_ip_Proxy/**sarg**

¡¡Importante!! Comprueba la ruta del archivo **access.log** de tu servicio Squid (*en el archivo squid.conf debiste indicar donde se ubica*) dentro del archivo de configuración del Sarg. Por ejemplo, si tienes instalado el **squid3** (*apt-get install squid3*), en lugar de squid, deberás modificar la ruta donde se encuentra el archivo de auditoría **access.log** a través de la directiva de configuración **access_log** en el fichero de configuración de **sarg**, **/etc/sarg/sarg.conf**:

```
[root@mv1]# nano /etc/sarg/sarg.conf
...
access_log /var/log/squid3/access.log
```

```
[root@mv1]# cd /etc/apache2
```

```
[root@mv1:/etc/apache2]# nano conf.d/sarg
Alias /sarg /var/lib/sarg
<Directory /var/lib/sarg>
        Allow from all
        DirectoryIndex index.html
</Directory>
```

Después tan sólo será necesario reiniciar el servicio de Apache, y acceder desde un cliente colocando en la barra de direcciones de su navegador: http://*dirección_ip_Proxy*/**sarg**, por ejemplo, **http://192.168.2.100/sarg**. Es posible que la primera vez que accedamos el sitio Web de Sarg éste esté vacío, ya que Sarg genera los informes una vez al día. Si queremos forzar a generar los informes, y así poder visualizar un informe de accesos a través de Squid mediante Sarg, deberemos ejecutar el comando "**sarg-reports**":

```
[root@mv1]# sarg-reports [ manual DD/MM/YYYY | today | daily | weekly | monthly ]
[root@mv1]# sarg-reports manual 20/12/2013
[root@mv1]# sarg-reports daily
[root@mv1]# sarg-reports mounthly
```

¡¡Aclaración!! El servicio **sarg** genera informes diariamente. Para ello, durante su instalación se configuro el servicio **cron** encargado de la automatización de tareas, para que diariamente, sin la intervención de ningún usuario, se generen los informes en función de la información encontrada en el archivo **access.log** *(comprueba en /etc/sarg/sarg.conf que la ruta del access.log es la indicada en squid.conf)*. En caso de no querer esperar a que sea **cron** quien los genere, pueden generarse manualmente ejecutando **sarg-reports daily** . De igual forma, para modificar su periodicidad sería conveniente cambiar la configuración del **cron**.

```
[root@mv1]# more /etc/cron.daily/sarg
#!/bin/sh
if [ -x /usr/sbin/sarg-reports ]; then
  /usr/sbin/sarg-reports daily
fi
```

¡¡Advertencia!! Una vez instalado y configurado Sarg podrá apreciarse que hipotéticamente es capaz de suministrarnos tanto informes diarios del trafico Web, como semanales y mensuales. Digo "hipotéticamente" ya que para poder generar dichos informes necesitaremos que Squid preserve los ficheros de auditoría **access.log** semanalmente o mensualmente, en el caso de querer generar informes con esa periodicidad, lo cual por defecto no es posible. Es decir, si consultamos el archivo de configuración referente a las rotaciones de los ficheros de auditoría de Squid (*/etc/logrotate.d*), veremos que por defecto estos rotan diariamente, **daily** (*access.log, access.log.1, access.log.2.gz*), con la finalidad de que los ficheros no se vuelvan muy voluminosos, perdiéndose su contenido al cabo de dos rotaciones, **rotate 2**, es decir, dos días.

```
[root@mv1]# more /etc/logrotate.d/squid3
/var/log/squid3/*.log {
        daily
        compress
        delaycompress
        rotate 2
        missingok
        nocreate
        sharedscripts
        postrotate
                test ! -e /var/run/squid3.pid || /usr/sbin/squid3 -k rotate
        endscript
}
```

Es decir, que si queremos generar informes semanales o mensuales habrá que modificar el parámetro **daily** por **weekly** o **monthly**.

(2ª opción) Instalar **lightsquid**. Al igual que **sarg**, se trata de otra herramienta software de

generación de informes a partir del archivo de auditoría de **squid access.log**, que luego nos permite visualizarlos vía Web desde un navegador Web. De igual forma que con **sarg**, habrá de instalar un software servidor Web (*p.e. apache2*) para dar servicio al sitio Web que contiene los datos obtenidos.

```
[root@mv1]# apt-get install apache2 lightsquid
```

Tras instalar **lightsquid** es conveniente echarle un ojo a su archivo de configuración **lightsquid.cfg** para comprobar sus valores por defecto, idioma, etc.

```
[root@mv1]# nano /etc/lightsquid/lightsquid.cfg
...
$lang             ="sp";
$decdelimiter       = ".";
...
```

Para chequear su configuración ejecutaremos el comando **check-setup.pl**:

```
[root@mv1]# /usr/share/lightsquid/check-setup.pl
```

Por último, configuraremos nuestro servidor Web. En el caso de utilizar apache2, durante la instalación de **lightsquid** se autogeneró un archivo de configuración que tendremos que repasar:

```
[root@mv1]# nano /etc/apache2/conf.d/lightsquid
Alias  /lightsquid/   /usr/lib/cgi-bin/lightsquid/
<Directory /usr/lib/cgi-bin/lightsquid/>
        DirectoryIndex index.cgi
        AddHandler cgi-script .cgi
</Directory>
<Location "/lightsquid/">
        # Para garantizar su consulta desde cualquier equipo cliente:
        Allow from all
</Location>
```

Ahora ya podremos acceder a la monitorización Web ofrecida por lightsquid desde cualquier equipo cliente de nuestra red, escribiendo en la barra de direcciones de su navegador la siguiente URL: http://*dirección_ip_Proxy*/**lightsquid/index.cgi**

Informe de accesos de usuarios de Squid
PerÃodo comprendido: Nov 2013

Calendar											
2013											
01	02	03	04	05	06	07	08	09	10	11	12

Fecha	Grupo	Usuarios	Excedidos	Bytes	Promedio	Hit %
22 Nov 2013	grp	8	1	52.1 M	6.5 M	4.38%
21 Nov 2013	grp	2	0	597.444	298.722	79.27%
19 Nov 2013	grp	2	0	885.147	442.573	77.01%
18 Nov 2013	grp	3	0	878.812	292.937	2.07%
15 Nov 2013	grp	3	0	5.1 M	1.7 M	20.53%
13 Nov 2013	grp	2	0	2.5 M	1.3 M	30.85%
Total/Promedio:		3	0	62.0 M	1.7 M	35.68%

LightSquid v1.8 (c) Sergey Erokhin AKA ESI.

¡¡Aclaración!! Al igual que con **sarg**, los informes en **lightsquid** son generados por el script **/usr/share/lightsquid/lightparser.pl** automáticamente gracias al servicio **cron**. Para modificar su periodicidad tan sólo habrá que modificar su configuración por defecto:

```
[root@mv1]# nano /etc/cron.d/lightsquid
0 4     * * *   root    [ -x /usr/share/lightsquid/lightparser.pl ] &&
                        /usr/shar e/lightsquid/lightparser.pl
```

Práctica Nº4.- Sistemas RAID. ACLs y Cuotas. Delegación de Privilegios con Sudo

En esta práctica veremos como administrar y gestionar varios discos en nuestra máquina para sacarles el máximo rendimiento haciendo de los sistemas **RAID**. Aunque actualmente la mayoría de las placas base ofrecen la posibilidad de agrupar los discos duros **SATA** del equipo implementando un sistema **RAID** con una gestión por parte de la **BIOS** totalmente transparente para el usuario, en esta práctica aprenderemos a implementar el sistema **RAID** por software lo que nos permitirá un mayor control por parte del usuario.

Además, veremos como gestionar listas de control de acceso, **ACLs**, con la finalidad poder establecer una mayor control de los permisos concedidos a los usuarios del sistemas sobre el sistema **RAID**, y aprenderemos a administrar cuotas en los sistemas **RAID** con la finalidad de poner un límite al espacio que puede usar cada usuario.

Por último, con la finalidad de evitar que sea el usuario "**root**" quien tenga que realizar todas las tareas anteriores de gestión, administración, configuración, etc. aprenderemos a configurar el **sudo**, el cual nos permitirá delegar esas tareas en otros usuarios del sistema diferentes al **root**.

4.1.- Principales Sistemas RAID: RAID0, RAID1, RAID5 y RAID6

Sin entrar mucho en el detalle de como funcionan los diferentes tipos de sistemas RAID (*se supone que en el módulo de hardware se describen teóricamente con relativo detalle*), cabría resaltar que su implementación nos aporta (1) mayor rapidez de acceso a los datos y (2) nos garantiza una mayor disponibilidad de los datos, aunque todo ello depende del tipo de sistema RAID. Centrándonos en los tipos que más se implementan (*http://es.wikipedia.org/wiki/RAID*):

(a) RAID0 o Data Striping: compuesto por dos o más discos, **persigue disminuir los tiempos invertidos por las operaciones de lectura y escritura de datos en el volumen resultante**. Para ello, RAID0 reparte de manera proporcional al número de discos las operaciones de lectura y escritura. Esto garantiza que la velocidad de acceso a los datos, y la velocidad de su recuperación aumentará en esa misma proporción. Es decir, cuando almacenamos un archivo, este se divide en partes y estas se reparten equitativamente entre los distintos discos. Como cada disco dispone de su propia controladora, si hay *n* discos, la velocidad de escritura y recuperación aumentará en *n* igualmente. Este tipo de sistema RAID **no garantiza una mayor disponibilidad de los datos**, o lo que es lo mismo, si por un casual se daña alguno de los discos falla el sistema entero. El tamaño resultante del volumen creado bajo un sistema RAID0 es la suma del espacio de los discos involucrados.

(b) RAID1 o Mirroring: compuesto por dos o más discos, **persigue garantizar una mayor disponibilidad de los datos ante posibles fallos o desastres que se puedan producir**. Para ello, RAID1 hace una copia exacta de los datos entre los distintos discos que forman parte del volumen. De esta forma, si por algún motivo un disco falla, y perdemos el acceso a sus datos, podremos

recuperarlos a partir de los discos espejo creados. Este tipo de sistema RAID **no mejora los tiempos de acceso a los datos**. El tamaño resultante del volumen creado bajo un sistema RAID1 sería únicamente el de uno de los discos que lo forman.

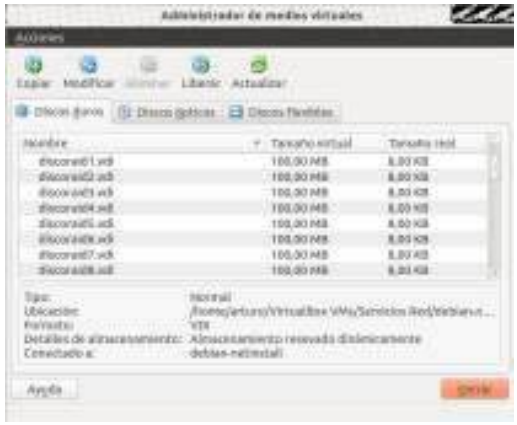
(c) RAID5/RAID6 o Data Striping with parity: compuesto por tres o más discos, persigue ofrecer las ventajas que ofrecen los dos sistemas RAID anteriores: (1) **mejorar los tiempos de acceso en operaciones de lectura y escritura a datos**, y (2) **garantizar una mayor disponibilidad de los datos mediante el uso de paridad**. Para ello, RAID5 y RAID6, por un lado distribuyen los datos entre varios de los discos que conforman el volumen (*striping*), tal como hacía un RAID0, y por otro lado, almacenan en el resto de los discos (*uno o más*) información redundante (*paridad*) que permita poder recuperar los datos en caso de fallo de cualquiera de los discos del volumen. El tamaño resultante del volumen creado bajo un sistema RAID5 o RAID6 es igual a la suma de los discos entre los cuales se realiza el striping. En concreto, RAID5 requiere al menos de 3 discos, donde uno de ellos se usa para redundancia (paridad), por lo que el espacio para datos del volumen será igual al número de discos usados menos 1, y RAID6 requiere al menos 4 discos, de los cuales dos de ellos se usan para redundancia, por lo que el espacio para datos del volumen será igual al número de discos usados menos 2.

4.2.- Administración de Sistemas RAID: mdadm

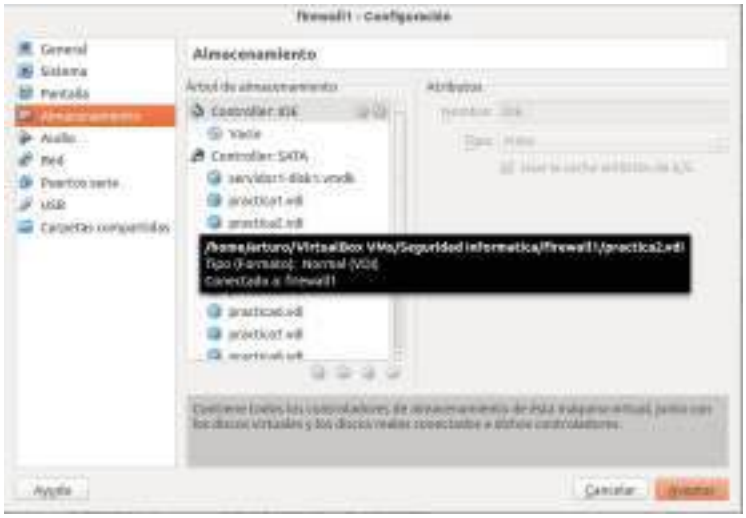
A continuación se detallarán los pasos para la implementación práctica de un sistema RAID bajo GNU/Linux con el software "**mdadm**". **mdadm** nos permite implementar entre otros un sistema RAID0, RAID1, RAID5, RAID6 o RAID10 (*para saber las posibles implementaciones con más detalle ejecutar:* ***man mdadm***).Además, con la finalidad de no tener que adquirir el material necesario para su realización (*equipos, discos duros, software, etc ...*), se realizará todo ello bajo una máquina virtual VirtualBox con sistema operativo GNU/Linux Ubuntu, ahorrándonos por tanto el coste de los discos duros IDE o SATA necesarios para la implementación de los volúmenes RAID.

A continuación, se describen los pasos a seguir para implementar un sistema RAID:

Paso Nº1.- Disponer de varios discos o particiones con los que crear los volúmenes RAID. En el caso de que las pruebas se realicen en un entorno virtualizado, por ejemplo mediante VirtualBox, será suficiente con crear varios discos virtuales desde el administrador de discos (*CTRL+D, Archivo -> Administrador de medios virtuales*).



Desde la configuración de la máquina virtual desde la que se va a realizar la práctica se pueden crear "Nuevos" discos virtuales, "Agregar" otros discos duros (**.vdi*) existentes que ya tengas, "Eliminar" discos o "Liberarlos" de la máquina virtual a la cual se encuentran asociados. Hay que recordar que en función del tipo de controladora, IDE o SATA, podremos agregar más o menos discos. Si tenemos en cuenta que con IDE tan sólo podemos agregar hasta un tope de 4 dispositivos, se aconseja hacer uso de una controladora SATA.



Una vez arrancada la máquina podremos comprobar que el sistema detecta correctamente los discos simplemente listándolos:

```
[root@mv1]# ls /dev/sd*
/dev/sda   /dev/sda2  /dev/sda6  /dev/sdc  /dev/sde  /dev/sdg  /dev/sdi
/dev/sda1  /dev/sda5  /dev/sdb  /dev/sdd  /dev/sdf  /dev/sdh
```

Paso Nº2.- mdadm nos permite crear volúmenes RAID haciendo uso de varios discos o de varias particiones de disco. En el caso de que queramos crear un volumen RAID haciendo uso de particiones, necesitaremos una herramienta de particionado de discos. Por ejemplo, podemos hacer uso de **fdisk** o **parted** en una interfaz de línea de comandos (*p.e. Ubuntu Server*), o **Gparted** si nuestro equipo dispon de una interfaz gráfica. Tanto parted, como Gparted, nos permiten igualmente dar formato a las particiones, aunque esto no será necesario, ya que una vez creado el volumen es cuando necesitaremos darle formato. Por ejemplo, en el caso de usar **fdisk**, las opciones más comunes serían las siguientes:

```
[root@mv1]# fdisk /dev/sdX
Orden (m para obtener ayuda): m
Orden  Acción
  ...
  d   Suprime una partición
  l   Lista los tipos de particiones conocidos
  n   Añade una nueva partición
  p   Imprime la tabla de particiones
  q   Sale sin guardar los cambios
```

```
...
w   Escribe la tabla en el disco y sale
x   Funciones adicionales (sólo para usuarios avanzados)
```

Paso Nº3.- Una vez que ya tenemos los discos agregados a nuestra máquina, y en en caso de ser necesario, particionados para hacer volúmenes RAID basados en particiones, pasaremos a crear el volumen RAID deseado mediante la herramienta software **mdadm**. En caso de no tener instalado el software "**mdadm**" será necesario instalarlo antes:

```
[root@mv1]# apt-get install mdadm
```

Además nos aseguraremos de que exista un subdirectorio dentro de /**dev**, llamado **md**, múltiples discos, **/dev/md**, donde almacenaremos la lista de enlaces (*links*) a los volúmenes RAID que vayamos a crear:

```
[root@mv1]# mkdir /dev/md
```

¡¡Aclaración!! mdadm nos permite crear directamente un dispositivo (*device*) o volumen RAID en **/dev/md*X***, pero no se aconseja ya que **mdadm** suele ser muy estricto con los identificadores asignados, pudiendo llegar a renombrarlos si **mdadm** lo considera necesario. Para poder asignar un nombre más amigable que haga referencia a nuestro volumen RAID, durante el proceso de creación, en lugar de indicar la ruta del dispositivo a crear, **/dev/md*X***, tan sólo asignaremos un nombre al enlace (*link*) al dispositivo RAID que se vaya a crear, *p.e.* **/dev/md/*nombre_maquina:miraid1***. Esto nos permitirá a posteriori hacer referencia al volumen RAID, tanto a través del dispositivo **/dev/md*X*** que crea **mdadm**, como a través del enlace **/dev/md/*nombre_maquina:miraid1*** que hemos asignado. Estos enlaces a los dispositivos reales deben ser creados en este subdirectorio obligatoriamente, **/dev/md**, ya que es allí donde mdadm espera que sean creados.

La sintaxis de **mdadm** que tendremos que usar para la creación de un volumen RAID será la siguiente:

```
mdadm --create /dev/md/nombre_maquina:nombre_volumen --level=raidY \
--raid-devices=N /dev/sdb /dev/sdc ... /dev/sdN
```

donde en *"nombre_maquina:nombre_volumen"* indicaremos un identificador para el enlace (*link*) a nuestro sistema RAID compuesto por el nombre de la máquina (*more /etc/hostname*) y un nombre para el volumen RAID, donde ***Y*** indica el nivel del sistema RAID (*0, Data Stripping, 1, Mirroring, 5, etc.*), y ***N*** el número de discos que utilizaremos en su implementación.
Además, tal como se ha dicho anteriormente, insistir en que el sistema RAID se puede implementar haciendo uso de los discos completos o de particiones. Por ejemplo, suponiendo que han sido particionados y formateados los discos previamente (*p.e. fdisk, Gparted*), en particiones de igual tamaño:

```
mdadm --create /dev/md/nombre_maquina:nombre_volumen --level=raidY --raid-devices=N
/dev/sdb1 /dev/sdc1 ... /dev/sdN1
```

donde el "1" indica que se va a hacer uso de la primera de las particiones de los discos utilizados.

¡¡Importante!! A la hora de crear un volumen RAID en el mundo real deberemos reflexionar a la hora de elegir entre formarlo con discos o con particiones, ya que aunque intuitivamente podamos pensar que vamos a garantizar una mayor eficiencia del espacio de los discos si usamos directamente discos, a veces resulta más aconsejable hacer uso de particiones, con la finalidad de asegurarnos los tamaños deseados. Es decir, una vez formado un volumen RAID, si tuviéramos que sustituir uno de los discos que lo forman, a veces puede resultar bastante complicado encontrar un disco del mismo tamaño al que se desea sustituir. En cambio, si hacemos uso de particiones, siempre podemos ajustar su tamaño mediante alguna de las herramientas software disponibles (*fdisk, Gparted, etc.*).

Por ejemplo, si dispusiéramos de dos discos, /dev/**sdb** y /dev/**sdc**, y quisieramos montar un volumen **RAID0**, identificándolo como **/dev/md/**equipo1:miraid0 (*md, multiples discos, hostname = mv1, identificador volumen RAID miraid0*) ejecutaríamos el siguiente comando:

```
[root@mv1]# mdadm --create /dev/md/mv1:miraid0 \
            --level=raid0 --raid-devices=2 /dev/sdb /dev/sdc
```

¡¡Advertencia!! Advertir que **/dev/md/mv1:miraid0** tan sólo es un enlace al dispositivo RAID que realmente crea **mdadm**:

```
[root@mv1]# ls -l /dev/md/mv1:miraid0
lrwxrwxrwx 1 root root 8 ene 21 19:04 /dev/md/mv1:miraid0 -> ../md127
```

Por ejemplo, en el caso anterior, **/dev/md/mv1:miraid0** es un enlace al dispositivo RAID creado **/dev/md127**.

Paso Nº4.- Comprobación del estado en que se encuentra la implementación de nuestro volumen RAID. A veces, la creación del volumen RAID no es inmediata. Para consultar el estado de su creación, ejecutaremos alguno de los siguientes comandos:

```
[root@mv1]# more /proc/mdstat
[root@mv1]# mdadm --detail /dev/md/mv1:miraid0
```

Donde **mdadm --detail /dev/md/***nombre_maquina:nombre_volumen* nos proporcionaría una información detallada del volumen RAID. Por ejemplo, si estuvieramos creando el volumen RAID0 anterior:

```
[root@mv1]# mdadm --create /dev/md/mv1:miraid0 \
            --level=raid0 --raid-devices=2 /dev/sdb /dev/sdc
[root@mv1]# mdadm --detail /dev/md/mv1:miraid0
```

Paso Nº5.- Formatearemos el sistema RAID implementado. Independientemente de que los discos o particiones que formen parte del sistema RAID ya estén formateados, habrá que explícitamente formatear el volumen RAID creado. Por ejemplo, siguiendo con el ejemplo anterior, si queremos darle formato "**ext4**", haríamos uso del comando **mkfs.ext4**:

```
[root@mv1]# mdadm --create /dev/md/mv1:miraid0 \
```

```
                --level=raid0 --raid-devices=2 /dev/sdb /dev/sdc
[root@mv1]# mdadm --detail /dev/md/mv1:miraid0
[root@mv1]# mkfs.ext4 /dev/md/mv1:miraid0
```

Paso Nº6.- Configuraremos el servicio "**mdadm**" para que cuando se reinicie el equipo la configuración del sistema RAID persista. Para ello, deberemos añadir al archivo de configuración del servicio de gestión software de los sistemas RAID "**/etc/mdadm/mdadm.conf**", las siguientes lineas correspondientes a la configuración RAID implementada. En primer lugar indicamos al servicio **mdadm** cual es la lista de dispositivos de almacenamiento vamos a hacer uso en la implementación de los volúmenes RAID, y a continuación incluiremos los detalles del último volumen RAID implementado:

```
[root@mv1]# echo "DEVICE /dev/sdb /dev/sdc" > /etc/mdadm/mdadm.conf
[root@mv1]# mdadm --detail --scan | tail -1 >> /etc/mdadm/mdadm.conf
```

¡¡Aclaración!! En los comandos anteriores se supone que estos son los primeros volúmenes RAID que se crean dentro del sistema, de hay que se use un único redireccionamiento '>' para introducir la lista de dispositivos en el **mdadm.conf**, lo cual sobrescribe todo lo que pudiera haber. En el caso de que ya hayan sido creados otros volúmenes RAID antes, utilizaremos un doble redireccionamiento:

```
[root@mv1]# echo "DEVICE /dev/sdb /dev/sdc" >> /etc/mdadm/mdadm.conf
```

En concreto, mediante el último comando anterior, escaneamos todos los sistemas RAID implementados en el sistema, y mediante el comando "tail -1", filtramos y añadimos al fichero de configuración del **mdadm** el último volumen RAID implementado. Se supone que en el caso de que hayan sido implementados ya varios, el resto habrán sido añadidos previamente.

¡¡Importante!! Al ejecutar el comando anterior "**mdadm --detail --scan**", dependiendo de la versión de mdadm que usemos, puede ser que veamos por pantalla algo parecido a lo siguiente:

```
...
ARRAY /dev/md0 level=raid0 num-devices=2 metadata=00.90 UUID=c498498c:783d...
ARRAY /dev/md5 level=raid5 num-devices=3 metadata=00.90 UUID=b5552a1f:1d6b...
ARRAY /dev/md1 level=raid1 num-devices=2 metadata=00.90 UUID=53185d99:2c47...
...
```

Si detrás de "metadata" nos aparece la versión, "**00.90**", deberemos corregirlo (*es un bug de mdadm*), y poner en su lugar "**0.90**", por lo que habrá que quitar un cero. Es decir, tras redireccionar la salida del comando anterior sobre el archivo "**mdadm.conf**" habrá que editarlo, por ejemplo con "**nano**", y quitar el "**0**" correspondiente:

```
ARRAY /dev/md0 level=raid0 num-devices=2 metadata=0.90 UUID=c498498c:783d...
ARRAY /dev/md5 level=raid5 num-devices=3 metadata=0.90 UUID=b5552a1f:1d6b...
ARRAY /dev/md1 level=raid1 num-devices=2 metadata=0.90 UUID=53185d99:2c47...
```

Paso 7.- Por último tan sólo nos quedará configurar el equipo para que el montaje del volumen RAID implementado sea automático como una nueva unidad de almacenamiento del equipo en los

posteriores reinicios del equipo. Para ello, deberemos editar el fichero de montajes automáticos del sistema GNU/Linux "**/etc/fstab**", e incluir el sistema RAID.

¡¡Importante!! Antes de configurar el **/etc/fstab** para que se realice el montaje automático del volumen RAID, es conveniente reiniciar el equipo, y comprobar que el sistema RAID implementado persiste, y es reconocido como un nuevo disposito del sistema dento del directorio /dev (*devices*): **/dev/md/***nombre_maquina:nombre_volumen*

7.1.- Creamos la carpeta donde se realizará el montaje automático:

```
[root@mv1]# mkdir /mnt/miraid0
```

7.2.- Editamos el fichero "**/etc/fstab**" incluyendo la siguiente línea al final del fichero (*siguiendo con el ejemplo anterior, **/dev/md/**mv1:miraid0*):

```
[root@mv1]# nano /etc/fstab
# Añadimos una línea similar a la siguiente
/dev/md/mv1:miraid0 /mnt/miraid0 ext4 user,rw,exec,acl,usrquota,grpquota,defaults 0 0
```

Las opciones **acl**, **usrquota**, **grpquota** se incluirán en el caso en que deseemos establecer listas de control de acceso (*ACL*) a usuarios, y gestionar cuotas a usuarios. Para ello deberán instalarse los programas **acl** y **quota** y configurarse adecuadamente, tal como veremos más tarde a lo largo de la práctica.

7.3.- Montamos el sistema raid y comprobaremos su correcto montaje:

```
[root@mv1]# mount /mnt/miraid
[root@mv1]# df -h
S.ficheros      Tamaño  Usados  Disp  Uso% Montado en
/dev/mdX        15G     7,6G    6,4G  55%  /mnt/miraid0
```

También podemos probar a reiniciar el sistema y comprobar que el volumen RAID persiste (*podemos ejecutar "df -h" y comprobar que nuestro sistema RAID esta montado*).

¡¡Aclaración!! Siguiendo con las advertencias anteriores, puede observarse una vez montado, que el dispositivo RAID que se monta **/dev/md*X*** es al que apunta el enlace **/dev/md/mv1:miraid0** indicado durante la creación del volumen RAID: "**mdadm --create /dev/md/**mv1:miraid0 ...".

Paso Nº8.- Comprobar el comportamiento del sistema RAID. La herramienta software "**mdadm**" además de crear el volumen RAID nos permite hacer las siguientes pruebas sobre él:

--fail, para provocar un fallo en un disco o partición del volumen RAID, y así comprobar que aún fallando un disco o partición siguen funcionando los sistemas RAID1 y RAID5.
--remove, para eliminar del sistema RAID un disco o partición que previamente haya fallado.
--add, para añadir un disco o partición al sistema RAID y así reemplazarlo.

Observando periódicamente el fichero "**/proc/mdstat**", o viendo los detalles del dispositivo

RAID "**mdadm --detail /dev/md*X***" podrá observarse el estado de los sistemas RAID implementados, y de los discos que lo componen.

Además, aunque **mdadm** tiene multitud de posibilidades, cabría destacar las siguientes:

Parámetro	Sintaxis Comando	Significado
--fail	**mdadm --fail /dev/mdX /dev/sdY**	Para provocar un **fallo en el disco sd*Y*** que forma parte del sistema RAID previamente implementado **md*X***
	mdadm --fail /dev/mdX /dev/sdYN	Para provocar un **fallo en la partición sd*YN*** que forma parte del sistema RAID previamente implementado **md*X***
--remove	**mdadm --remove /dev/mdX /dev/sdY** **mdadm --remove /dev/mdX**	Para eliminar del sistema RAID **md*X*** el disco o partición, **sd*Y*** o **sd*YZ***, que previamente haya fallado. También nos permite eliminar el sistema RAID una vez parado el servicio (*--stop*)
--add	**mdadm --add /dev/mdX /dev/sdZ**	Para añadir el disco o partición, **sd*Z*** o **sd*ZN***, al sistema RAID **md*X*** para reemplazar el disco o partición que haya previamente fallado, **sd*Y*** o **sd*YN***
--query	**mdadm --query /dev/sdX**	Para saber si un disco **/dev/sdX** pertenece a algún volumen RAID implementado
--stop	**mdadm --stop /dev/mdX**	Para parar el servicio ofrecido por un volumen RAID **/dev/mdX**
	mdadm --stop --scan	Para todos los volúmenes RAID existentes
--zero-superblock	**mdadm --zero-superblock /dev/sdX**	Elimina el superbloque de un disco que ha estado formando parte de un sistema RAID. Es importante hacerlo cuando queremos deshacernos de un volumen RAID, tras pararlo (*--stop*) y eliminarlo (*--remove*)

¡¡Importante!! Además del software de gestión de sistemas RAID, **mdadm**, para la realización de los ejercicios prácticos que se proponen a continuación pueden resultar útiles los siguientes comandos para administrar los discos:

Comandos de Administración de Discos	Significado

fdisk /dev/sdb	Administrar las particiones del disco (*opciones*): **n**, crear; **d**, borrar; …; **w**, guardar tabla particiones
sfdisk -d /dev/sdb | **sfdisk** /dev/sdc **--force**	Particiona un disco de forma idéntica a otro
smartctl -i /dev/hdb **hdparm** -i /dev/hdb	Obtiene un número de serie del disco para tenerlo identificado por si falla y hay que reemplazarlo
dd if=/dev/sda of=/dev/sdb bs=2M conv=noerror,sync **dd** if=/dev/sda | pv -cbrt | of=/dev/sdb	Duplica un disco
dd if=/dev/zero of=/dev/sdb bs=512 count=1	Limpiar un disco

Ej. Práctico 4.2.1: Implementación de Volúmenes RAID0, RAID1 y RAID5

Añade **8 discos a una máquina virtual Virtualbox** en la cual se haya instalado previamente un sistema operativo **Ubuntu o Debian**, mediante su administrador de discos. Suponiendo que la máquina dispone originalmente de un único disco, /dev/sda, los nuevos discos añadidos pasarán a ser identificados por el sistema como /dev/**sdb**, /dev/**sdc**, …, /dev/**sdi**. Después instala el software de gestión de volúmenes RAID **mdadm** y siguiendo los pasos de configuración explicados en el apartado 4.2 configura los siguientes volúmenes RAID (*suponiendo que el nombre de la maquina es mv1*). Una vez realizada la práctica, y realizadas las pruebas convenientes, elimina alguno de los volúmenes RAID.

Tipo RAID	Dispositivo RAID	Número de discos y Discos	
RAID0	**/dev/md/mv1:raid0**	**2**	**/dev/sdb, /dev/sdc**
RAID1	**/dev/md/mv1:raid1**	**2**	**/dev/sdd, /dev/sde**
RAID5	**/dev/md/mv1:raid5**	**3**	**/dev/sdf, /dev/sdg, /dev/sdh**

Punto Montaje
/mnt/miraid0
/mnt/miraid1
/mnt/miraid5

Una vez creados los tres volúmenes, reinicia el sistema y comprueba que siguen persistiendo. De igual forma, provoca fallos mediante **mdadm** sobre los sistemas RAID1 y RAID5, y corrobora la correcta reconstrucción del volumen haciendo uso del octavo disco, /dev/**sdi**.

Solución Ej. Pr. 4.2.1.I.- Implementación de volúmenes RAID0, RAID1 y RAID5

Comenzaremos agregando los 8 discos a la máquina virtual (*hostname = mv1*), y comprobando que son reconocidos por el sistema mostrando la lista de dispositivos SATA detectados (***sda** es el disco donde esta instalado el sistema, y los discos **sdb** a **sdi** son los 8 discos con los que realizaremos la práctica*):

```
[root@mv1]# ls /dev/sd*
/dev/sda     /dev/sdb     /dev/sdf
/dev/sda1    /dev/sdc     /dev/sdg
```

```
/dev/sda2      /dev/sdd      /dev/sdh
/dev/sda5      /dev/sde      /dev/sdi
```

Ahora pasaremos a crear y formatear los volúmenes RAID especificados en el enunciado:

```
[root@mv1]# mkdir /dev/md
[root@mv1]# mdadm --create /dev/md/mv1:raid0 --level=raid0 \
--raid-devices=2 /dev/sdb /dev/sdc
[root@mv1]# mdadm --create /dev/md/mv1:raid1 --level=raid1 \
--raid-devices=2 /dev/sdd /dev/sde
[root@mv1]# mdadm --create /dev/md/mv1:raid5 --level=raid5 \
--raid-devices=3 /dev/sdf /dev/sdg /dev/sdh
[root@mv1]# mdadm --detail /dev/md/mv1:raid0
[root@mv1]# mdadm --detail /dev/md/mv1:raid1
[root@mv1]# mdadm --detail /dev/md/mv1:raid5
[root@mv1]# mkfs.ext4 /dev/md/mv1:raid0
[root@mv1]# mkfs.ext4 /dev/md/mv1:raid1
[root@mv1]# mkfs.ext4 /dev/md/mv1:raid5
```

A continuación introduciremos en el archivo de configuración del mdadm la especificación de los tres volúmenes RAID que acabamos de crear, además de indicarle la lista de dispositivos que formarán parte de los volúmenes RAID, en el caso de que no se haya hecho previamente:

```
[root@mv1]# echo "DEVICE /dev/sdb /dev/sdc /dev/sdd /dev/sde /dev/sdf \
                  /dev/sdg /dev/sdh" > /etc/mdadm/mdadm.conf
[root@mv1]# mdadm --detail --scan | tail -3 >> /etc/mdadm/mdadm.conf
```

¡¡Aclaración!! En los comandos anteriores se supone que estos son los primeros volúmenes RAID que se crean dentro del sistema, de hay que se use un único redireccionamiento '>' para introducir la lista de dispositivos en el **mdadm.conf**, lo cual sobrescribe todo lo que pudiera haber. En el caso de que ya hayan sido creados otros volúmenes RAID antes, utilizaremos un doble redireccionamiento:

```
[root@mv1]# echo "DEVICE /dev/sdb /dev/sdc /dev/sdd /dev/sde /dev/sdf \
                  /dev/sdg /dev/sdh" >> /etc/mdadm/mdadm.conf
```

Una vez creados los volúmenes RAID, sería conveniente reiniciar el equipo para comprobar que persisten, antes de proceder a configurar el montaje automático de los volúmenes RAID mediante **/etc/fstab**:

```
[root@mv1]# init 6
[root@mv1]# ls /dev/md/*
/dev/md/mv1:raid0 /dev/md/mv1:raid1 /dev/md/mv1:raid5
```

```
[root@mv1]# mkdir /mnt/miraid0
[root@mv1]# mkdir /mnt/miraid1
[root@mv1]# mkdir /mnt/miraid5
[root@mv1]# nano /etc/fstab
```

```
…
/dev/md/mv1:raid0 /mnt/miraid0 ext4 user,rw,exec,defaults 0 0
/dev/md/mv1:raid1 /mnt/miraid1 ext4 user,rw,exec,defaults 0 0
/dev/md/mv1:raid5 /mnt/miraid5 ext4 user,rw,exec,defaults 0 0
```

Por último, realizaremos el montaje manualmente, aunque sería conveniente reiniciar el equipo para comprobar que el montaje se realiza automáticamente al iniciarse la máquina mediante la información del archivo **/etc/fstab**:

```
[root@mv1]# mount /mnt/miraid0
[root@mv1]# mount /mnt/miraid1
[root@mv1]# mount /mnt/miraid5
[root@mv1]# init 6
```

¡¡Sugerencia!! Tras implementar los volúmenes RAID sería interesante provocar fallos en alguno de los discos de los sistemas RAID1 y RAID5 (*en el RAID0 no se puede provocar, ya que provocaría la perdida de todos los datos*) y comprobar que los datos siguen estando disponibles, eliminar alguno de los discos, y agregar nuevos discos, mediante las opciones de **mdadm** y el uso del octavo disco restante **/dev/sdi**:

```
[root@mv1]# mdadm --fail /dev/md/mv1:raid1 /dev/sdd
[root@mv1]# mdadm --detail /dev/md/mv1:raid1
[root@mv1]# mdadm --remove /dev/md/mv1:raid1 /dev/sdd
[root@mv1]# mdadm --detail /dev/md/mv1:raid1
[root@mv1]# mdadm --add /dev/md/mv1:raid1 /dev/sdi
[root@mv1]# mdadm --detail /dev/md/mv1:raid1
```

Por último aprenderemos a eliminar un volumen RAID sin dejar rastros en los discos que han formado parte de dicho RAID. Comprueba su correcta eliminación reiniciando el sistema. A modo de ejemplo eliminaremos el RAID1 implementado:

```
[root@mv1]# ls -l /dev/md/mv1:miraid1
lrwxrwxrwx 1 root root 8 ene 21 19:34 /dev/md/mv1:miraid1 -> ../md128
```

Teniendo en cuenta que el dispositivo **/dev/md/mv1:miraid1** no es más que un enlace a el dispositivo real, a la hora de eliminarlo tendremos que hacer referencia a él:

```
[root@mv1]# mdadm --stop /dev/md/mv1:raid1
[root@mv1]# mdadm --remove /dev/md128
[root@mv1]# mdadm --zero-superblock /dev/sdd /dev/sde
[root@mv1]# init 6
…
[root@mv1]# ls -l /dev/md/*
[root@mv1]# df -h
```

4.3.- Administración de Usuarios y Grupos de Usuarios en GNU/Linux

Antes de ver como se gestionan las ACLs asignadas a usuarios repasaremos como crear, modificar o borrar cuentas de usuario en la máquina. La administración de cuentas de usuario en GNU/Linux puede realizarse a través de aplicaciones en modo gráfico (**GUI, Graphics User Interface**) o en modo comando (**CUI, Command User Interface**). En relación a las aplicaciones gráficas GUI suelen ser dependientes de la distribución de GNU/Linux con la que se este trabajando, lo que implica que el nombre de dicha aplicación y su aspecto será diferente en cada una de ellas (*Ubuntu, OpenSuse, etc.*). Por contra, en un entorno CUI (*Interfaz de Usuario de Línea de Comandos*), los comandos GNU/Linux utilizados en la gestión de cuentas de usuario suelen ser independientes de la distribución, razón por la cual, en esta práctica se hará uso de ellos.

En concreto, para la gestión de cuentas de usuario y grupos de usuarios en GNU/Linux disponemos de los siguientes comandos: "**useradd**", "**usermod**" "**userdel**", "**groupadd**", "**groupmod**", "**groupdel**" y "**passwd**". Su sintaxis podríamos resumirla a grandes rasgos de la siguiente forma:

Comando y Descripción	Sintaxis y Ejemplos
useradd *Crea un nuevo usuario*	**useradd -m -d** "*directorio HOME*" **-s** "*SHELL*" **** **-g** "*grupo usuarios*" **-G** "*grupos secundarios*" \ "*nombre usuario*" **useradd -m -d** /home/dominio/usuario1 **-s** /bin/bash **** **-g** dominio **-G** empleados,gestores usuario1 **useradd -m -d** /home/empleado1 **-s** /bin/bash **-g** empleados \ **-k** /etc/skel-empleado **-G** empresa empleado1
usermod *Modifica las propiedades de un usuario existente*	**usermod -m -d** "*directorio HOME*" **-s** "*SHELL*" **** **-g** "*grupo usuarios*" **-G** "*grupos secundarios*" "*nombre usuario*" **usermod -G** administradores usuario1 **usermod -d** /mnt/usuarios/usuario1 **-g** empleados usuario1
userdel *Elimina un usuario*	**userdel** "*nombre usuario*" **userdel** usuario1
groupadd *Crea un nuevo grupo de usuarios*	**groupadd** "*nombre grupo*" **groupadd** sercretaria
groupmod *Modifica un grupo existente*	**groupmod -n** "*nuevo nombre grupo*" "*nombre grupo*" **groupmod -n** gestion secretaria
groupdel *Elimina un grupo de usuarios*	**groupdel** "*nombre grupo*" **groupdel** gestion
passwd	**passwd** "*nombre usuario*"

Asigna una contraseña al usuario	

¡¡Importante!! A la hora de crear un usuario mediante **useradd** hay algunas de las opciones que sería interesante conocer en detalle:

- Opción "**-m**": sólo será necesaria en el caso de que el directorio HOME (*p.e. /home/empleado1*) a asignar a un usuario no exista y queremos que se cree en el momento de ejecutar el comando **useradd**. Es decir, que si el directorio HOME ya existe o lo creamos previamente mediante **mkdir** esta opción no será necesaria.
- Opciones "**-g**" y "**-G**": nos permiten agregar al usuario a algún grupo del sistema. Con la opción "**-g**" le cual será el grupo principal al que pertenecerá, y con "**-G**" indicamos otros posibles grupos de pertenencia. Es necesario que el grupo o grupos indicados existan previamente, sino será necesario crearlos mediante la aplicación "**groupadd**".
- Opción "**-s**": indica la shell o interprete de comandos que será asignado al usuario. En función de la shell asignada el usuario contará con la posibilidad de ejecutar en el equipo una mayor o menor cantidad de comandos del sistema. Ejecutando "more /etc/shells" podemos conocer las shells que están disponibles en nuestro sistema bajo las cuales un usuario puede iniciar sesión el la misma. Destacar que en ocasiones, por cuestiones de seguridad, nos puede interesar que el usuario a crear no tenga una shell válida, **/bin/false**, para evitar que el usuario no pueda iniciar sesión por determinadas vías (*p.e. directamente en el equipo, vía ssh, etc.*), y permitir únicamente acceso a aquellos servicios en los cuales se haya indicado expresamente (*p.e. samba, proftpd, etc.*).
- Opción "**-k**": nos permite crear a un usuario con un perfil especifico, es decir, con una estructura de directorios y archivos concreta. Por ejemplo, si quisiéramos que al crear un usuario este ya dispusiera de un conjunto preconfigurado de directorios y archivos dentro de su directorio HOME podríamos hacer uso de esta opción (*ver el siguiente ejercicio como ejemplo*).

Ej. Práctico 4.3.1: Creación de usuarios y grupos en GNU/Linux

Crea los grupos de usuarios **grupoftp1** y **grupoftp2**, y los usuarios **usuftp1**, …, **usuftp5** dentro de los grupos anteriores, teniendo en cuenta las especificaciones de la siguiente tabla:

Usuarios	Grupo Usuarios	Shell	Directorio Home	Esqueleto *(skel)* del Home *(Estructura de Archivos)*
usuftp1	**grupoftp1**	**/bin/false**	**/mnt/miraid5/usuftp*X*** *(X se corresponde con el identificador de cada uno de los usuarios 1, …, 5)*	**/etc/skel-ftp** Dir.: **datos, juegos** Arch.: **bienvenido.txt** Accesos Directos: **/var/www**
usuftp2				
usuftp3				
usuftp4	**grupoftp2**	**/bin/bash**		
usuftp5				

Donde el volumen RAID donde serán creados el HOME de los usuarios, será uno de los creados en ejercicios prácticos anteriores:

Tipo RAID	Dispositivo RAID	Número de discos y Discos		Punto Montaje
RAID5	**/dev/md/mv1:raid5**	**3**	**/dev/sdf, /dev/sdg, /dev/sdh**	**/mnt/miraid5**

¡¡Observación!! En el caso de que no se haya implementado el volumen RAID sugerido en los ejercicios prácticos anteriores, y quiera realizarse el ejercicio práctico, puede indicarse como directorio HOME otra ruta del sistema de archivos cualesquiera (*p.e. /home/usuftpX*).

Solución Ej. Pr. 4.3.1.I.- Creación de Usuarios y Grupos en GNU/Linux

Para crear las cuentas de usuario y grupos solicitados haremos uso de los comandos del sistema **useradd**, **groupadd** y **passwd**. En el caso de que alguno de los usuarios ya existiese, deberíamos hacer uso del comando **usermod**.

Comenzaremos creando el esqueleto de HOME que queremos que tengan los usuarios, según se ha especificado en el enunciado, y después crearemos los grupos y usuarios respectivamente. En concreto, a modo de ejemplo, el esqueleto HOME del usuario estará compuesto por dos directorios personales, datos y juegos, un archivo de bienvenida, bienvenido.txt, y un acceso directo a los sitios web alojados en la máquina (*Apache sugiere almacenarlos en /var/www*):

```
[root@mv1]# mkdir -p /etc/skel-ftp/datos /etc/skel-ftp/juegos
[root@mv1]# echo "Bienvenidos a este servidor!! ..." > /etc/skel-ftp/bienvenido.txt
[root@mv1]# ln -s /var/www /etc/skel-ftp/sitiosweb-apache
```

```
[root@mv1]# groupadd grupoftp1
[root@mv1]# groupadd grupoftp2
[root@mv1]# useradd -m -d /mnt/raid51/usuftp1 -s /bin/false \
                -g grupoftp1 -k /etc/skel-ftp usuftp1
[root@mv1]# useradd -m -d /mnt/raid51/usuftp2 -s /bin/false \
                -g grupoftp1 -k /etc/skel-ftp usuftp2
[root@mv1]# useradd -m -d /mnt/raid51/usuftp3 -s /bin/false \
                -g grupoftp1 -k /etc/skel-ftp usuftp3
[root@mv1]# useradd -m -d /mnt/raid51/usuftp4 -s /bin/bash \
                -g grupoftp2 -k /etc/skel-ftp usuftp4
[root@mv1]# useradd -m -d /mnt/raid51/usuftp5 -s /bin/bash \
                -g grupoftp2 -k /etc/skel-ftp usuftp5
[root@mv1]# passwd usuftp1
[root@mv1]# passwd usuftp2
[root@mv1]# passwd usuftp3
[root@mv1]# passwd usuftp4
[root@mv1]# passwd usuftp5
```

Por último, para comprobar que el esqueleto establecido en **/etc/skel-ftp** de directorios y archivos se ha creado correctamente para cada uno de los usuarios anteriores, podría listarse el contenido de alguno de los directorios HOME:

```
[root@mv1]# ls -l /mnt/raid51/usuftp1
```

Además sería conveniente comprobar que únicamente los usuarios con shell válida pueden iniciar sesión (*p.e. conexión ssh*) en el equipo, o suplantarse. En el caso de que no tenga una shell válida, **/bin/false**, habrá que acceder a través de algún servicio que se este ofreciendo en el sistema y esté configurado para que este tipo de usuarios puedan iniciar sesión (*p.e. si en la configuración de proftpd añadimos la directiva "RequireValidShell off", el servicio FTP admitirá la conexión de usuarios con shell falsa*).

4.4.- Administración de Listas de Control de Acceso (ACLs): acl

En esta parte de la presente práctica aprenderemos a gestionar las listas de control de acceso (**ACLs**) en un sistema GNU/Linux, con la finalidad de administrar de una manera más adecuada los permisos concedidos a usuarios sobre el sistema de archivos entre las distintas cuentas de usuario dadas de alta en la máquina.

Una de las mayores limitaciones que presenta GNU/Linux es la correspondiente a la administración de la seguridad y gestión de los permisos asociados a los archivos y directorios del sistema de ficheros. Por defecto, esta gestión se reduce a poder controlar los permisos de **lectura** (*r, read*), **escritura** (*w, write*), y **ejecución** (*x, execute*) correspondientes al usuario, al grupo de usuarios propietarios asignado a un fichero o directorio y al resto de usuarios. Estos permisos pueden consultarse ejecutando el comando "**ls**" en formato largo "**ls -l**":

```
[root@linux]# ls -l
-rw-rw----  1 arturo administradores  51470  2009-07-20 09:34  archivo.conf
drwxrwxr-x  2 arturo usuarios         4096   2009-07-22 12:29  videos/
lrwxrwxrwx  1 arturo usuarios         7      2009-10-22 08:21  enlace1 -> /tmp/tmp.txt
```

tipo	**permisos**	enlaces duros	**propietario**	**grupo**	tamaño	fecha/hora	**nombre**
-	**rw-rw----**	1	arturo	administradores	51470	2009-07-20 09:34	archivo.conf
d	**rwxrwxr-x**	2	arturo	usuarios	4096	2009-07-22 12:29	videos/
l	**rwxrwxrwx**	1	arturo	usuarios	7	2009-10-22 08:21	enlace1

Para gestionar estos permisos en GNU/Linux se hace uso de los comandos del sistema "**chmod**", "**chown**" y "**chgrp**". A modo de ejemplo:

Por ejemplo, para modificar los permisos de un archivo de tal forma que el usuario propietario pueda leer y escribir (*modificar*) sobre él, el grupo de propietarios tan sólo leer, y el resto de usuarios ni siquiera leer su contenido (*con el símbolo "+" se concede ese permiso, y con un "-" se deniega*):

```
[root@linux]# chmod u+rw-x,g+r-wx,o-rwx "nombre fichero"
```

En el caso de no indicar o omitir en el comando "chmod" a quien queremos que afecte el cambio de permisos "u" (*usuario propietario*), "g" (*grupo propietario*) o "o" (*otros*), por defecto el cambio se producirá para todos. Por ejemplo, si queremos que cualquier usuario pueda ejecutar un determinado fichero, o acceder al contenido de un determinado directorio:

```
[root@linux]# chmod +x "nombre fichero/directorio"
```

En el siguiente ejemplo se muestra como modificar los permisos de un directorio de tal forma que únicamente pueda acceder a su contenido (*permiso de ejecución, x*), ver su contenido (*permiso de lectura, r*), y escribir en él (*permiso de escritura o modificación, w*) el propietario del directorio únicamente, pero nadie más, ejecutaríamos el siguiente comando:

```
[root@linux]# chmod -R u+rwx,g-rwx,o-rwx "nombre directorio"
```

La opción "**-R**" nos permitirá que los permisos asignados sean recursivos, es decir, sean heredados por todos los archivos y subcarpetas que contenga el directorio afectado.

Para cambiar de usuario propietario o grupo de propietarios de un archivo o directorio:

```
[root@linux]# chown "nombre usuario" "nombre fichero/nombre directorio"
[root@linux]# chgrp "nombre grupo" "nombre fichero/nombre directorio"
```

En el caso de que queramos que el cambio sea recursivo, de tal forma que afecte al directorio indicado y todo lo que este contenga, se hará uso de la opción "**-R**":

```
[root@linux]# chown -R "nombre usuario" "nombre directorio"
[root@linux]# chgrp -R "nombre grupo" "nombre directorio"
```

Como se puede advertir a través de los ejemplos anteriores, la concesión de privilegios a usuarios y grupos sobre archivos y directorios de manera nativa esta muy limitada. Es decir, si quisiéramos especificar que sobre un archivo o directorio diferentes usuarios y grupos, tuvieran a su vez distintos privilegios, por defecto en GNU/Linux sería imposible. Para suplir esta carencia se hace uso de "**Listas de Control de Acceso**", **ACL**. Estas nos van a permitir asignar una lista de nombres de usuarios y grupos con sus privilegios de manera individualizada.

Para saber como hacer uso de las ACLs seguiremos los siguientes pasos:

4.4.1.- Comprobación de que nuestro Kernel admite Listas de Control de Acceso, ACLs, dentro de nuestro sistema de archivos (p.e. **ext4**). En concreto, el siguiente comando nos informará de la posibilidad de gestionar ACLs en los distintos sistemas de archivos soportados por GNU/Linux (**ext3**, **ext4**, **jfs**, **xfs**, **nfs**, etc):

```
[root@linux]# more /boot/config-`uname -r` | grep ACL
CONFIG_EXT2_FS_POSIX_ACL=y
CONFIG_EXT3_FS_POSIX_ACL=y
CONFIG_EXT4_FS_POSIX_ACL=y
CONFIG_REISERFS_FS_POSIX_ACL=y
CONFIG_JFS_POSIX_ACL=y
...
```

4.4.2.- Instalar el paquete software "**acl**", el cual nos permitirá gestionar ACLs dentro de nuestro sistema de archivos. De esta forma dispondremos de las utilidades de gestión de las listas de control

de acceso "**getfacl**" y "**setfacl**".

```
[root@linux]# apt-get install  acl
```

4.4.3.- Editar el fichero "**/etc/fstab**" para activar la gestión de Listas de Control de Acceso, ACLs, en las particiones deseadas de manera permanente. De esta forma, cuando nuestro equipo se reinicie consultará el archivo **fstab** y advertirá que la unidad de almacenamiento especificada tiene soporte para ACLs:

```
[root@linux]# nano /etc/fstab
...
/dev/particion /punto/de/montaje ext4 defaults,acl 0 0
```

¡¡Advertencia!! Otra opción sería montar la partición manualmente. Si quisiéramos activarlos únicamente de manera temporal para la sesión en en que nos encontramos ejecutaríamos el siguiente comando:

```
[root@linux]#  mount -t ext4 /dev/particion -o defaults,acl /punto/de/montaje
```

Por último, si no queremos reiniciar el equipo deberemos rehacer el montaje para que surtan efectos los cambios realizados en "**/etc/fstab**" (*sólo en el caso de que no hayamos realizado el montaje manualmente mediante "mount"*):

```
[root@linux]# mount –o remount /punto/de/montaje
```

A partir de este momento, la unidad de almacenamiento especificada ya estará preparada para soportar ACLs. Por tanto, a continuación se mostrará como utilizar las herramientas de gestión de ACLs: "**getfacl**" y "**setfacl**". Advertir que no es necesario que sea el usuario "root" el que haga uso de estas herramientas de gestión, sino el mero propietario del archivo o directorio.

Comando y Descripción	Ejemplos de Uso
setfacl Permite asignar una ACL a un directorio (*con -**R** de manera recursiva a todos sus subdirectorios*)	[usuario@linux]$ **setfacl -m user**:*nombre_usuario*:r-x \ *nombre_directorio|nombre_archivo* [usuario@linux]$ **setfacl -Rm user**:*nombre_usuario*:r-x \ *nombre_directorio* [usuario@linux]$ **setfacl -m u**:*nombre_usuario*:r-x \ *nombre_directorio|nombre_archivo* [usuario@linux]$ **setfacl -Rm group**:*nombre_grupo*:r-x \ *nombre_directorio* [usuario@linux]$ **setfacl -Rm g**:*nombre_grupo*:r-x *nombre_directorio* [usuario@linux]$ **setfacl -Rm ** **user**:*nombre_usuario*:r-x,**user**:*otro_nombre*:rwx,... \ *nombre_direc* [usuario@linux]$ **setfacl -x user**:*nombre_usuario nombre_directorio* [usuario@linux]$ **setfacl -Rx user**:*nombre_usuario* \ *nombre_directorio* [usuario@linux]$ **setfacl -b** *nombre_directorio*

	[usuario@linux]$ **setfacl -Rb** *nombre_directorio*
getfacl Permite conocer las ACL de un directorio (*con -**R** de manera recursiva, la del directorio y la de sus subdirectorios; con -**p** las rutas serán absolutas desde la raíz /*)	[usuario@linux]$ **getfacl -p** "*nombre directorio*" [usuario@linux]$ **getfacl -Rp** "*nombre directorio*"

¡¡Importante!! En "**setfacl**", la opción "**-m**" nos deja modificar las ACL, la "**-x**" las elimina para un usuario, y "**-b**" elimina todas las ACL impuestas sin excepción, dejando únicamente los permisos nativos del sistema. El comando "**chacl**" es equivalente a "**setfacl -m**". En el caso de querer modificar las ACL de un directorio, "**-m**", y que además tenga un efecto recursivo sobre todos sus subdirectorios, "**-R**", se especificará, "**-Rm**", primero "**R**" y luego "**m**"; de igual forma con el resto de opciones.

4.4.5.- Por último, señalar que **getfacl** nos permite guardar las ACLs asignadas con la finalidad de poder restaurarlas posteriormente:

```
[usuario@linux]$ getfacl -Rp /ruta/directorio > /ruta/get_rep.acl
[usuario@linux]$ setfacl --restore=/ruta/get_rep.acl
```

Ej. Práctico 4.4.1: Gestión de ACLs en GNU/Linux

Siguiendo con el ejercicio práctico anterior 4.3.1, asigna a los usuarios creados las **ACLs** que se especifican a continuación en la siguiente tabla. Además deberás hacer un backup de las ACLs asignadas en **/root/backup-acls.bkp**, para a continuación eliminar las ACLs y probar a restaurarlas.

Usuarios	Grupo Usuarios	Shell	Directorio Home
usuftp1	**grupoftp1**	**/bin/false**	**/mnt/miraid5/usuftp*X*** *(X se corresponde con el identificador de cada uno de los usuarios 1, …, 5)*
usuftp2			
usuftp3			
usuftp4	**grupoftp2**	**/bin/bash**	
usuftp5			

Donde el volumen RAID donde serán creados el HOME de los usuarios, será el creado en el ejercicio práctico anterior 4.2.1:

Tipo RAID	Dispositivo RAID	Número de discos y Discos		Punto Montaje
RAID5	**/dev/md/mv1:raid5**	**3**	**/dev/sdf, /dev/sdg, /dev/sdh**	**/mnt/miraid5**

En cuanto a las ACLs a asignar a los usuarios anteriores sobre diferentes subdirectorios dentro del volumen RAID:

Directorio	Usuarios / Grupos	ACL	Fichero Backup ACLs
/mnt/miraid5/videos **/mnt/miraid5/musica** **/mnt/miraid5/elibros**	**usuftp1, usuftp2**	**rwx**	**/root/backup-acls.bkp**
	usuftp3, grupoftp2	**r-x**	
/mnt/miraid5/documentacion **/mnt/miraid5/basura**	**grupoftp1**	**r--**	
	grupoftp2	**rwx**	
/mnt/miraid5/informes **/mnt/miraid5/facturas** **/mnt/miraid5/nominas**	**grupoftp1, usuftp4**	**rwx**	
	usuftp5	**---**	

Solución Ej. Pr. 4.4.1.I.- Gestión de ACLs en GNU/Linux

Haciendo uso de los comandos setfacl y getfacl, asignaremos y visualizaremos las ACLs, además de hacer el backup de las ACLs solicitado:

```
[root@linux]# setfacl -m u:usuftp1:rwx,g:usuftp2:rwx \
              /mnt/miraid5/videos /mnt/miraid5/musica /mnt/miraid5/elibros
[root@linux]# setfacl -m u:usuftp3:r-x,g:grupoftp2:r-x \
              /mnt/miraid5/videos /mnt/miraid5/musica /mnt/miraid5/elibros
[root@linux]# setfacl -m g:grupoftp1:r--,g:grupoftp2:rwx \
              /mnt/miraid5/documentacion /mnt/miraid5/basura
[root@linux]# setfacl -m g:grupoftp1:rwx,u:usuftp4:rwx,u:usuftp5:--- \
              /mnt/miraid5/informes /mnt/miraid5/facturas /mnt/miraid5/nominas
[root@linux]# getfacl -Rp /mnt/miraid5 | more
[root@linux]# getfacl -Rp /mnt/miraid5 > /root/backup-acls.bkp
```

Para eliminar todas las ACLs asignadas:

```
[root@linux]# setfacl -Rb /mnt/miraid5
```

Y por último, probaremos a restaurarlas, y las volver a mostrarlas:

```
[root@linux]# setfacl --restore=/root/backup-acls.bkp
[root@linux]# getfacl -Rp /mnt/miraid5 | more
```

4.5.- Administración de Cuotas: quota y quotatool

En este apartado aprenderemos a asignar cuotas de espacio en disco para que cada usuario pueda hacer uso del espacio existente de una manera racionada.

¡¡**Importante**!! Para establecer cuotas de espacio en disco en GNU/Linux es necesario instalar el paquete software "**quota**". Mediante este software pueden establecerse límites a nivel de bloque (*número de KBytes*) o inodo (*número de archivos*), o ambos simultáneamente. Al igual que en Microsoft Windows las cuotas se deben establecer a nivel de partición de disco, y no puede establecerse una cuota independiente por carpeta o directorio. Al igual que con las ACLs, para indicar al sistema sobre que particiones se van a establecer las cuotas será necesario editar el fichero "**/etc/fstab**" y agregar los parámetros "**usrquota**" y "**grpquota**" en la cuarta columna de la línea asociada a la partición o unidad de almacenamiento sobre la cual queremos asignar cuotas. Después, será necesario rehacer el montaje de aquellas particiones afectadas en el fichero "**fstab**", p.e., "**mount –o remount /mnt/datos**". Ejecutando "**ls –l** *raíz_patición*" deberían aparecer los archivos "**aquota.user**" y "**aquota.group**" que nos advierten de la habilitación de cuotas. A continuación se mostrarán los pasos para la correcta configuración de cuotas.

4.5.1.- Instalación del paquete software "**quota**" (*posteriormente instalaremos "**quotatool**" para gestionar las cuotas de "quota" de una manera más sencilla*):

```
[root@linux]# apt-get install  quota
```

4.5.2.- Editaremos el fichero "**/etc/fstab**" para activar las cuotas de espacio en las particiones o unidades de almacenamiento deseadas de manera permanente:

```
/dev/particion /punto/de/montaje ext4 defaults,acl,usrquota,grpquota 0 2
```

Si quisiéramos activarlas únicamente de manera temporal para la sesión en en que nos encontramos ejecutaríamos el siguiente comando:

```
[root@linux]#  mount -t ext4 /dev/particion -o defaults,acl,usrquota,grpquota /punto/de/montaje
```

4.5.3.- Deberemos rehacer el montaje para que surtan efectos los cambios realizados en "**/etc/fstab**" (*en caso de que no hayamos realizado el montaje manualmente mediante "mount"*):

```
[root@linux]# mount –o remount /punto/de/montaje
[root@linux]# ls –l raíz_patición
```

4.5.4.- Ahora la partición o punto de montaje ya esta preparada para soportar cuotas, pero antes de activarlas mediante el comando **quotaon**, deberemos chequear el sistema mediante **quotacheck** para comprobar que puede soportar cuotas: "**quotacheck**", "**quotaon**", "**quotaoff**".

¡¡Importante!! Tras el chequeo mediante **quotacheck** ya podremos activar el sistema de cuotas con **quotaon**. No obstante, si ya existe un sistema de cuotas previo será necesario desactivarlo

mediante **quotaoff** antes de realizar el chequeo, ya que si no nos pueden aparecer errores.

```
[root@linux]# quotaoff -v /punto/de/montaje
[root@linux]# quotaoff -a
```

Comando	Ejemplo y Descripción
quotacheck	[root@linux]# **quotacheck –augmv**
	Chequea los sistemas de archivos que tienen configuradas cuotas. Verifica y repara el control de cuotas (*-a, todos los sitemas, -u, por usuario, -g, por grupo, -m, evita su desmontaje, -v, modo verboso*)
	[root@linux]# **quotacheck –ugmv** */punto_de_montaje*
	Chequea únicamente la partición o sistema de archivos indicado, "*/punto_de_montaje*"
quotaon	[root@linux]# **quotaon -avug**
	Activa el sistema de cuotas sobre todos los puntos de montaje que lo soporten, "-a", en modo verboso, "-v", tanto para cuotas de usuario, "-u", como de grupo, "-g"
	[root@linux]# **quotaon -ugv** */punto_de_montaje*
	Activa el sistema de cuotas para usuarios y grupos sobre el sistema de archivos o partición indicada (*p.e. /home*)
quotaoff	[root@linux]# **quotaoff -v** */punto_de_montaje*
	Desactiva el sistema de cuotas sobre el sistema de archivos o partición indicada (*p.e. /home*)

4.5.5.- A continuación se mostrará como utilizar las distintas herramientas de gestión de cuotas de que se dispone con la instalación de "**quota**": "**edquota**", "**quota**" y "**repquota**". No obstante, para la asignación de cuotas, como veremos más adelante, una mejor opción es hacer uso de la herramienta software "**quotatool**".

¡¡Importante!! A la hora de establecer cuotas, tenemos que tener claro de antemano los tipos de límites que pueden establecerse, y el significado del tiempo de gracia. En concreto, en sistemas GNU/Linux pueden limitarse dos aspectos:

(1) El **espacio máximo** que puede ser ocupado por un usuario o grupo de usuarios sobre el sistema de archivos o punto de montaje que se especifique.
(2) El **número máximo de inodos** que puede crear un usuario o grupo de usuarios en el sistema de archivos o punto de montaje que se especifique. De esta forma, estamos limitando el número máximo de archivos y directorios que un usuario puede crear dentro del punto de montaje que se especifique.

Además, existen dos tipos de límites, **duro** (*hard*) y **blando** (*soft*). El límite duro, es un límite que no puede llegar a ser superado por el usuario, en cambio, el límite blando puede ser superado hasta que se alcance el límite duro, teniendo en cuenta, que a partir del momento en que el

usuario se salta el límite blando se empieza a contabilizar un tiempo límite durante el cual el usuario deberá solventar el problema de cuota que tiene, conocido como **tiempo de gracia**, transcurrido el cual el usuario ya no podrá seguir continuar ocupando más espacio, o creando más archivos o directorios.

Aunque el comando **edquota** nos permite editar y asignar cuotas, y por tanto, límites blandos y duros, y establecer un tiempo de gracia adecuado, como se verá más adelante, es mucho más cómodo hacer uso de la herramienta **quotatool**.

Comando	Ejemplo y Descripción
edquota	[root@linux]# **edquota** -u *nombre_usuario* –f *punto_de_montaje*
	Edita las cuotas sobre el punto de montaje indicado para el usuario deseado
	[root@linux]# **edquota** -u *nombre_usuario*
	Edita las cuotas para el usuario indicado sobre todos los sistemas de archivos que tengan habilitado el sistema de cuotas previamente mediante **quotaon**
	[root@linux]# **edquota** -g *nombre_grupo*
	Igual que con la opción "**-u**" pero a nivel de grupos de usuarios
	[root@linux]# **edquota –t**
	Permite establecer un tiempo de gracia a nivel global. Este empieza a tenerse en cuenta a partir del momento en que un usuario se salta el límite blando (*soft*)
	[root@linux]# **edquota** –u *nombre_usuario* –T
	Permite establecer un tiempo de gracia para el usuario indicado, siendo este obligatoriamente menor al global
	[root@linux]# **edquota** –p *usuario1 usuario2 usuario3 …*
	Permite establecer la cuota del usuario usuario1 como prototipo (*-p*) para el resto de usuarios. Es una forma de replicar cuotas de manera generalizada
quota	[root@linux]# **quota** –s -u *nombre_usuario*
	Verifica las cuotas de disco establecidas para el usuario indicado
	[root@linux]# **quota** –s -g *grupo_usuarios*
	Verifica las cuotas de disco establecidas para el grupo indicado
	[root@linux]# **quota**
	Verifica las cuotas del usuario que tiene activa la sesión
repquota	[root@linux]# **repquota –s** *punto_de_montaje*
	Genera un informe o reporte global de las cuotas de todos los usuarios del sistema de archivos o punto de montaje indicado (*p.e. /home*)
	[root@linux]# **repquota –sugv** *punto_de_montaje*
	Genera un reporte global de las cuotas de todos los usuarios y grupos del sistema de archivos o punto de montaje indicado (*p.e. /mnt/datos*)

	[root@linux]# **repquota –sugav**
	Genera un reporte global de las cuotas de todos los usuarios y grupos de todos los sistemas de ficheros o puntos de montaje que tienen habilitadas las cuotas
warnquota	[root@linux]# **warnquota**
	Avisa a los usuarios por mensajería que hayan excedido las cuotas en su nivel suave, SOFT. En "/etc/warnquota.conf" podemos configurar los mensajes que se enviarán a estos usuarios

4.5.6.- Como ya se ha advertido en el apartado anterior, aunque el comando edquota nos permite editar y asignar cuotas, una forma más sencilla de hacerlo es mediante "**quotatool**", que como veremos, nos facilitará muchísimo su gestión. Para poder utilizarla, deberemos instalarla en primer lugar:

```
[root@linux]# apt-get install  quotatool
```

Mediante las opciones o parámetros del comando "**quotatool**" podremos establecer la cuota deseada, según lo siguiente:

Parámetro **quotatool**	Utilidad
"**-u | -g**"	Nos permite indicar el nombre del usuario o grupo de usuarios al que deseamos que le afecte la cuota
"**-b**"	Nos permite indicar cual será el límite a establecer en relación al máximo espacio en disco (*bloques*) que se podrá ocupar
"**-i**"	Nos permite indicar cual será el límite a establecer en relación al máximo número de ficheros (*inodos*) que se podrán crear
"**-q**"	Nos permite establecer un límite blando. Una vez sobrepasado este límite, se empezará a contabilizar el tiempo o periodo de gracia, durante el cual el usuario podrá seguir guardando información, o almacenando ficheros, hasta que se alcance el límite duro. Transcurrido ese tiempo, el usuario ya no podrá guardar más información en la partición correspondiente hasta que no reduzca su ocupación por debajo de la cantidad establecida como límite blando
"**-l**"	Nos permite establecer un límite duro
"**-t**"	Nos permite establecer un tiempo de gracia. Este tiempo empezará a contabilizarse una vez que un usuario se salte el límite blando impuesto en cuanto a cantidad e espacio o número de inodos

A continuación se muestra la sintaxis del comando quotatool, y algunos ejemplos de uso:

```
[root@linux]# quotatool -u|-g nombre_usuario|grupo -b|-i -q limite_blando /punto/de/montaje
[root@linux]# quotatool -u|-g  nombre_usuario|grupo -b|-i -l limite_duro /punto/de/montaje
[root@linux]# quotatool -u|-g -b|-i -t "tiempo_gracia" /punto/de/montaje
```

Comando y Descripción
[root@linux]# **quotatool -u arturo -b -q 500MB /mnt/datos**
Establece un límite blando, -q, de espacio en disco, -b, de 500MB al usuario, -u, "arturo" en el punto de montaje o sistema de archivos /mnt/datos
[root@linux]# **quotatool -u arturo -b -l 700MB /mnt/datos**
Similar al anterior, pero en este caso se ha establecido un límite duro, -l, de 700MB
[root@linux]# **quotatool -u arturo -b -q 500MB -l 700MB /mnt/datos**
Una convinación de los dos comandos anteriores
[root@linux]# **quotatool -u arturo -i -q 100000 -l 100500 /mnt/datos**
Establece un límite relativo al número de inodos o archivos que puede crear el usuario, -u, "arturo", en el punto de montaje /mnt/datos, con un límite blando, -q, de 100000 inodos, y uno duro de 100500 inodos
[root@linux]# **quotatool -u -b -t "5 days" /mnt/datos**
Establece un tiempo de gracia, -t, de 5 días, "5 days", con carácter general para todos los usuarios, para subsanar el problema de cuotas en el caso de saltarse el límite blando establecido al límite de espacio en disco, -b
[root@linux]# **quotatool -u arturo -b -q 500MB -l 700MB -i -q 100000 -l 100500 /mnt/datos**
Establece un límite blando y duro tanto de espacio en disco como de inodos al usuario "arturo" sobre el punto de montaje /mnt/datos

Ej. Práctico 4.5.1: RAID, ACLs y Cuotas

Agrega cinco nuevos discos de 350MB cada uno a una de las máquinas virtuales que tengas disponible en VirtualBox (*p.e. /dev/sdb, ... y /dev/sdf*) con la finalidad de repasar todo lo visto en la presente práctica: (A) Implementación de volúmenes RAID5 y RAID6 con la finalidad de aumentar la fiabilidad y rapidez de acceso a datos de nuestro equipo, (B) la gestión de las ACLs que nos permitirán controlar los privilegios de los distintos usuarios dados de alta en el equipo, y (C) la administración de Cuotas que nos permitirán establecer límites de espacio en disco (*a modo de ejemplo, se supondrá que el nombre de la máquina donde se implementa la práctica es "mv1"*).

Para ello, deberemos cumplir los siguientes aspectos:

(A) Implementaremos volúmenes RAID haciendo uso de particiones de disco, en lugar de discos enteros. Por ello, comenzaremos particionando cada uno de los discos haciendo uso de alguna herramienta de particionado, p.e. **fdisk**, de tal forma que cada disco disco de **350MB** sea dividido en tres particiones: 1) **120MB**, 2) **130MB** y 3) **100MB**.

A continuación agruparemos las distintas particiones de los **cuatro primeros discos** y formaremos un volumen RAID5 y dos volúmenes RAID6 en formato **ext4** con las características que se muestran en la siguiente tabla, teniendo en cuenta, que los volúmenes RAID deberán montarse de mantera automática mediante **/etc/fstab** con las opciones de ACL y cuotas para usuario y grupo (*el quinto disco /dev/sdf se usará para suplir fallos provocados intencionadamente a los*

volúmenes RAID):

Volúmenes RAID	4 Particiones que forman los RAID	Punto de Montaje
RAID5 /dev/md/mv1:raid51	120MB (p.e. /dev/sdb1, ..., /dev/sde1)	**/mnt/raid51**
RAID6 /dev/md/mv1:raid61	130MB (p.e. /dev/sdb2, ..., /dev/sde2)	**/mnt/raid61**
RAID6 /dev/md/mv1:raid62	100MB (p.e. /dev/sdb3, ..., /dev/sde3)	**/mnt/raid62**

(B) Deberá configurarse las siguientes ACLs sobre diferentes subcarpetas ubicadas en los puntos de montaje anteriores asociadas a los siguientes usuarios, que deberás crear previamente:

Usuarios	Grupo Usuarios	Shell	Directorio Home
usuftp1	**grupoftp1**	**/bin/false**	**/mnt/raid51/usuftp*X*** *(X se corresponde con el identificador de cada uno de los usuarios 1, …, 5)*
usuftp2			
usuftp3			
usuftp4	**grupoftp2**	**/bin/bash**	
usuftp5			

En cuanto a las ACLs a asignar:

Directorios	Usuarios / Grupos	ACL	Backup
/mnt/raid51/usuftp*X* *Directorios HOME*	**grupoftp1, grupoftp2**	**rwx**	**/root/acl-raid51.bkp**
/mnt/raid61/videos /mnt/raid61/musica /mnt/raid61/elibros	**usuftp1, usuftp2**	**rwx**	**/root/acl-raid61.bkp**
	usuftp3, grupoftp2	**r-x**	
/mnt/raid61/documentacion /mnt/raid61/basura	**grupoftp1**	**r--**	
	grupoftp2	**rwx**	
/mnt/raid62/informes /mnt/raid62/facturas /mnt/raid62/nominas	**grupoftp1, usuftp4**	**rwx**	**/root/acl-raid62.bkp**
	usuftp5	**---**	

(C) Deberán asignarse cuotas sobre los puntos de montaje anteriores, teniendo en cuenta las ACLs anteriores (*si un usuario no tiene permisos de escritura, no tiene sentido asignarle cuota*) que el **tiempo de gracia** deberá cambiarse para que sea **2 días**:

Punto de Montaje	Usuario / Grupo	Cuota Bloque	Cuota Inodos
/mnt/raid51 *Directorios HOME y Subdirectorios*	**usuftp1, usuftp2**	10MB/14MB	100/110
	usuftp3, usuftp4	5MB/8MB	18/22
	usuftp5	4MB/6MB	18/22

/mnt/raid61	**grupoftp1**	-	-
	usuftp4, usuftp5	18MB/20MB	120/135
/mnt/raid62	**usuftp1, usuftp4**	30MB/31MB	600/720
	usuftp2, usuftp3	15MB/18MB	180/200
	usuftp5	-	-

Por último realizaremos las comprobaciones del correcto funcionamiento de la configuración implementada. Para ello, configura **tres servicios FTP** (*p.e. proftpd, apt-get install proftpd*) por los puertos **21001**, **21002** y **21003** de tal forma que puedan acceder a él los usuarios anteriores. Comprueba el límite de cuota y los permisos establecidos por las ACLs:

Sitio FTP	Usuarios Permitidos (LOGIN)	DefaultRoot (Raiz del FTP)
FTP1 – Puerto 21001	Todos	*~ Directorio HOME*
FTP2 – Puerto 21002	grupoftp1, usuftp4	/mnt/raid61
FTP3 – Puerto 21003	grupoftp2, usuftp1	/mnt/raid62

¡¡Aclaración!! A continuación se muestra un ejemplo de configuración del servicio FTP **proftpd** (*p.e. dirección ip del servidor 192.168.1.1*), teniendo en cuenta que el archivo de configuración del servicio **proftpd** esta ubicado en"**/etc/proftpd/proftpd.conf**":

```
[root@mv1]# nano /etc/proftpd/proftpd.conf
# Incluir la siguiente configuración al comienzo del archivo de configuración proftpd.conf
<VirtualHost 192.168.1.1>
        Port 21001
        RequireValidShell off
        DefaultRoot ~
        <Limit LOGIN>
                AllowGroup grupoftp1 grupoftp2
                DenyAll
        </Limit>
</VirtualHost>
<VirtualHost 192.168.1.1>
        Port 21002
        RequireValidShell off
        DefaultRoot /mnt/raid61
        <Limit LOGIN>
                AllowGroup grupoftp1
                AllowUser usuftp4
                DenyAll
        </Limit>
</VirtualHost>
<VirtualHost 192.168.1.1>
        Port 21003
        RequireValidShell off
```

```
        DefaultRoot /mnt/raid62
        <Limit LOGIN>
                AllowGroup grupoftp2
                AllowUser usuftp1
                DenyAll
        </Limit>
</VirtualHost>
```

Solución Ej. Pr. 4.5.1.I.- Gestión de Volúmenes RAID5 y RAID6, ACLs y Cuotas

Para dar solución al ejercicio práctico planteado seguiremos los siguientes pasos:

(A) Implementación de los volúmenes RAID5 y RAID6.

Una vez agregados los 5 discos a la máquina (*p.e. /dev/sdb*, …, */dev/sdf*), comenzaremos la práctica particionando uno de los discos mediante **fdisk**, creando tres particiones primarias de 120MB, 130MB y 100MB respectivamente, para después hacer una copia del particionado sobre el resto mediante **sfdisk**:

```
[root@mv1]# fdisk /dev/sdb
Command (m for help): n
Partition type:
  p   primary (0 primary, 0 extended, 4 free)
  e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-716799, default 2048): 2048
Last sector, +sectors or +size{K,M,G} (2048-716799, default 716799): +120MB

Command (m for help): n
Partition type:
  p   primary (1 primary, 0 extended, 3 free)
  e   extended
Select (default p): p
Partition number (1-4, default 2): 2
First sector (236423-716799, default 236423): 236423
Last sector, +sectors or +size{K,M,G} (236423-716799, default 716799): +130MB

Command (m for help): n
Partition type:
  p   primary (2 primary, 0 extended, 2 free)
  e   extended
Select (default p): p
Partition number (1-4, default 3): 3
First sector (490329-716799, default 490329): 490329
Last sector, +sectors or +size{K,M,G} (490329-716799, default 716799): 716799
```

```
Command (m for help): p
Disk /dev/sdq: 367 MB, 367001600 bytes
255 heads, 63 sectors/track, 44 cylinders, total 716800 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
Device Boot      Start         End      Blocks   Id  System
/dev/sdq1            2048      236422      117187+  83  Linux
/dev/sdq2          236423      490328      126953   83  Linux
/dev/sdq3          490329      716799      113235+  83  Linux

Command (m for help): w
[root@mv1]# sfdisk -d /dev/sdb | sfdisk /dev/sdc --force
[root@mv1]# sfdisk -d /dev/sdb | sfdisk /dev/sdd --force
[root@mv1]# sfdisk -d /dev/sdb | sfdisk /dev/sde --force
[root@mv1]# sfdisk -d /dev/sdb | sfdisk /dev/sdf –force
```

Después, antes de empezar a crear los volúmenes RAID, comprobaremos que los discos están correctamente particionados listando los dispositivos reconocidos por la máquina:

```
[root@mv1]# ls /dev/sd*
/dev/sda   /dev/sdb   /dev/sdc   /dev/sdd   /dev/sde   /dev/sdf
/dev/sda1  /dev/sdb1  /dev/sdc1  /dev/sdd1  /dev/sde1  /dev/sdf1
/dev/sda2  /dev/sdb2  /dev/sdc2  /dev/sdd2  /dev/sde2  /dev/sdf2
/dev/sda5  /dev/sdb3  /dev/sdc3  /dev/sdd3  /dev/sde3  /dev/sdf3
```

Ahora pasaremos a crear y formatear los volúmenes RAID: un RAID5 con las 4 primeras particiones de cuatro de los discos, y dos RAID6 con el resto de particiones, tal como se especifica en la tabla del enunciado (*las particiones del último disco /dev/sdf las usaremos en pruebas de fallos*):

```
[root@mv1]# mkdir /dev/md
[root@mv1]# mdadm --create /dev/md/mv1:raid51 --level=raid5 \
--raid-devices=4 /dev/sdb1 /dev/sdc1 /dev/sdd1 /dev/sde1
[root@mv1]# mdadm --create /dev/md/mv1:raid61 --level=raid6 \
--raid-devices=4 /dev/sdb2 /dev/sdc2 /dev/sdd2 /dev/sde2
[root@mv1]# mdadm --create /dev/md/mv1:raid62 --level=raid6 \
--raid-devices=4 /dev/sdb3 /dev/sdc3 /dev/sdd3 /dev/sde3
[root@mv1]# mdadm --detail /dev/md/mv1:raid51
[root@mv1]# mdadm --detail /dev/md/mv1:raid61
[root@mv1]# mdadm --detail /dev/md/mv1:raid62
[root@mv1]# mkfs.ext4 /dev/md/mv1:raid51
[root@mv1]# mkfs.ext4 /dev/md/mv1:raid61
[root@mv1]# mkfs.ext4 /dev/md/mv1:raid62
```

Por último, introduciremos en el archivo de configuración del mdadm la especificación de los tres volúmenes RAID que acabamos de crear, más la lista de dispositivos utilizados:

```
[root@mv1]# echo "DEVICE /dev/sd*" >> /etc/mdadm/mdadm.conf
[root@mv1]# mdadm --detail --scan | tail -3 >> /etc/mdadm/mdadm.conf
```

Una vez creados los dispositivos RAID, sería conveniente reiniciar el equipo para comprobar que persisten, antes de proceder a configurar el montaje automático de los volúmenes RAID mediante **/etc/fstab** con soporte para acl y cuotas:

```
[root@mv1]# init 6
[root@mv1]# ls /dev/md/*
/dev/md/mv1:raid51 /dev/md/mv1:raid61 /dev/md/mv1:raid62
```

```
[root@mv1]# nano /etc/fstab
…
/dev/md/mv1:raid51 /mnt/raid51 ext4 user,rw,exec,defaults,acl,usrquota,grpquota 0 0
/dev/md/mv1:raid61 /mnt/raid61 ext4 user,rw,exec,defaults,acl,usrquota,grpquota 0 0
/dev/md/mv1:raid62 /mnt/raid62 ext4 user,rw,exec,defaults,acl,usrquota,grpquota 0 0
```

Por último, comprobaríamos su correcto montaje, y que al reiniciar la máquina el montaje se realiza automáticamente:

```
[root@mv1]# mount  /mnt/raid51
[root@mv1]# mount  /mnt/raid61
[root@mv1]# mount  /mnt/raid62
[root@mv1]# df -h
[root@mv1]# init 6
…
[root@mv1]# df -h
```

¡¡Sugerencia!! Tras implementar los volúmenes RAID sería interesante provocar fallos en alguno de los discos de los sistemas RAID1 y RAID5 (*en el RAID0 no se puede provocar, ya que provocaría la perdida de todos los datos*) y comprobar que los datos siguen estando disponibles. El disco **/dev/sdf** puede usarse para suplir los fallos provocados mediante **mdadm**.

Por ejemplo, si provocamos un fallo en la primera partición del disco **/dev/sde1**, podremos comprobar que los datos persisten al tratarse de un RAID5, y posteriormente suplir dicha partición con la primera partición del disco **/dev/sdf1**:

```
[root@mv1]# mdadm --fail /dev/md/mv1:raid51 /dev/sde1
[root@mv1]# mdadm --detail /dev/md/mv1:raid51
[root@mv1]# mdadm --remove /dev/md/mv1:raid51 /dev/sde1
[root@mv1]# mdadm --detail /dev/md/mv1:raid51
[root@mv1]# mdadm --add /dev/md/mv1:raid51 /dev/sdf1
[root@mv1]# mdadm --detail /dev/md/mv1:raid51
```

(B) Continuaremos la práctica con la creación de los usuarios y la gestión de ACLs, según se

especifica en el enunciado del ejercicio práctico. Para la creación de los usuarios y grupos se hará uso de los comandos **useradd** y **groupadd**, siendo necesario usar **usermod** en el caso de que alguno de los usuarios ya existe previamente:

```
[root@mv1]# groupadd grupoftp1
[root@mv1]# groupadd grupoftp2
[root@mv1]# useradd -m -d /mnt/raid51/usuftp1 -s /bin/false -g grupoftp1 usuftp1
[root@mv1]# useradd -m -d /mnt/raid51/usuftp2 -s /bin/false -g grupoftp1 usuftp2
[root@mv1]# useradd -m -d /mnt/raid51/usuftp3 -s /bin/false -g grupoftp1 usuftp3
[root@mv1]# useradd -m -d /mnt/raid51/usuftp4 -s /bin/false -g grupoftp2 usuftp4
[root@mv1]# useradd -m -d /mnt/raid51/usuftp5 -s /bin/false -g grupoftp2 usuftp5
[root@mv1]# passwd usuftp1
[root@mv1]# passwd usuftp2
[root@mv1]# passwd usuftp3
[root@mv1]# passwd usuftp4
[root@mv1]# passwd usuftp5
```

En cuanto a las ACLs solicitadas en el enunciado, las estableceremos mediante la utilización del comando **setfacl**, comprobando su asignación mediante el comando **getfacl**. De igual forma, **getfacl** nos permitirá hacer un backup de las ACLs asignadas (*sobre los directorios HOME no es necesario asignar ACLs, ya que por defecto son propietarios de su HOME y ya tienen permisos rwx*):

```
[root@mv1]# mkdir /mnt/raid61/videos /mnt/raid61/musica /mnt/raid61/elibros
[root@mv1]# mkdir /mnt/raid61/documentacion /mnt/raid61/basura
[root@mv1]# mkdir /mnt/raid62/informes /mnt/raid62/facturas /mnt/raid62/nominas
[root@mv1]# setfacl -m u:usuftp1:rwx,u:usuftp2:rwx,u:usuftp3:r-x,g:grupoftp2:r-x \
             /mnt/raid61/videos /mnt/raid61/musica /mnt/raid61/elibros
[root@mv1]# setfacl -m g:grupoftp1:r--,g:grupoftp2:rwx \
             /mnt/raid61/documentacion /mnt/raid61/basura
[root@mv1]# setfacl -m g:grupoftp1:rwx,u:usuftp4:rwx,u:usuftp5:--- \
             /mnt/raid62/informes /mnt/raid62/facturas /mnt/raid62/nominas
```

```
[root@linux]# getfacl -Rp /mnt/raid51 | more
[root@linux]# getfacl -Rp /mnt/raid61 | more
[root@linux]# getfacl -Rp /mnt/raid62 | more
[root@linux]# getfacl -Rp /mnt/raid51 > /root/acl-raid51.bkp
[root@linux]# getfacl -Rp /mnt/raid61 > /root/acl-raid61.bkp
[root@linux]# getfacl -Rp /mnt/raid62 > /root/acl-raid62.bkp
```

Si quisiéramos eliminar todas las ACLs asignadas, y probar luego a restaurarlas:

```
[root@linux]# setfacl -Rb /mnt/raid51
[root@linux]# setfacl -Rb /mnt/raid61
[root@linux]# setfacl -Rb /mnt/raid62
```

Y por último, probaremos a restaurarlas, y y volver a mostrar:

```
[root@linux]# setfacl --restore=/root/acl-raid51.bkp
[root@linux]# setfacl --restore=/root/acl-raid61.bkp
[root@linux]# setfacl --restore=/root/acl-raid62.bkp
[root@linux]# getfacl -Rp /mnt/raid51 | more
[root@linux]# getfacl -Rp /mnt/raid61 | more
[root@linux]# getfacl -Rp /mnt/raid62 | more
```

(C) En relación a las cuotas especificadas en el enunciado, su asignación la haremos mediante el comando **quotatool**, pero antes deberemos activar los sistemas de **quota** en los volúmenes RAID recientemente creados. Para ello, en primer lugar desactivaremos todos los sistemas de cuota que pudieran estar activados en el equipo, **quotaoff -a**, para evitar conflictos, y después chequearemos y activaremos las cuotas en el RAID5 y los RAID6:

```
[root@mv1]# quotaoff -a
[root@mv1]# quotacheck -ugm /mnt/raid51
[root@mv1]# quotacheck -ugm /mnt/raid61
[root@mv1]# quotacheck -ugm /mnt/raid62
[root@mv1]# quotaon -ugv /mnt/raid51
[root@mv1]# quotaon -ugv /mnt/raid61
[root@mv1]# quotaon -ugv /mnt/raid62
[root@mv1]# quotatool -u usuftp1 -b -q 10MB -l 14MB -i -q 100 -l 110 /mnt/raid51
[root@mv1]# quotatool -u usuftp2 -b -q 10MB -l 14MB -i -q 100 -l 110 /mnt/raid51
[root@mv1]# quotatool -u usuftp3 -b -q 5MB -l 8MB -i -q 12 -l 22 /mnt/raid51
[root@mv1]# quotatool -u usuftp4 -b -q 5MB -l 8MB -i -q 12 -l 22 /mnt/raid51
[root@mv1]# quotatool -u usuftp5 -b -q 4MB -l 6MB -i -q 12 -l 22 /mnt/raid51
[root@mv1]# quotatool -u usuftp4 -b -q 18MB -l 20MB -i -q 120 -l 135 /mnt/raid61
[root@mv1]# quotatool -u usuftp5 -b -q 18MB -l 20MB -i -q 120 -l 135 /mnt/raid61
[root@mv1]# quotatool -u usuftp1 -b -q 30MB -l 31MB -i -q 600 -l 720 /mnt/raid62
[root@mv1]# quotatool -u usuftp4 -b -q 30MB -l 31MB -i -q 600 -l 720 /mnt/raid62
[root@mv1]# quotatool -u usuftp2 -b -q 15MB -l 18MB -i -q 180 -l 200 /mnt/raid62
[root@mv1]# quotatool -u usuftp3 -b -q 15MB -l 18MB -i -q 180 -l 200 /mnt/raid62
[root@mv1]# quotatool -u -b -i -t "2 days" /mnt/raid51
[root@mv1]# quotatool -u -b -i -t "2 days" /mnt/raid61
[root@mv1]# quotatool -u -b -i -t "2 days" /mnt/raid62
```

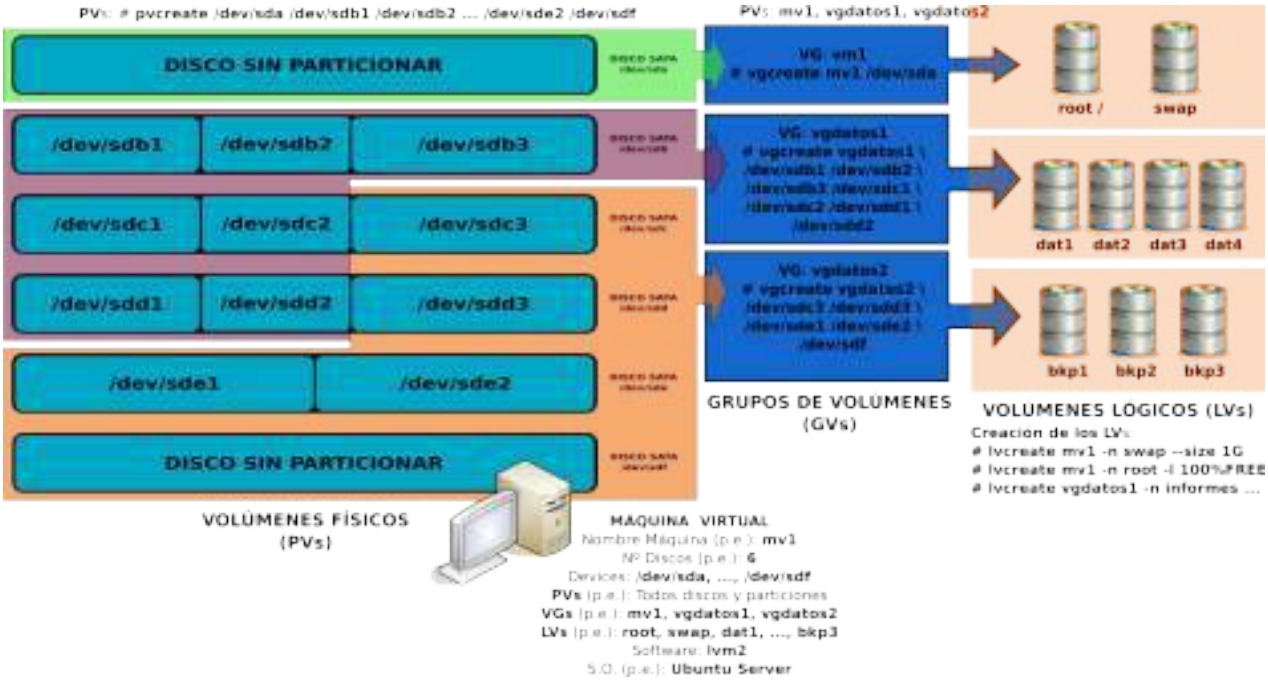
Por último, sería cuestión de probar tanto las ACLs como las cuotas asignadas mediante algún tipo de servicio (*p.e. proftpd, samba, etc.*) a través del cual accedieran los usuarios creados anteriormente. Para comprobar los permisos concedidos mediante las ACLs tan sólo será necesario intentar crear algún directorio o modificar el nombre del algún archivo. Para comprobar los límites de cuota deberemos subir algo al servidor e ir visualizando simultáneamente el informe de cuotas del punto de montaje, *p.e.* "**repquota -suv /mnt/raid61**", hasta comprobar que ya no nos deja almacenar más datos:

```
[root@mv1]# apt-get install proftpd
[root@mv1]# nano /etc/proftpd/proftpd.conf
# Incluir la siguiente configuración al comienzo del archivo de configuración proftpd.conf
```

```
<VirtualHost 192.168.1.1>
        Port 21001
        RequireValidShell off
        DefaultRoot ~
        <Limit LOGIN>
                AllowGroup grupoftp1 grupoftp2
                DenyAll
        </Limit>
</VirtualHost>
<VirtualHost 192.168.1.1>
        Port 21002
        RequireValidShell off
        DefaultRoot /mnt/raid61
        <Limit LOGIN>
                AllowGroup grupoftp1
                AllowUser usuftp4
                DenyAll
        </Limit>
</VirtualHost>
<VirtualHost 192.168.1.1>
        Port 21003
        RequireValidShell off
        DefaultRoot /mnt/raid62
        <Limit LOGIN>
                AllowGroup grupoftp2
                AllowUser usuftp1
                DenyAll
        </Limit>
</VirtualHost>
...
```

4.6.- Gestión de Volúmenes Lógicos Dinámicos: LVM

Una característica muy interesante que presentan las actuales distribuciones de GNU/Linux, es que al igual que Windows, se nos permite crear, gestionar y administrar volúmenes lógicos dinámicos mediante **LVM** (*Logical Volume Manager*). En concreto, actualmente cuando instalamos una distribución GNU/Linux (*p.e. Ubuntu Server o Debian*) se nos da la opción de instalar el sistema sobre un volumen lógico dinámico LVM (*todas las particiones /, /home, …, swap, a excepción de la /boot*). Esta es una característica muy interesante, ya que a diferencia de los clásicos particionados de disco presenta varias ventajas, entre las cuales podrían destacarse las siguientes:



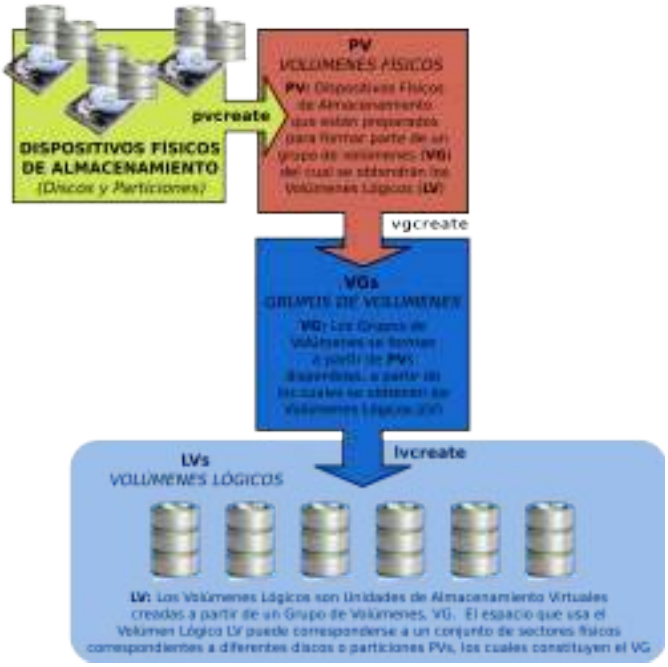
(1) Permite crear volúmenes del tamaño deseado a partir de discos o particiones variopintas de diferentes tamaños.
(2) Cuando se requiere más espacio en uno de los volúmenes creados, podemos añadir un nuevo disco o partición a la gestión LVM, y a posteriori extender o aumentar el tamaño del volumen sin que eso afecte a los datos que ya hay guardados en el volumen, y siendo totalmente transparente para el usuario final.
(3) Según lo anterior, nos permite redimensionar el volumen en caliente, sin necesidad de desmontarlo en el caso de que sea para aumentar el tamaño, permitiendo que los usuarios puedan estar trabajando con los datos que alberga el volumen lógico que se esta redimensionando, siendo esta operación totalmente transparente para ellos. Este aspecto es de los más importantes, si tenemos en cuenta los problemas que nos surgían cuando trabajábamos con las clásicas particiones. En el caso de que lo que queramos sea reducir el tamaño del volumen lógico, si que será necesario desmontar el volumen lógico para evitar la perdida de datos.
(4) Nos permite "particionar" o crear volúmenes lógicos a partir de un volumen RAID de una forma muy sencilla, versátil y eficiente.

(5) Nos permite asignar nombres más amigables e identificables a los dispositivos o volúmenes de almacenamiento. En lugar de /dev/sda1 o /dev/md126, podemos usar vol_facturas o vol_informes, o simplemente facturas e informes.

En relación a los inconvenientes que presenta la gestión LVM, destacar la fragmentación de los datos que provoca la implementación de un volumen formado por particiones no contiguas, discos o volúmenes RAID independientes.

En realidad se trata de una capa de abstracción por encima de los discos y particiones que tengamos, haciendo creer al usuario final que la estructura de sus sistemas de almacenamiento es completamente diferente y mucho más versátil. Por ejemplo, tal como se puede observar en la anterior figura, podríamos encontrarnos con un equipo que dispusiera de varios discos variopintos, particionados o no, de igual o diferente tamaño, y gracias a la gestión LVM, al final de cara al usuario tan sólo advertiría que de seis unidades o volúmenes lógicos de almacenamiento (*LVs*)

Además, para una correcta comprensión de como gestiona LVM el espacio disponible, sería conveniente antes entender el significado de los siguientes términos:



- **PVs**: Los Volúmenes Físicos (*Physical Volumes*) se corresponden con discos, particiones o volúmenes RAID que han sido preparados mediante la herramienta software **pvcreate** para ser gestionados mediante LVM. Para saber que discos, particiones o volúmenes RAID son aptos para ser transformados en **PVs** puede ejecutarse el comando **lvmscandisk**. Todas las operaciones que pueden realizarse sobre los **PVs** pueden consultarse listando los comandos disponibles en el sistema que comienzan por "pv*".
- **VGs**: Los **PVs** deben agruparse antes de pasar a formar Volúmenes Lógicos. El resultante de la agrupación da lugar a un Grupo de Volúmenes (**VG**). El tamaño resultante disponible en el Grupo de Volúmenes para la formación de Volúmenes Lógicos (**LV**), obviamente, es la suma de los espacios proporcionados por cada uno de los **PVs** que lo forman. Para la creación de un **VG** debe

hacerse uso del comando **vgcreate**. Cualquier otro tipo de operación a realizar sobre un **VG** puede consultarse a través del conjunto de comandos del sistema que comienzan por "vg*".
- **LVs**: Son los dispositivos (*devices*) o unidades de almacenamiento que el usuario puede formatear y montar como si se tratasen de una partición más. Estos Volúmenes Lógicos (**LVs**) deben crearse a partir del espacio disponible en un **VG** mediante el uso del comando del sistema lvcreate. Cualquier otro tipo de operación a realizar sobre un **LV** puede consultarse a través del conjunto de comandos del sistema que comienzan por "lv*".

4.6.1.- Comandos LVM para la gestión de volúmenes lógicos

Para poder comprobar las bondades de la gestión LVM será necesario tener instalado el software correspondiente **lvm2**:

```
[root@mv1]# apt-get install lvm2
```

Una forma de visualizar en pantalla todos los comandos existentes en el sistema para la gestión LVM junto con una breve descripción de su utilidad es ejecutar "**lvm**" y a continuación escribir "**help**":

```
[root@mv1]# lvm
lvm> help
  Available lvm commands:
  Use 'lvm help <command>' for more information

  dumpconfig      Dump active configuration
  formats         List available metadata formats
  help            Display help for commands
  lvchange        Change the attributes of logical volume(s)
  lvconvert       Change logical volume layout
  lvcreate        Create a logical volume
  lvdisplay       Display information about a logical volume
…
lvm> quit
  Exiting.
```

Tras observar la gran lista de comandos existentes, a continuación se detallarán los que se consideran más importantes en la gestión LVM:

Comando	Ejemplo / Descripción
lvmdiskscan	[root@mv1]# **lvmdiskscan**
	Lista todos los dispositivos físicos, particiones y discos, de nuestro equipo que podrían ser usados en la gestión **LVM**. Para que puedan formar parte de un volumen lógico, previamente han de convertirse en volúmenes físicos, **PV**.
pvcreate	[root@mv1]# **pvcreate /dev/sdc /dev/sdd1 /dev/md/mv1:raid51**
	Prepara un disco, partición o volumen RAID de nuestro sistema para ser

	gestionado mediante **LVM**, convirtiéndolos en un nuevo **PV** (*Volumen Físico*).
pvscan **pvs** **pvdisplay**	[root@mv1]# **pvscan | pvs | pvdisplay**
	Cualquiera de estos tres comandos nos listan, con mayor o menor detalle, los volúmenes físicos disponibles en nuestra máquina mediante los cuales podremos crear grupos de volúmenes lógicos, **VG**, y a partir de estos los volúmenes lógicos necesarios, **LV**.
pvremove	[root@mv1]# **pvremove /dev/sdd1**
	Deshace la creación de un volumen físico, **PV**.
vgcreate	[root@mv1]# **vgcreate nombre_grupo /dev/sdc /dev/md/mv1:raid51**
	Crea un grupo de volúmenes, **VG**, a partir de uno o varios volúmenes físicos, **PV**. El tamaño disponible dentro del **VG** para la creación de nuevos volúmenes lógicos, **LV**, será la suma de los espacios de los distintos volúmenes físicos, **PV**, que lo forman.
vgextend	[root@mv1]# **vgextend nombre_grupo /dev/sdd1 /dev/sde**
	Extiende o incrementa el número de volúmenes físicos, **PV**, que forman parte del grupo lógico indicado. Esto se traduce en un aumento del tamaño disponible en el grupo, lo que nos permitirá a posteriori asignar más espacio físico a los volúmenes lógicos, **LV**, que se han formado a partir de ese grupo lógico, **VG**.
vgscan **vgs** **vgdisplay**	[root@mv1]# **vgscan | vgs | vgdisplay**
	Cualquiera de estos tres comandos nos listan, con mayor o menor detalle, los grupos de volúmenes lógicos, **VG**, que ya han sido creados.
vgreduce **vgremove**	[root@mv1]# **vgreduce nombre_grupo /dev/sde** [root@mv1]# **vgextend nombre_grupo**
	Nos permiten reducir y eliminar un grupo de volúmenes lógicos.
lvcreate	[root@mv1]# **lvcreate** ***nombre_grupo*** **-n** ***nombre_volumen*** **--size** ***tamaño*** [root@mv1]# **lvcreate grupo1 -n informes --size 10G** [root@mv1]# **lvcreate grupo1 -n informes --extents 50%VG** [root@mv1]# **lvcreate grupo1 -n informes -l 50%VG** [root@mv1]# **lvcreate grupo1 -n informes --extents 70%FREE** [root@mv1]# **lvcreate grupo1 -n informes -l 70%FREE**
	Crea un nuevo volumen lógico dentro del grupo de volúmenes indicado, del tamaño que especifiquemos, **--size**. Además, tal como se muestra en los ejemplos, podemos indicar en lugar de un tamaño concreto, un porcentaje del tamaño total del grupo de volúmenes, o un porcentaje respecto a la cantidad de espacio libre que queda en el grupo de volúmenes (*opción* ***--extents*** *o* ***-l***).
lvscan **lvs** **lvdisplay**	[root@mv1]# **lvscan | lvs | lvdisplay**
	Cualquiera de estos tres comandos nos listan, con mayor o menor detalle, los grupos de volúmenes lógicos, **VG**, que ya han sido creados.
lvextend	[root@mv1]# **lvextend** */dev/nombre_grupo/nombre_volumen* **--size** ***tamaño*** [root@mv1]# **lvextend** */dev/nombre_grupo/nombre_volumen* **--size 20G**

	[root@mv1]# **lvextend** */dev/nombre_grupo/nombre_volumen* **--extents 8%FREE**
	Nos permite agrandar el tamaño de un volumen lógico sin necesidad de su desmontaje previo. Antes de agrandarlo deberemos comprobar que disponemos de suficiente espacio libre dentro del grupo de volúmenes al que pertenece. Tras el redimensionamiento será necesario reformatearlo mediante **resize2fs**, es decir, extenderemos el sistema de archivos sobre la parte del volumen extendida. [root@mv1]# **lvextend** */dev/nombre_grupo/nombre_volumen* **--size 20G** [root@mv1]# **resize2fs** */dev/nombre_grupo/nombre_volumen*
lvreduce **lvremove**	[root@mv1]# **lvreduce** */dev/nombre_grupo/nombre_volumen* **--size** ***tamaño*** [root@mv1]# **lvremove -f** /dev/*nombre_grupo/nombre_volumen*
	Nos permiten reducir y eliminar un volumen lógico. En el caso de querer reducir el tamaño de un volumen lógico es conveniente desmontarlo antes mediante **umount**, chequear el sistema de archivos mediante **e2fsck** y ajustar el sistema de archivos al nuevo tamaño del volumen mediante "**resize2fs** *nombre_volumen nuevo_tamaño*". [root@mv1]# **umount** */dev/nombre_grupo/nombre_volumen* [root@mv1]# **e2fsck -f** */dev/nombre_grupo/nombre_volumen* [root@mv1]# **resize2fs** */dev/nombre_grupo/nombre_volumen nuevo_tamaño*

Ej. Práctico 4.6.1: Gestión Básica de Volúmenes Lógicos LVM (I)

Agrega tres discos de 50G adicionales (*a parte del disco donde se encuentra instalado el sistema, /dev/sda*) a una máquina virtual que tengas en VirtualBox (*p.e. mv1*), p.e. bajo sistema operativo Ubuntu Server, e indica todos los comandos que ejecutarías para hacer la gestión de discos LVM que se especifica en la siguiente tabla. Deberás crear los PVs, VGs y LVs que se indican en la tabla, darles formato **ext4** y montarlos de manera automática en los directorios especificados (*no existen, deberás crearlos*) a través del archivo **/etc/fstab**.

VGs	Discos 50G - PVs	LVs	Punto Montaje
grupo1	**/dev/sdb** **/dev/sdc** **/dev/sdd**	**documentos** **75G**	**/mnt/documentos**
		imagenes *(espacio restante)*	**/mnt/imagenes**

Solución Ej. Pr. 4.6.1.I.- Creación de LVs a partir de Varios Discos Físicos

Teniendo en cuenta toda la teoría previa, para dar solución al ejercicio práctico planteado se irán detallando a continuación todos los pasos necesarios:

(1) Convertimos mediante **pvcreate** los tres discos físicos agregados a la máquina virtual en volúmenes físicos (PVs) aptos para la gestión LVM. Antes sería conveniente comprobar que esos discos aparecen en la lista de dispositivos mostrada por **lvmdiskscan**, lo cual nos indica que pueden ser usados para dicha gestión, y que la conversión no tendría porque dar ningún tipo de problemas.

```
[usuario@mv1]$ sudo su
[root@mv1]# lvmdiskscan
[root@mv1]# pvcreate /dev/sdb /dev/sdc /dev/sdd
[root@mv1]# pvs
```

(2) A continuación agruparemos los PVs en un único grupo de volúmenes (VG), con un tamaño resultante suma de los tres PVs (*≈150G*), del cual extraeremos a posteriori los volúmenes lógicos:

```
[root@mv1]# vgcreate grupo1 /dev/sdb /dev/sdc /dev/sdd
[root@mv1]# vgs
[root@mv1]# lvcreate grupo1 -n documentos --size 75G
[root@mv1]# lvcreate grupo1 -n imagenes -l 100%FREE
[root@mv1]# lvs
```

¡¡Observación!! Tal como se puede observar a través de los comandos anteriores, cuando queremos repartir el espacio total de un grupo de volúmenes entre varios volúmenes lógicos, con la finalidad de aprovechar al máximo todo su espacio, al último de ellos no se le asignará el espacio que matemáticamente nos reste (*no sirve hacer cuentas redondas*), sino el 100% del espacio libre disponible. Es decir, si el tamaño total de la agrupación de PVs es de 150G, y vamos a crear dos

LVs, si al primero le asignamos 75G, eso no quiere decir que queden otros 75G para el otro LV, ya que el sistema al crear un LV redondea de la forma que él considera más conveniente, según la estructura de los discos.

(3) Por último, daremos formato a los volúmenes anteriores, configuraremos el archivo **/etc/fstab** y los montaremos. Para comprobar su persistencia será conveniente reiniciar el equipo (*init 6*).

```
[root@mv1]# mkfs.ext4 /dev/grupo1/documentos
[root@mv1]# mkfs.ext4 /dev/grupo1/imagenes
[root@mv1]# mkdir /mnt/documentos /mnt/imagenes
[root@mv1]# nano /etc/fstab
…
# Añadiremos las siguientes líneas:
/dev/grupo1/documentos    /mnt/documentos    ext4    defaults 0 0
/dev/grupo1/imagenes      /mnt/imagenes              ext4    defaults 0 0
```

```
[root@mv1]# mount /mnt/documentos
[root@mv1]# mount /mnt/imagenes
[root@mv1]# df -h
[root@mv1]# init 6
```

Ej. Práctico 4.6.2: Gestión Básica de Volúmenes Lógicos LVM (II)

Si en el ejercicio práctico anterior hemos aprendido como crear volúmenes lógicos a partir de discos o particiones físicas, en el presente ejercicio resaltaremos una de las ventajas más interesantes que presenta la gestión LVM: la facilidad para reducir y aumentar el tamaño de los volúmenes lógicos.

Para ello llevaremos a cabo las siguientes acciones:

(1) Agrega un nuevo disco de 50G a la máquina virtual, **/dev/sde**, y agrégalo al grupo de volúmenes anterior **VG grupo1**.

(2) Con el nuevo espacio disponible en el grupo agranda el **LV documentos** en 30G (*75G + 30G = 105G*).

(3) Reduce el tamaño del **LV imagenes** en unos 10G y comprueba que el espacio disponible en el grupo aumenta en la misma relación (*≈75G – 10G ≈ 65G*).

(4) Vuelve a aumentar el tamaño del LV documentos en otros 25G. Advierte que mientras queda espacio disponible en el grupo VG grupo1 puede asignarse éste de una manera muy sencilla a cualquiera de los LVs que lo forman.

Solución Ej. Pr. 4.6.2.I.- *Reducir y Aumentar el tamaño de LVs*

Para dar solución al ejercicio práctico planteado se irán detallando a continuación todos los pasos necesarios:

(1) Una vez agregado el nuevo disco **/dev/sde** a la máquina virtual, lo convertiremos en un nuevo PV mediante **pvcreate**. Después extenderemos la agrupación **VG grupo1** mediante el nuevo **PV /dev/sde**.

```
[usuario@mv1]$ sudo su
[root@mv1]# lvmdiskscan | grep /dev/sde
[root@mv1]# pvcreate /dev/sde
[root@mv1]# pvs
[root@mv1]# vgextend grupo1 /dev/sde
[root@mv1]# vgs
```

(2) A continuación agrandaremos mediante el comando **lvextend/lvresize** el **LV documentos** en 30G. Una vez agrandado el espacio disponible por el volumen, será necesario extender igualmente el sistema de archivos mediante el comando resize2fs (*puede advertirse ejecutando "df -h" que una cosa es el espacio del volumen, y otra es el espacio del volumen que tiene formato, y que por tanto, puede ser usado para almacenar información*).

```
[root@mv1]# lvextend /dev/grupo1/documentos --size +30G
[root@mv1]# lvs
[root@mv1]# vgs
[root@mv1]# df -h
[root@mv1]# resize2fs /dev/grupo1/documentos
[root@mv1]# df -h
```

(3) Ahora reduciremos el tamaño del **LV imagenes** en 10G (*≈75G – 10G ≈ 65G*). El tamaño que libere lo recibirá el **VG grupo1**, el cual podrá asignarse a posteriori a cualquier LV.

```
[root@mv1]# umount /dev/grupo1/imagenes
[root@mv1]# e2fsck -f /dev/grupo1/imagenes
[root@mv1]# resize2fs /dev/grupo1/imagenes 65G
[root@mv1]# lvreduce /dev/grupo1/imagenes --size 65G
[root@mv1]# lvs
[root@mv1]# vgs
```

(4) Por último, volveremos a aumentar el tamaño del **LV documentos** en otros 25G, gracias al espacio aportado por la reducción del volumen anterior.

```
[root@mv1]# lvextend /dev/grupo1/documentos --size +25G
[root@mv1]# lvs
[root@mv1]# vgs
[root@mv1]# df -h
[root@mv1]# resize2fs /dev/grupo1/documentos
[root@mv1]# df -h
```

¡¡Observación!! Tras realizar el ejercicio práctico anterior, deberíamos haber advertido los siguientes aspectos a tener en cuenta:

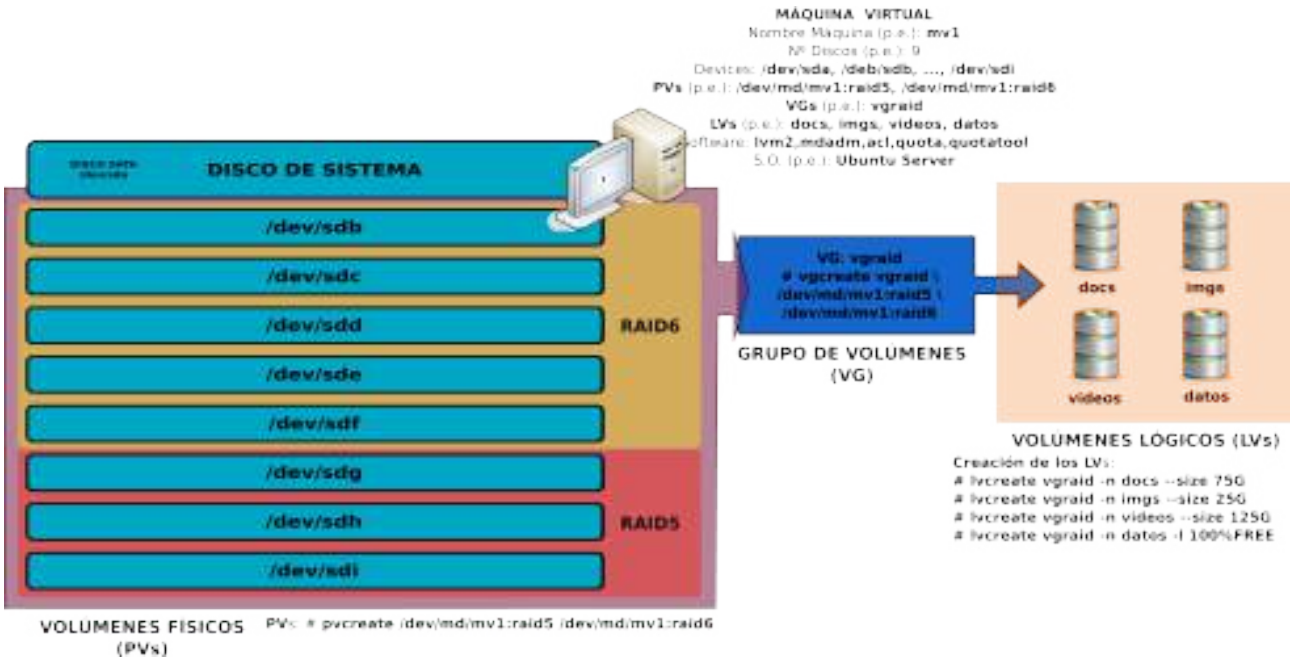
(I) Para reducir un LV es necesario desmontarlo previamente (*umount*).
(II) Una vez desmontado el LV a reducir deberá reducirse primero su sistema de archivos (*resize2fs*), y posteriormente el tamaño del volumen (*lvreduce*). En caso contrario los índices del sistema de archivos podrían apuntar a direcciones no existentes creando inconsistencias. Además

antes de reducir el tamaño del sistema de archivos conviene forzar un chequeo (*e2fsck -f*).
(III) Para aumentar el tamaño de un LV no es necesario su desmontaje, puede hacerse en caliente (*lvextend*).
(IV) Una vez aumentado el tamaño del LV será necesario aumentar el tamaño de su sistema de archivos, ya que de poco sirve aumentar el tamaño del LV si éste no tiene formato. Puede comprobarse mediante un "df -h" que hasta que no se redimensiona el sistema de archivos, para el sistema, el tamaño del volumen lógico no varía.

Ej. Práctico 4.6.3: Gestión LVM desde un RAID5 y un RAID6

Con la finalidad de repasar todo lo visto, implementa todo lo que se indica en la siguiente tabla sobre una máquina virtual (*p.e. mv1 – Ubuntu Server 12.04*): **(A) Volúmenes RAID, (B) Gestión LVM, (C) ACLs** y **(D) Cuotas**.

RAID	Discos 50G	VG	LVs	Punto Montaje	ACLs	Cuotas (b/i)
RAID6	**/dev/sdb /dev/sdc /dev/sdd /dev/sde /dev/sdf**	**vgraid**	**docs 75G**	**/mnt/docs**	usu1:rwx usu2:r-x usu3:rwx	usu1:50G/80000 usu2:- usu3:25G/20000
			imgs 25G	**/mnt/imgs**	usu1:rwx usu2:--- usu3:r-x	usu1:15G/40000 usu2:- usu3:-
RAID5	**/dev/sdg /dev/sdh /dev/sdi**		**videos 125G**	**/mnt/vid**	usu1:rwx usu2:rwx usu3:rwx	usu1:25G/4000 usu2:50G/10000 usu3:50G/10000
			datos 100%FREE	**/mnt/datos**	usu1:rwx usu2:rwx usu3:---	usu1:10G/60000 usu2:10G/60000 usu3:-



Solución Ej. Pr. 4.6.3.I.- Cómo Crear LVs desde un RAID5 y un RAID6

Para dar solución al ejercicio práctico planteado a continuación se irán detallando todos los pasos necesarios:

(1) En primer lugar será necesario agregar al equipo con el que se va a realizar la practica (*p.e. máquina virutal Ubuntu Server*) los 8 discos con los que se implementarán los volúmenes RAID. De esta forma, la máquina dispondrá de 9 discos (*el disco donde se encuentra instalado el sistema más los 8 discos anteriores*).

(2) En segundo lugar comprobaremos que nuestro equipo dispone de todo el software necesario para la realización de la práctica:

(a) **mdadm**: necesario para la creación, gestión y mantenimiento de los volúmenes RAID.
(b) **lvm2**: necesario para la gestión de los volúmenes lógicos. Nos permitirá crear los volúmenes físicos PVs, agruparlos en un grupo de volúmenes VG y esta agrupación dividirla en volúmenes lógicos dinámicos LVs.
(c) **acl**: necesario para la administración de las listas de control de acceso.
(d) **quota** y **quotatool**: necesarios para la asignación de cuotas de espacio en disco a los usuarios.

Para su instalación ejecutaríamos el siguiente comando:

```
[root@mv1]# apt-get install  mdadm lvm2 acl quota quotatool
```

Otra opción hubiera sido crear un pequeño script que compruebe que software tiene ya instalado la máquina y en caso de ser necesario instalarlo:

```
[root@mv1]# nano /root/comprobacion-software.sh
#!/bin/bash
clear
echo "Comprobamos si tenemos instalado el Software necesario ..."
PROGRAMAS="mdadm lvm2 acl quota quotatool"
for PROGRAMA in $PROGRAMAS
do
        if test $(dpkg --get-selections | grep "^$PROGRAMA" | wc -l) -eq 0
        then
                echo "No dispones del software $PROGRAMA. Pulsa una tecla para instalarlo ..."
                apt-get install $PROGRAMA
        else
                echo "Ok!! Ya tienes instalado $PROGRAMA. Pulsa una tecla para continuar"
                read
        fi
done
```

```
[root@mv1]# chmod +x /root/comprobacion-software.sh
[root@mv1]# /root/comprobacion-software.sh
```

(3) En tercer lugar, comprobaremos que el sistema reconoce los nueve discos y en caso afirmativo crearemos los volúmenes RAID5 y RAID6 indicados en el enunciado de la práctica. Tras la creación de los volúmenes RAID es aconsejable reiniciar la máquina y comprobar que los volúmenes RAID persisten.

```
[root@mv1]# nano /root/raid.sh
echo "Listamos los discos y creamos los Volúmenes RAID ..."
NUMDISCOS=$(ls -l /dev/sd? | wc -l)
if test $NUMDISCOS -eq 9
then
        if test ! -d /dev/md
        then
                mkdir /dev/md
        fi
        mdadm --create /dev/md/mv1:raid6 --level=raid6 --raid-devices=5  \
                /dev/sdb /dev/sdc /dev/sdd /dev/sde /dev/sdf
        mdadm --create /dev/md/mv1:raid5 --level=raid5 --raid-devices=3  \
                /dev/sdg /dev/sdh /dev/sdi
        echo "DEVICE /dev/sdb /dev/sdc /dev/sdd /dev/sde /dev/sdf \
                /dev/sdg /dev/sdh /dev/sdi"  >>  /etc/mdadm/mdadm.conf
        mdadm --detail --scan | tail -2 >>  /etc/mdadm/mdadm.conf
        clear
        echo "Listamos los RAID creados. Pulsa una tecla para reiniciar y comprobar ..."
        ls -l /dev/md/
        read
        init 6
else
        echo "No se han detectado todos los discos necesarios para la práctica ..."
fi
```

```
[root@mv1]# chmod +x /root/raid.sh
[root@mv1]# /root/raid.sh
```

¡¡Recordatorio!! Recalcar que **/dev/md/mv1:raid5** y **/dev/md/mv1:raid6** no son los *devices* o dispositivos RAID creados, sino unos enlaces o accesos directos a ellos. Para conocer el nombre asignado por **mdadm** a los volúmenes RAID podemos comprobar a que dispositivo enlazan ejecutando "**ls -l /dev/md/**". Por tanto, **/dev/md/mv1:raid5** y **/dev/md/mv1:raid6** no son más que unos enlaces con unos nombres mucho más amigables para el usuario que nos permiten hacer referencia al dispositivo RAID en su futura gestión y mantenimiento. Dicho nombre esta compuesto por el "*nombre_maquina:nombre_volumen_RAID*".

(4) Tras crear los volúmenes RAID pasaremos a la gestión LVM: creación de los PVs, agrupación de estos en un único VG, y división en LVs. Tras la creación de los volúmenes lógicos les daremos formato (*p.e. ext4*) y los montaremos vía **/etc/fstab**.

```
[root@mv1]# nano /root/gestion-lvm.sh
clear
echo "Comprobamos que lvm2 reconoce los RAID como dispositivos aptos para la gestión LVM"
lvmdiskscan | grep /dev/md
read
echo "Creamos los PVs, VG y LVs ..."
pvcreate /dev/md/mv1:raid5 /dev/md/mv1:raid6
```

```
vgcreate vgraid /dev/md/mv1:raid5 /dev/md/mv1:raid6
lvcreate vgraid -n docs --size 75G
lvcreate vgraid -n imgs --size 25G
lvcreate vgraid -n videos --size 125G
lvcreate vgraid -n datos -l 100%FREE
lvs
echo "Pasamos a dar formato y montarlos automáticamente mediante fstab ..."
mkfs.ext4 /dev/vgraid/docs
mkfs.ext4 /dev/vgraid/imgs
mkfs.ext4 /dev/vgraid/videos
mkfs.ext4 /dev/vgraid/datos
mkdir /mnt/docs /mnt/imgs /mnt/vid /mnt/datos
echo "/dev/vgraid/docs /mnt/docs ext4 defaults,exec,rw,acl,usrquota 0 0" >> /etc/fstab
echo "/dev/vgraid/imgs /mnt/imgs ext4 defaults,exec,rw,acl,usrquota 0 0" >> /etc/fstab
echo "/dev/vgraid/videos /mnt/vid ext4 defaults,exec,rw,acl,usrquota 0 0" >> /etc/fstab
echo "/dev/vgraid/datos /mnt/datos ext4 defaults,exec,rw,acl,usrquota 0 0" >> /etc/fstab
mount /mnt/docs
mount /mnt/imgs
mount /mnt/vid
mount /mnt/datos
clear
echo "Las unidades de almacenamiento montadas tras toda la configuración son:"
echo ""
df -h
```

```
[root@mv1]# chmod +x /root/gestion-lvm.sh
[root@mv1]# /root/gestion-lvm.sh
```

Convendría reiniciar la máquina y comprobar que el montaje de los volúmenes lógicos persiste:

```
[root@mv1]# init 6
```

(5) Por último asignaremos las ACLs y Cuotas a los usuarios indicados en el enunciado sobre los volúmenes lógicos recientemente creados. Antes será necesario crear las cuentas de usuario.

```
[root@mv1]# nano /root/gestion-acl-cuotas.sh
clear
echo "Ahora pasaremos a asignar las ACLs y Cuotas indicadas en la tabla. Pulsa una tecla ..."
read
echo "Creamos las cuentas de usuario del sistema: usu1, usu2 y usu3"
useradd -m -d /home/usu1 -s /bin/bash usu1
echo "Password para el usuario usu1: "
passwd usu1
useradd -m -d /home/usu2 -s /bin/bash usu2
echo "Password para el usuario usu2: "
passwd usu2
```

```
useradd -m -d /home/usu3 -s /bin/bash usu3
echo "Password para el usuario usu3: "
passwd usu3
echo "Asignamos ACLs a los usuarios anteriores sobre los LVs creados ..."
setfacl -m u:usu1:rwx,u:usu2:r-x,u:usu3:rwx  /mnt/docs
setfacl -m u:usu1:rwx,u:usu3:r-x  /mnt/imgs
setfacl -m u:usu1:rwx,u:usu2:rwx,u:usu3:rwx  /mnt/vid
setfacl -m u:usu1:rwx,u:usu2:rwx  /mnt/datos
echo "Ahora Chequeamos, Activamos y Asignamos las Cuotas a los usuarios sobre los LVs ..."
quotaoff -a
quotacheck -um /mnt/docs
quotacheck -um /mnt/imgs
quotacheck -um /mnt/vid
quotacheck -um /mnt/datos
quotaon -uv /mnt/docs
quotaon -uv /mnt/imgs
quotaon -uv /mnt/vid
quotaon -uv /mnt/datos
echo "Establecemos Cuotas para el usuario usu1 ..."
quotatool -u usu1 -b -l 50G -i -l 80000 /mnt/docs
quotatool -u usu1 -b -l 15G -i -l 40000 /mnt/imgs
quotatool -u usu1 -b -l 25G -i -l 4000 /mnt/vid
quotatool -u usu1 -b -l 10G -i -l 60000 /mnt/datos
echo "Establecemos Cuotas para el usuario usu2 ..."
quotatool -u usu2 -b -l 50G -i -l 10000 /mnt/vid
quotatool -u usu2 -b -l 10G -i -l 60000 /mnt/datos
echo "Establecemos Cuotas para el usuario usu3 ..."
quotatool -u usu3 -b -l 25G -i -l 20000 /mnt/docs
quotatool -u usu3 -b -l 50G -i -l 1000 /mnt/vid
echo ""
echo "Ahora tan sólo cabe reiniciar y comprobar que los LVs persisten. Pulsa una tecla ..."
read
init 6
```

```
[root@mv1]# chmod +x /root/gestion-acl-cuotas.sh
[root@mv1]# /root/gestion-acl-cuotas.sh
```

Una vez que ya tenemos todo configurado podríamos ejecutar el siguiente script para comprobar que ha surtido efecto todo lo anterior:

```
[root@mv1]# nano /root/script-comprobacion.sh
#!/bin/bash
clear
echo "Comprobamos que los Volúmenes RAID persisten …"
echo ""
mdadm --detail --scan
read
```

```
clear
echo "Comprobamos que los LVs se han automontado correctamente …"
echo ""
df -h
read
clear
echo "Comprobamos que los ACLs se han asignado correctamente a cada LV …"
echo ""
getfacl -p /mnt/docs
getfacl -p /mnt/imgs
getfacl -p /mnt/vid
getfacl -p /mnt/datos
read
clear
echo "Mostramos y comprobamos el informe de Cuotas asignadas a cada LV …"
echo ""
repquota -suv /mnt/docs
repquota -suv /mnt/imgs
repquota -suv /mnt/vid
repquota -suv /mnt/datos
```

```
[root@mv1]# chmod +x /root/script-comprobacion.sh
[root@mv1]# /root/script-comprobacion.sh
```

Ej. Práctico 4.6.4: Redimensionamiento de LVs

Siguiendo con el ejercicio práctico anterior: Si quisiéramos extender el espacio disponible en el volumen lógico "**datos**" en 5G, quitándoselos previamente al volumen "**videos**", ¿Cuáles serían los pasos de configuración a seguir?

Solución Ej. Pr. 4.6.4.I.- *Cómo Redimensionar LVs*

Para resolver el presente ejercicio práctico deberemos tener en cuenta los siguientes aspectos:

(I) Para reducir un LV es necesario desmontarlo previamente (*umount*).
(II) Una vez desmontado el LV a reducir deberá reducirse primero su sistema de archivos (*resize2fs*), y posteriormente el tamaño del volumen (*lvreduce*). En caso contrario los índices del sistema de archivos podrían apuntar a direcciones no existentes creando inconsistencias. Además antes de reducir el tamaño del sistema de archivos conviene forzar un chequeo (*e2fsck -f*).
(III) Para aumentar el tamaño de un LV no es necesario su desmontaje, puede hacerse en caliente (*lvextend*).
(IV) Una vez aumentado el tamaño del LV será necesario aumentar el tamaño de su sistema de archivos, ya que de poco sirve aumentar el tamaño del LV si éste no tiene formato. Puede comprobarse mediante un "**df -h**" que hasta que no se redimensiona el sistema de archivos, para el sistema, el tamaño del volumen lógico no varía.
(V) Si tenemos en cuenta que el tamaño del LV **/dev/vgraid/videos** creado en el ejercicio práctico anterior tenía un tamaño de 125G, y queremos quitarle 5G para asignárselos a otro LV, su nuevo tamaño será de 120G.

Teniendo en cuenta todas las premisas anteriores, la lista de comandos que serían necesarios ejecutar serían los siguientes:

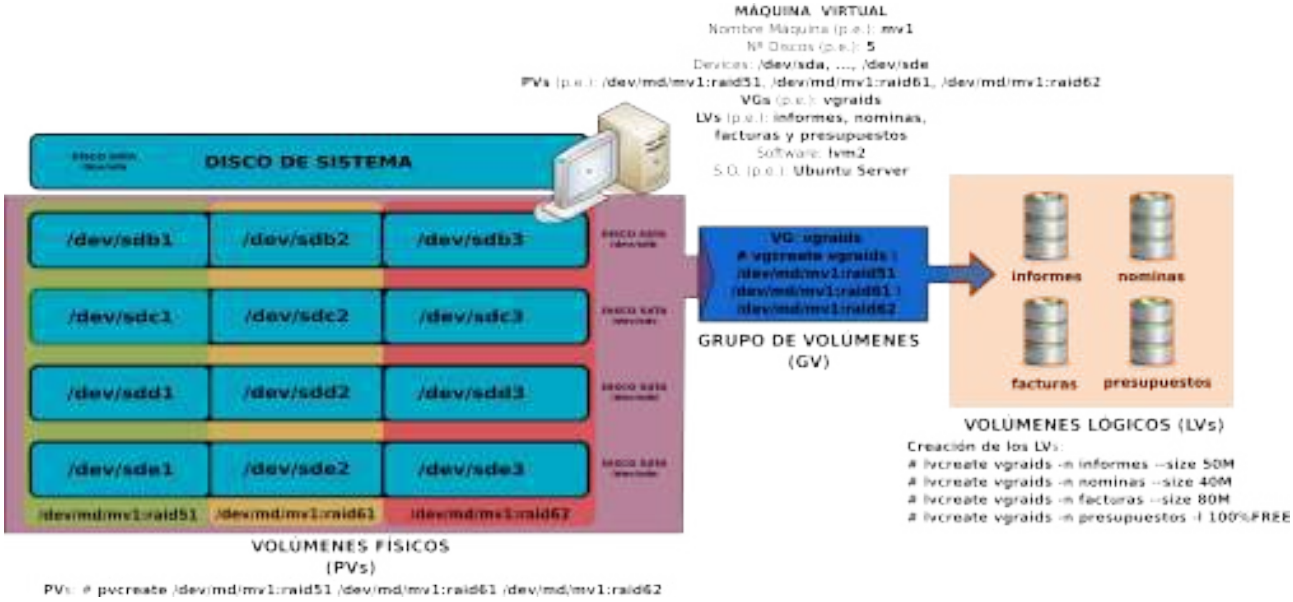
```
[root@mv1]# nano /root/script-lvm-redimensionar.sh
umount /mnt/vid
e2fsck -f /dev/vgraid/videos
resize2fs /dev/vgraid/videos 120G
lvreduce /dev/vgraid/videos --size 120G
lvs
mount /mnt/vid
lvextend /dev/vgraid/datos --size +5G
resize2fs  /dev/vgraid/datos
clear
echo "Las unidades de almacenamiento montadas tras el redimensionamiento son:"
echo ""
df -h
```

Ej. Práctico 4.6.5: Gestión LVM desde tres RAID creados mediante Particiones

Observa la figura siguiente, crea los tres volúmenes RAID que se especifican y haz la gestión LVM que se indica en la tabla (*repasar los ejercicios prácticos anteriores*). En concreto, deberás crear un RAID5 y dos RAID6 mediante las particiones de 4 discos. Conviértelos en volúmenes físicos, PV, aptos para ser gestionados mediante LVM. Después crea un grupo de volúmenes lógico llamado **vgraids** con ellos, a partir del cual se formarán los siguientes volúmenes lógicos: **informes**, **nominas**, **facturas** y **presupuestos**.

Volúmenes RAID	4 Particiones que forman los RAID
RAID5 /dev/md/mv1:raid51	120MB (p.e. /dev/sdb1, ..., /dev/sde1)
RAID6 /dev/md/mv1:raid61	130MB (p.e. /dev/sdb2, ..., /dev/sde2)
RAID6 /dev/md/mv1:raid62	100MB (p.e. /dev/sdb3, ..., /dev/sde3)

Volúmenes Físicos PV	Grupo de Volúmenes Lógico VG	Volúmenes Lógicos LV	Tamaño	Punto de Montaje
/dev/md/mv1:raid51 **/dev/md/mv1:raid61** **/dev/md/mv1:raid62**	**vgraids**	**informes**	**50M**	**/mnt/informes**
		nominas	**40M**	**/mnt/nominas**
		facturas	**80M**	**/mnt/facturas**
		presupuestos	**100%FREE**	**/mnt/presupuestos**



Solución Ej. Pr. 4.6.5.I.- Gestión LVM a partir de dos RAID6

Para la realización del ejercicio práctico propuesto seguiremos los siguientes pasos:

(1) Comenzaremos implementado los volúmenes RAID5 y RAID6. Antes deberemos agregar los cuatro discos a la máquina (*p.e. máquina virtual Ubuntu Server*) y posteriormente particionarlos (*particionaremos uno y clonaremos su estructura mediente sfdisk*).

Es decir, una vez agregados los 4 discos a la máquina (*p.e. /dev/sdb, …, /dev/sde*), comenzaremos la práctica particionando uno de los discos mediante **fdisk**, creando tres particiones primarias de 120MB, 130MB y 100MB respectivamente, para después hacer una copia del particionado sobre el resto mediante **sfdisk**:

```
[root@mv1]# fdisk /dev/sdb
Command (m for help): n
Partition type:
  p   primary (0 primary, 0 extended, 4 free)
  e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-716799, default 2048): 2048
Last sector, +sectors or +size{K,M,G} (2048-716799, default 716799): +120MB

Command (m for help): n
Partition type:
  p   primary (1 primary, 0 extended, 3 free)
  e   extended
Select (default p): p
Partition number (1-4, default 2): 2
First sector (236423-716799, default 236423): 236423
Last sector, +sectors or +size{K,M,G} (236423-716799, default 716799): +130MB

Command (m for help): n
Partition type:
  p   primary (2 primary, 0 extended, 2 free)
  e   extended
Select (default p): p
Partition number (1-4, default 3): 3
First sector (490329-716799, default 490329): 490329
Last sector, +sectors or +size{K,M,G} (490329-716799, default 716799): 716799

Command (m for help): p
Disk /dev/sdq: 367 MB, 367001600 bytes
255 heads, 63 sectors/track, 44 cylinders, total 716800 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
Device Boot      Start         End      Blocks   Id  System
/dev/sdq1            2048      236422      117187+  83  Linux
/dev/sdq2          236423      490328      126953   83  Linux
/dev/sdq3          490329      716799      113235+  83  Linux
```

```
Command (m for help): w
[root@mv1]# sfdisk -d /dev/sdb | sfdisk /dev/sdc --force
[root@mv1]# sfdisk -d /dev/sdb | sfdisk /dev/sdd --force
[root@mv1]# sfdisk -d /dev/sdb | sfdisk /dev/sde --force
```

Después, antes de empezar a crear los volúmenes RAID, comprobaremos que los discos están correctamente particionados listando los dispositivos reconocidos por la máquina:

```
[root@mv1]# ls /dev/sd*
/dev/sda    /dev/sdb    /dev/sdc    /dev/sdd    /dev/sde
/dev/sda1   /dev/sdb1   /dev/sdc1   /dev/sdd1   /dev/sde1
/dev/sda2   /dev/sdb2   /dev/sdc2   /dev/sdd2   /dev/sde2
/dev/sda5   /dev/sdb3   /dev/sdc3   /dev/sdd3   /dev/sde3
```

Ahora pasaremos a crear los volúmenes RAID: un RAID5 con las 4 primeras particiones de los cuatro discos agregados, y dos RAID6 con el resto de particiones, tal como se especifica en la tabla del enunciado:

```
[root@mv1]# mkdir /dev/md
[root@mv1]# mdadm --create /dev/md/mv1:raid51 --level=raid5 \
--raid-devices=4 /dev/sdb1 /dev/sdc1 /dev/sdd1 /dev/sde1
[root@mv1]# mdadm --create /dev/md/mv1:raid61 --level=raid6 \
--raid-devices=4 /dev/sdb2 /dev/sdc2 /dev/sdd2 /dev/sde2
[root@mv1]# mdadm --create /dev/md/mv1:raid62 --level=raid6 \
--raid-devices=4 /dev/sdb3 /dev/sdc3 /dev/sdd3 /dev/sde3
[root@mv1]# echo "DEVICE /dev/sd*" >> /etc/mdadm/mdadm.conf
[root@mv1]# mdadm --detail --scan | tail -3 >> /etc/mdadm/mdadm.conf
```

Antes de pasar al siguiente paso sería recomendable reiniciar la máquina y comprobar que los volúmenes RAID persisten.

```
[root@mv1]# init 6
[root@mv1]# ls -l /dev/md/
```

(2) Tras crear los RAID, haciendo uso de los comandos LVM descritos anteriormente, (a) comenzaremos listando con **lvmdiskscan** cuales son las unidades de almacenamiento de nuestro sistema que son aptas para ser convertidas en volúmenes físicos **PV**, y así comprobar que en dicha lista se encuentran nuestros volúmenes RAID, (b) crearemos los volúmenes físicos **PV** mediante **pvcreate**, (c) crearemos un grupo de volúmenes lógicos **VG** a partir de los **PV** anteriores mediante **vgcreate**, y (d) por último crearemos las unidades lógicas mediante **lvcreate**:

```
[root@mv1]# lvmdiskscan
…
        /dev/md125 [     358,50 MiB]
        /dev/md126 [     259,00 MiB]
        /dev/md127 [     197,00 MiB]
```

```
...
```

Para identificar que dispositivos RAID son los listados anteriormente, podemos ver a donde enlazan los links de los dispositivos RAID creados:

```
[root@mv1]# ls -l /dev/md/
...
lrwxrwxrwx 1 root root 8 ene 17 15:40 mv1:raid51 -> ../md125
lrwxrwxrwx 1 root root 8 ene 17 15:40 mv1:raid61 -> ../md126
lrwxrwxrwx 1 root root 8 ene 17 15:40 mv1:raid62 -> ../md127
```

```
[root@mv1]# pvcreate /dev/md125 /dev/md126 /dev/md127
  Writing physical volume data to disk "/dev/md125"
  Physical volume "/dev/md125" successfully created
  Writing physical volume data to disk "/dev/md126"
  Physical volume "/dev/md126" successfully created
  Writing physical volume data to disk "/dev/md127"
  Physical volume "/dev/md127" successfully created
```

```
[root@mv1]# vgcreate vgraids /dev/md125 /dev/md126 /dev/md127
  Volume group "vgraids" successfully created
```

```
[root@mv1]# lvcreate vgraids -n informes --size 50M
[root@mv1]# lvcreate vgraids -n nominas --size 40M
[root@mv1]# lvcreate vgraids -n facturas --size 80M
[root@mv1]# lvcreate vgraids -n presupuestos -l 100%FREE
[root@mv1]# lvscan
  ACTIVE            '/dev/vgraids/informes' [52,00 MiB] inherit
  ACTIVE            '/dev/vgraids/nominas' [40,00 MiB] inherit
  ACTIVE            '/dev/vgraids/facturas' [80,00 MiB] inherit
  ACTIVE            '/dev/vgraids/presupuestos' [280,00 MiB] inherit
```

Por último, formatearemos y montaremos los volúmenes lógicos anteriores:

```
[root@mv1]# mkfs.ext4 /dev/vgraids/informes
[root@mv1]# mkfs.ext4 /dev/vgraids/nominas
[root@mv1]# mkfs.ext4 /dev/vgraids/facturas
[root@mv1]# mkfs.ext4 /dev/vgraids/presupuestos
[root@mv1]# mdkir /mnt/informes /mnt/nominas /mnt/facturas /mnt/presupuestos
[root@mv1]# nano /etc/fstab
...
/dev/vgraids/informes /mnt/informes ext4 defaults,acl,usrquota,grpquota 0 0
/dev/vgraids/nominas /mnt/nominas ext4 defaults,acl,usrquota,grpquota 0 0
/dev/vgraids/facturas /mnt/facturas ext4 defaults,acl,usrquota,grpquota 0 0
/dev/vgraids/presupuestos /mnt/presupuestos ext4 defaults,acl,usrquota,grpquota 0 0
```

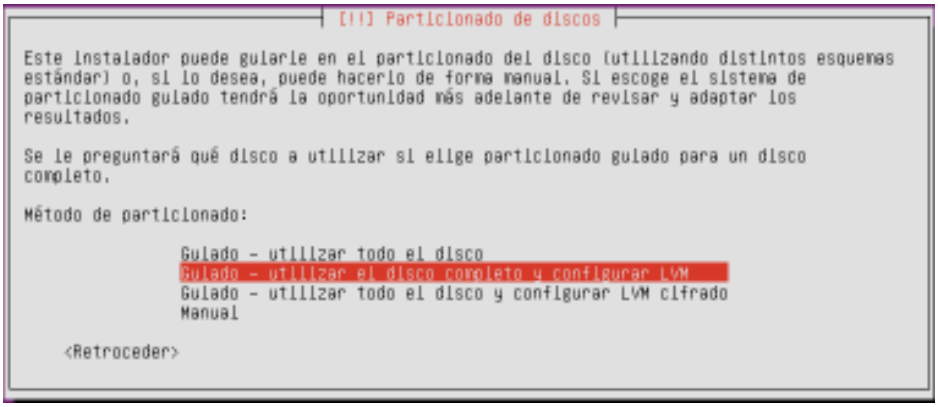
```
[root@mv1]# mount /mnt/informes
```

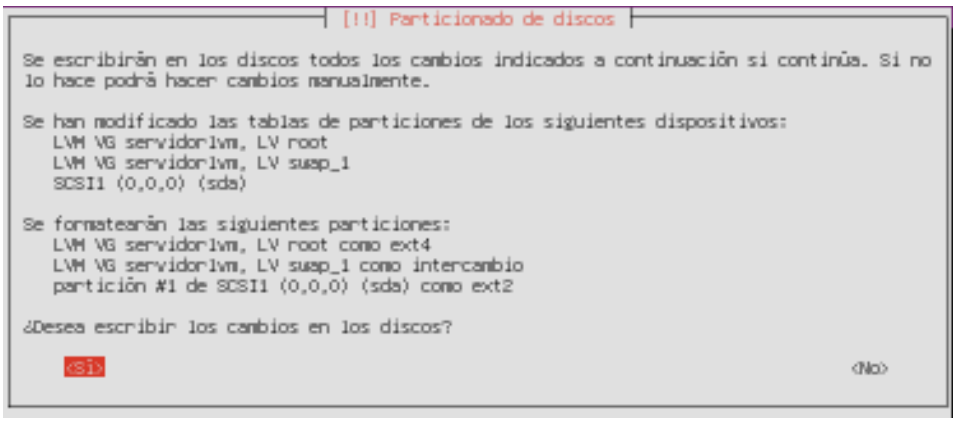
```
[root@mv1]# mount /mnt/nominas
[root@mv1]# mount /mnt/facturas
[root@mv1]# mount /mnt/presupuestos
```

Ej. Práctico 4.6.6: Extender el Volumen Raíz del Sistema (/) sin Desmontar

A continuación comprobaremos que podemos aumentar el espacio disponible en un volumen lógico sin necesidad de su desmontaje. En concreto, redimensionaremos el volumen raíz, /, donde se ha instalado todo el sistema, siendo este el más crítico ya que en caso de un descuido el sistema dejaría de iniciarse correctamente. En concreto extenderemos el volumen root / en 4GB.

Para ello, se aconseja crear una nueva máquina virtual (*p.e. Ubuntu Server*) de tal forma que durante la fase de instalación correspondiente al particionado, se seleccione como "*Método de particionado*" "*Guiado – utilizar el disco completo y configurar LVM*", tal como se muestra en la siguientes figura. De esta forma, tal como se puede observar en las siguientes capturas de pantalla, tendremos un volumen lógico raíz, /, además del volumen lógico swap, donde se encuentra todo el sistema de archivos de nuestra distribución, el cual pasaremos a extenderlo una vez terminada la instalación.





Solución Ej. Pr. 4.6.6.I.- Cómo Extender el Volumen Lógico root /

Para ello seguiremos los siguientes pasos (*repasar antes la sintaxis de los comandos de gestión LVM explicados anteriormente*):

1) Agregaremos un nuevo disco SATA a la máquina virtual (*p.e. /dev/sdb*). Por ejemplo, si el disco es de 8GB, 4GB los usaremos para extender el volumen root y los 4GB restantes los usaremos en el siguiente ejercicio práctico, donde aprenderemos a reducir el tamaño de un volumen lógico.
2) Convertiremos el disco anterior en un nuevo PV:

```
[root@mv1]# pvcreate /dev/sdb
```

3) Extenderemos el Grupo de Volúmenes al cual pertenece el Volumen Lógico root / mediante el comando **vgextend**:

```
[root@mv1]# vgextend mv1 /dev/sdb
```

4) Extenderemos el Volumen Lógico root / con cualquiera de los comandos siguientes (*se supone que tanto /dev/sda como /dev/sdb son de 8GB*):

```
[root@mv1]# lvextend /dev/mv1/root --size 12G
[root@mv1]# lvextend /dev/mv1/root --size +4G
[root@mv1]# lvextend /dev/mv1/root --extents 75%VG
[root@mv1]# lvextend /dev/mv1/root --extents +50%FREE
```

5) Reformatearemos el volumen para dar formato al espacio del volumen agregado, y comprobaremos el nuevo tamaño del volumen:

```
[root@mv1]# resize2fs /dev/mv1/root
[root@mv1]# df -h
```

Ej. Práctico 4.6.7: Reducir el tamaño de un Volumen Lógico

Si en el ejercicio práctico anterior hemos aprendido a extender un LV, ahora repasaremos como reducir el espacio de un LV (*repasar los ejercicios practicos anteriores donde ya se han redimensionado LVs*).

¡¡Importante!! A diferencia de cuando extendíamos un volumen lógico LV, debemos tener en cuenta que ahora será necesario su desmontaje previo. Al tener que desmontar el LV para hacer este tipo de operación no lo haremos sobre el volumen root (*ya que sino el sistema dejaría de funcionar*). Por ello, crearemos un nuevo LV (*p.e. datos*) a partir del espacio libre restante que nos queda en el VG del ejercicio práctico anterior.

Solución Ej. Pr. 4.6.7.I.- Cómo Reducir el tamaño de un Volumen Lógico

Comenzaremos la práctica creando un nuevo LV llamado p.e. datos, con el espacio restante de la práctica anterior (*p.e. 4GB, si el disco /dev/sdb era de 8GB*) al cual le reduciremos el tamaño a posteriori:

```
[root@mv1]# df -h
[root@mv1]# lvcreate mv1 -n datos --extents 100%FREE
```

```
[root@mv1]# mkfs.ext4 /dev/mv1/datos
[root@mv1]# mkdir /mnt/datos
[root@mv1]# nano /etc/fstab
...
/dev/mv1/datos /mnt/datos ext4 defaults,acl,usrquota,grpquota 0 0
[root@mv1]# mount /mnt/datos
[root@mv1]# df -h
```

Una vez montado el nuevo LV datos, y comprobado que funciona correctamente, pasaremos a su reducción de tamaño, siendo necesario su desmontaje, redimensionamiento del sistema de archivos y redimensionamiento del LV:

```
[root@mv1]# umount /mnt/datos
[root@mv1]# e2fsck -f /dev/mv1/datos
[root@mv1]# resize2fs /dev/mv1/datos 2G
[root@mv1]# lvreduce /dev/mv1/datos --size 2G
[root@mv1]# lvs | lvscan | lvdisplay
[root@mv1]# mount /mnt/datos
[root@mv1]# df -h
```

4.7.- Delegación de Privilegios a Usuarios del Sistema: SUDO

Para terminar con la presente práctica, aprenderemos a delegar privilegios de administración (*root*) a usuarios del sistema diferentes del **root**. En concreto, haremos uso del paquete software sudo, el cual nos permite delegar privilegios y ejecutar comandos del sistema en nombre del usuario **root**, o de cualquier otro.

¡¡Importante!! Dependiendo de la distribución GNU/Linux con la que estemos trabajando podremos encontrarnos el paquete **sudo** preinstalado y preconfigurado (p.e. Ubuntu), o no (p.e. Debian). En caso de no disponer de **sudo** habrá que instalarlo previamente para poder llevar a cabo la práctica:

```
[root@linux]# apt-get install sudo
```

Por ejemplo, si tuviéramos instalado en nuestra máquina el software servidor **apache**, deberíamos tener una cuenta de usuario llamada **www-data** a través de la cual apache interactúa con el sistema. Si quisiéramos crear un directorio en nombre de dicho usuario, tendríamos la opción de suplantarlo mediante el comando "**su www-data**" (*debe tener una shell válida*) y después crear el directorio, o directamente hacer uso de sudo en el caso que se haya configurado correctamente para ello:

```
[usuario@linux]$ sudo -u www-data mkdir directorio1
[root@linux]# sudo -u www-data mkdir directorio1
```

Otro ejemplo ilustrativo sería conceder privilegios en nombre del root a un usuario cualquiera del sistema pudiera instalar un programa en el sistema. Como puede advertirse a continuación, en el caso de que el usuario en nombre de quien ejecutamos el comando sea el root, puede omitirse (*-u root*) al ser este el usuario por defecto:

```
[usuario@linux]$ sudo -u root apt-get install openshot
[usuario@linux]$ sudo apt-get install openshot
```

¡¡Advertencia!! Al ejecutar un programa mediante sudo, éste nos solicitará nuestra password con la finalidad de corroborar que el usuario que lo esta ejecutando es el usuario al que se le concedieron los privilegios. Es decir, al solicitar **sudo** la password al usuario esta evitando que otro usuario pueda ejecutar comandos indebidamente (*p.e. si nos dejáramos una sesión abierta, otro usuario podría ejecutar comandos privilegiados en nuestro nombre*). En el caso de que la introducción de esta password nos resulte muy tedioso, o queramos ejecutar un comando del sistema a través del sudo de manera desatendida (*sin estar presentes para poder introducir la password*), podremos evitar su solicitud incluyendo la directiva **NOPASSWD:** en el archivo de configuración del sudo, tal como se explicará más adelante.

4.7.1.- Configuración de Sudo: /etc/sudoers.

Con la finalidad de gestionar los privilegios a conceder será necesario que el usuario

administrador **root** edite el fichero de configuración de **sudo**, **/etc/sudoers**. Para ello ejecutaremos el comando **visudo**, el cual hará uso de manera indirecta del editor **nano**, y al mismo tiempo hará una comprobación sintáctica de las modificaciones que realicemos antes de salir del editor.

```
[root@linux]# visudo
```

La sintaxis que deberemos cumplir a la hora de asignar privilegios, por cada línea que añadamos en la configuración del **sudo**, deberá ser la siguiente (*suponemos que mv1 es el nombre de la máquina*, *more /etc/hostname*):

```
<nombre usuario / grupo>     <máquina> = ( <usuario suplantado> ) <comando>
arturo          mv1 = (root) /usr/bin/apt-get install proftpd, /usr/bin/apt-get install apache2
arturo          ALL = (ALL) NOPASSWD: ALL
%gadmin1        mv1 = (root) NOPASSWD: /sbin/route
```

Donde cada uno de los campos a incluir representa lo siguiente:

- **<nombre usuario / grupo>**: se corresponde con el nombre de la cuenta de usuarios o grupo de usuarios a los que se les quiere dar permisos o privilegios administrativos. Deberá corresponderse con una cuenta de usuario o grupo registrada en el sistema. Además, en el caso de hacer referencia a un grupo, éste deberá ir precedido de un signo de tanto por cien, **%**, que advertirá de ello.
- **<máquina>**: indicaremos sobre que máquina concedemos los permisos administrativos. Normalmente será la máquina local, "**mv1**", "**localhost**", etc (*more /etc/hostname*). Puede usarse la palabra clave **ALL** para hacer referencia a cualquier máquina.
- **<usuario suplantado>**: indicaremos el nombre de "en nombre de quien" llevamos a cabo esos privilegios administrativos. Generalmente será el usuario "**root**", en cuyo caso se podrá omitir. En el caso de usar la palabra clave **ALL** querrá decir que el usuario o grupo de usuarios afectado podrá ejecutar comandos en nombre de cualquier usuario que haya dado de alta en el equipo.
- **<comando>**: indicaremos que comando permitimos que ejecute el usuario o grupo de usuarios indicado. En el caso de ser más de un comando se separarán por comas. Se aconseja hacer uso de la ruta absoluta del *path* donde se encuentra el ejecutable. En el caso de usar la palabra clave **ALL** querremos decir que el usuario o grupo de usuarios afectados podrá ejecutar cualquier comando del sistema en nombre del usuario suplantado. Además, existe la posibilidad de incluir la directiva **NOPASSWD:** con la finalidad de evitar que el usuario tenga introducir la password, una vez ejecutado un comando mediante **sudo**, que corrobore que es él a quien se le ha dado los privilegios para ejecutar el comando.

Además, con la finalidad de facilitar la configuración de **sudo**, dentro del archivo "**/etc/sudoers**" pueden definirse "alias", mediante los cuales se nos permitirá en una única línea conceder varios privilegios, en diferentes máquinas y a varios usuarios simultáneamente (*sin necesidad de que pertenezcan a un mismo grupo*). Para ello haremos uso de las directivas de configuración (*son sensibles a mayúsculas y minúsculas*):

A) **Host_Alias**: Nos permite crear "alias" para una o varias máquinas.
B) **User_Alias**: Nos permite crear "alias" para uno o varios usuarios.
C) **Cmnd_Alias**: Nos permite crear "alias" para uno o varios comandos.

A modo de ejemplo:

```
[root@linux]# visudo
Host_Alias    SALA = equipo1, equipo2, equipo3, 192.168.121.12, informatica.ies.com
User_Alias    PRIVILEGIADOS = arturo, juanjo, maria
Cmnd_Alias  PERMITIDOS = /sbin/route, /sbin/reboot
# Una vez creados los alias haremos uso de ellos siguiendo la sintaxis descrita anteriormente:
PRIVILEGIADOS   SALA = (root) NOPASSWD: PERMITIDOS
```

Además, tenemos la posibilidad de crear un archivo de auditoría de quien hace uso del **sudo** y los privilegios que se le han concedido añadiendo la siguiente línea al archivo de configuración del **sudo** (*no es necesario crear el archivo sudo.log, lo crea el sistema en el momento en que un usuario haga uso de sus privilegios mediante sudo*):

```
[root@linux]# visudo
...
Defaults      logfile=/var/log/sudo.log, log_year
```

Por último señalar que un usuario del sistema puede saber los privilegios que se le han concedido a través de **sudo** ejecutando el comando "**sudo -l**":

```
[arturo@linux]$ sudo -l
Password:
User arturo may run the following commands on this host:
        (root) /etc/init.d/smbd, /etc/init.d/ssh
        (root) /usr/bin/apt-get install *, /usr/bin/apt-get remove
        (root) NOPASSWD: /sbin/route, /sbin/reboot
```

Ej. Práctico 4.7.1: Delegación de Privilegios a Usuarios el Sistema

Configura **sudo** teniendo en cuenta las premisas establecidas en la siguiente tabla (*deberás crear los usuarios y grupos de usuarios previamente*), asumiendo que a excepción de los usuarios del grupo de usuarios "grupolinux1" (**NOPASSWD:**), el resto de usuarios deberán introducir su password al hacer uso de **sudo**:

Usuarios	Grupo Usuarios	User_Alias	Cmnd_Alias	Usuario Suplantado
usulinux1	grupolinux1	*Ninguno*	SOFTINSTALAR, ADMCUENTAS	**root**
usulinux2			SOFTDIR	**www-data**
usulinux3	grupolinux2	GRP2	SOFTINSTALAR, SOFTACTUALIZAR	**root**
usulinux4				
usulinux5	grupolinux3	*Ninguno*	SOFTSERVICIOS	**root**

Donde los alias de comandos anteriores hacen referencia a los siguientes comandos del

sistema:

Cmnd_Alias	Lista de Comandos
SOFTINSTALAR	**/usr/bin/apt-get install *, /usr/bin/apt-get remove *, /usr/bin/apt-get purge ***
SOFTACTUALIZAR	**/usr/bin/apt-get update, /usr/bin/apt-get upgrade**
ADMCUENTAS	**/usr/sbin/useradd, /usr/sbin/usermod, /usr/sbin/userdel**
SOFTDIR	**/bin/mkdir, /bin/cp, /bin/rm, /bin/mv**
SOFTSERVICIOS	**/usr/bin/apt-get install apache2, /usr/bin/apt-get install proftpd, /usr/bin/apt-get install bind9, ...**

Solución Ej. Pr. 4.7.1.I.- Configuración de Sudo. Como Delegar Privilegios

Para dar solución al problema planteado seguiremos los siguientes pasos: **(1)** Crearemos los usuarios y grupos de usuarios del sistema, y después **(2)** añadiremos la configuración necesaria en **sudo** mediante **visudo**.

```
[root@linux]# groupadd grupolinux1
[root@linux]# groupadd grupolinux2
[root@linux]# groupadd grupolinux3
[root@linux]# useradd -g grupolinux1 -m -d /home/usulinux1 -s /bin/bash usulinux1
[root@linux]# useradd -g grupolinux1 -m -d /home/usulinux2 -s /bin/bash usulinux2
[root@linux]# useradd -g grupolinux2 -m -d /home/usulinux3 -s /bin/bash usulinux3
[root@linux]# useradd -g grupolinux2 -m -d /home/usulinux4 -s /bin/bash usulinux4
[root@linux]# useradd -g grupolinux3 -m -d /home/usulinux5 -s /bin/bash usulinux5
[root@linux]# passwd usulinux1
[root@linux]# passwd usulinux2
[root@linux]# passwd usulinux3
[root@linux]# passwd usulinux4
[root@linux]# passwd usulinux5
```

```
[root@linux]# visudo
User_Alias GRP2 = usulinux3, usulinux4
Cmnd_Alias SOFTINSTALAR = /usr/bin/apt-get install *, /usr/bin/apt-get remove *,
                          /usr/bin/apt-get purge *
Cmnd_Alias SOFTACTUALIZAR = /usr/bin/apt-get update, /usr/bin/apt-get upgrade
Cmnd_Alias ADMCUENTAS = /usr/sbin/useradd, /usr/sbin/usermod, /usr/sbin/userdel
Cmnd_Alias SOFTDIR = /bin/mkdir, /bin/cp, /bin/rm, /bin/mv
%grupolinux1        ALL = (root)  NOPASSWD: SOFTINSTALAR,  ADMCUENTAS
%grupolinux1        ALL = (www-data)  NOPASSWD: SOFTDIR
GRP2                ALL = (root)  SOFTINSTALAR,  SOFTACTUALIZAR
usulinux5           ALL = (root) /usr/bin/apt-get install apache2,
                          /usr/bin/apt-get install proftpd, /usr/bin/apt-get install bind9, ...
Defaults     logfile=/var/log/sudo.log, log_year
```

...

Por último, comprobaremos que surte efecto los privilegios concedidos mediante **sudo** ejecutando algunos de los comandos anteriores desde alguna de las cuentas de usuario afectadas:

```
[root@linux]# chown -R www-data /var/www
[root@linux]# su usulinux1
[usulinux1@linux]$ sudo -l
[usulinux1@linux]$ sudo apt-get install openssh-server
[usulinux1@linux]$ sudo -u www-data mkdir /var/www/sitioweb1
[usulinux1@linux]$ ls -l /var/www
```

```
[root@linux]# su usulinux5
[usulinux5@linux]# sudo -l
[usulinux5@linux]# apt-get install apache2
```

Ej. Práctico 4.7.2: Delegación de la gestión LVM a un Usuario del Sistema

Crea un grupo de usuarios del sistema llamado "**gestores-discos**" compuesto por las cuentas de usuario "**gestor1**", "**gestor2**" y "**gestor3**", y concédeles los privilegios mediante **sudo** con el uso de **Cmnd_Alias** para que puedan hacer uso de los comandos básicos de gestión de volúmenes RAID, comandos de gestión LVM, y comandos de gestión de ACLs y Cuotas vistos en los ejercicios prácticos anteriores: **pvcreate**, **vgcreate**, **vgextend**, …, **resize2fs**, **mount**, **umount**, etc. Comprueba posteriormente suplantando a dicho usuario que puede llevar a cabo una gestión del almacenamiento del sistema sin necesidad de suplantar al usuario **root**.

Grupo Usuarios	Usuarios	Usuario Suplantado	Cmnd_Alias	Privilegios / Comandos
gestores-discos	**gestor1** **gestor2** **gestor3**	**root**	**SOFTLVM**	**/sbin/pvcreate,** **/sbin/vgcreate,** **/sbin/vgextend,** **/sbin/vgreduce,** **/sbin/lvreduce, …**
			SOFTDISCOS	**/sbin/fdisk, /sbin/resize2fs,** **/sbin/e2fsck,** **/sbin/mkfs.ext4, ...**
			SOFTRAID	**/sbin/mdadm**
			SOFTACLS	**/bin/setfacl, /bin/getfacl**
			SOFTCUOTAS1	**/sbin/quotacheck,** **/sbin/quotaon,** **/sbin/quotaoff**
			SOFTCUOTAS2	**/usr/sbin/repquota,** **/usr/sbin/quotatool**
			SOFTMOUNT	**/bin/mount, /bin/umount**

Solución Ej. Pr. 4.7.2.I.*- Cómo Delegar la Gestión LVM a un Grupo de Usuarios*

Para dar solución al problema práctico planteado seguiremos los siguientes pasos:

(1) Crearemos el grupo de usuarios y los usuarios que lo componen:

```
[root@linux]# groupadd gestores-discos
[root@linux]# useradd -m -d /home/gestor1 -s /bin/bash -g gestores-discos gestor1
[root@linux]# useradd -m -d /home/gestor2 -s /bin/bash -g gestores-discos gestor2
[root@linux]# useradd -m -d /home/gestor3 -s /bin/bash -g gestores-discos gestor3
```

(2) Editaremos la configuración de sudo con visudo añadiendo las siguientes líneas:

```
[root@linux]# visudo
# Crearemos los alias de los comandos indicados en el enunciado:
Cmnd_Alias SOFTLVM=/sbin/pvcreate, /sbin/vgcreate, /sbinvgextend, ...
Cmnd_Alias SOFTDISCOS=/sbin/fdisk, /sbin/resize2fs, /sbin/e2fsck, /sbin/mkfs.ext4, ...
Cmnd_Alias SOFTRAID=/sbin/mdadm
Cmnd_Alias SOFTACLS=/sbin/setfacl, /bin/getfacl
Cmnd_Alias SOFTCUOTAS1=/sbin/quotacheck, /sbin/quotaon, /sbin/quotaoff
Cmnd_Alias SOFTCUOTAS2=/usr/sbin/repquota, /usr/sbin/quotatool
Cmnd_Alias SOFTMOUNT= /bin/mount, /bin/umount
# Concederemos los privilegios al grupo gestores-discos:
%gestores-discos ALL = (root) SOFTLVM, SOFTDISCOS, SOFTRAID, SOFTACLS, SOFTCUOTAS1, SOFTCUOTAS2, SOFTMOUNT
# Si quisieramos que los usuarios del grupo gestores-discos no tuvieran que introducir password al ejecutar los comandos anteriores mediante sudo:
# %gestores-discos ALL = (root) NOPASSWD: SOFTLVM, SOFTDISCOS, SOFTRAID, SOFTACLS, SOFTCUOTAS1, SOFTCUOTAS2, SOFTMOUNT
```

Práctica Nº5.- Implementación de un Cluster Hadoop. Sistema de Archivos Distribuido HDFS

En la presente práctica implementaremos un cluster compuesto por varios equipos (*máquinas virtuales*) mediante la herramienta de software libre de la comunidad Apache **hadoop**, a través de los repositorios de **cloudera**. Además, cabría aclarar antes de nada, que las pretensiones de esta práctica son **(1)** que el alumno comprenda como se implementa un cluster, **(2)** que entienda en que consiste un sistema de archivos distribuido HDFS (*Hadoop Distributed File System*) y **(3)** que advierta que beneficios puede aportarnos. Por ello, con la finalidad de centrarnos en esos aspectos y facilitar otros más enrevesados como es la fase de instalación del cluster, se hará uso de unos repositorios de hadoop cloudera correspondientes a una versión antigua, en lugar del código fuente disponible correspondiente a la última versión de Apache hadoop.

Un **cluster** consiste en la agrupación de un conjunto de equipos con la finalidad de poder repartir un trabajo en común entre todas las máquinas que lo componen (*p.e. ejecución de instrucciones de un programa paralelizable*).

5.1.- Introducción a Hadoop y su Sistema de Archivos HDFS

Hadoop es un proyecto desarrollado por la comunidad Apache Software Fundation que aglutina diferentes subproyectos, donde se desarrolla software de código abierto. En concreto, proporciona un framework, escrito en Java, sobre el cual desarrollar aplicaciones distribuidas que requieren un uso intensivo de datos y de alta escalabilidad.

Se presenta como una solución para los programadores sin experiencia en desarrollo de aplicaciones para entornos distribuidos, dado que oculta la implementación de detalles propios de estos sistemas: paralelización de tareas, administración de procesos, balanceo de carga y tolerancia a fallos. Hadoop está inspirado en las publicaciones de Google sobre el modelo de programación MapReduce y sobre su sistema de ficheros distribuido denominado GFS (*Google File System*). Por tanto, Hadoop implementa, entre otras cosas, el paradigma MapReduce y un sistema de ficheros distribuido denominado HDFS (*Hadoop Distributed File System*).

HDFS es un sistema de ficheros pensado para almacenar grandes cantidades de información, del orden de terabytes o petabytes, tolerante a fallos y diseñado para ser instalado en máquinas de bajo coste. La información es dividida en bloques, que son almacenados y replicados en los discos locales de los nodos del cluster. Tiene muchas similitudes con otros sistemas de ficheros distribuidos, pero es diferente en varios aspectos:

(A) Una diferencia notable es, que está pensado para aplicaciones que siguen un modelo de una sola escritura y muchas lecturas, permitiendo relajarlos requisitos de control de concurrencia, simplificando la coherencia de los datos, proporcionando como consecuencia un acceso de alto rendimiento.

(B) Otra cualidad única de HDFS es que parte de la suposición que, por lo general es mejor ubicar

la lógica de procesamiento cerca de los datos en lugar de mover los datos al espacio de aplicación.

HDFS se compone de un grupo de nodos interconectados, donde residen los archivos y directorios. Presenta una arquitectura master/worker basada en un único nodo maestro, denominado **NameNode**, que maneja el espacio de nombres del sistema y regula el acceso de los clientes a los ficheros, redirigiéndolos a los nodos de datos que contienen la información, denominados **DataNodes**, que son los encargados de gestionar el almacenamiento en los discos locales del propio nodo. Tanto el NameNode como los DataNodes son componentes software, diseñados para funcionar, de manera desacoplada, en máquinas genéricas a través de sistemas operativos heterogéneos.

HDFS ha sido construido utilizando el lenguaje de programación Java, por lo tanto, cualquier máquina que soporte dicho lenguaje puede ejecutar HDFS. Una instalación típica consta de una máquina dedicada, donde se ejecutará el NameNode, y en cada una de las máquinas restantes que constituyen el clúster, se ejecutará un DataNode.

Una aplicación cliente, que desea leer un fichero en HDFS, debe contactar primero con el NameNode, para determinar el lugar en el cual está almacenada la información que requiere. En respuesta al cliente, el NameNode retorna el identificador del bloque más relevante y el nodo en el cual está almacenado. A continuación, el cliente contacta con el DataNode para recuperar la información requerida.

Una característica importante del diseño de este sistema de ficheros es, que la información nunca se mueve al NameNode. Toda la transferencia de información se produce directamente entre los clientes y los nodos de datos. La comunicación con el NameNode sólo implica transferencia de metainformación. Los DataNodes están periódicamente facilitando información de estado al NameNode, dado que este último no puede conectarse directamente con el primero. El NameNode simplemente se limita a responder a las peticiones realizadas por el DataNode, el cual mantiene un servidor de sockets abierto mediante el cual un cliente u otro DataNode puede leer o escribir información.

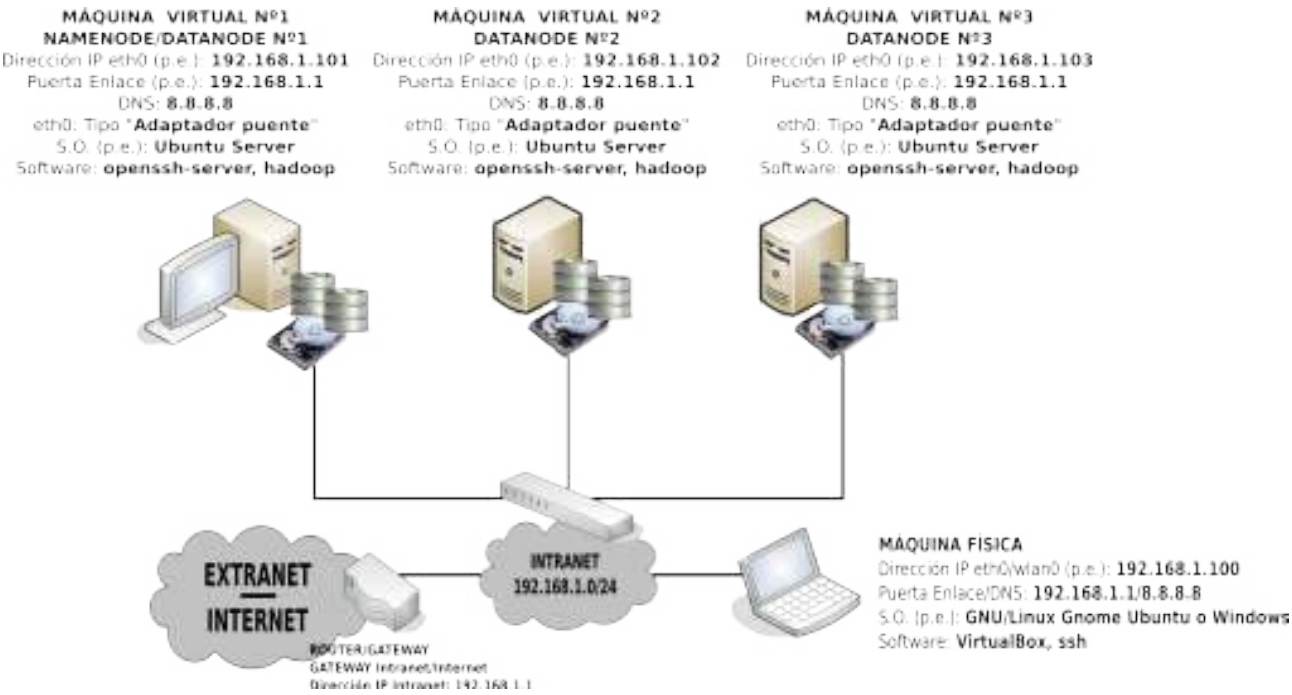
Para que el sistema de ficheros sea tolerante a fallos, HDFS replica los bloques de ficheros. Una aplicación puede especificar el número de réplicas para un fichero en el momento que es creado, pudiendo ser cambiado en cualquier momento. El NameNode toma las decisiones relativas a la replicación de bloques. Uno de los objetivos principales de HDFS es dar soporte a ficheros de gran tamaño. El tamaño típico de un bloque de fichero en HDFS es 64Mb o 128Mb. Un fichero está compuesto de uno o varios bloques de 64/128Mb y HDFS trata de colocar cada bloque en nodos de datos separados, distribuyendo la información a lo largo del clúster. Los bloques no siempre pueden ser colocados de manera uniforme en los nodos de datos, lo que significa que el espacio disponible por uno o más nodos de datos puede estar infrautilizado. Otro caso común, que provoca que la distribución de los datos entre los diferentes nodos no esté balanceada, es la adición de nuevos nodos de datos al cluster.

HDFS proporciona rebalanceo de bloques de datos utilizando diferentes modelos. Un modelo permite mover los bloques de un nodo de datos a otro, de forma automática, si el espacio libre en un nodo cae demasiado.

Otro modelo permite crear, dinámicamente, réplicas adicionales para un determinado fichero, si se produce un aumento repentino de la demanda, rebalanceando otros bloques en el clúster. HDFS también proporciona comandos que permiten realizar tareas de reajuste de forma manual.

5.2.- Implementación de un Sistema de Archivos Distribuido HDFS

En la presente práctica se pretende implementar un sistema de archivos distribuido **HDFS** (*Hadoop Distributed File System*), con la finalidad de **aumentar la velocidad en las operaciones de lectura y escritura** sobre el sistema de archivos (*se reparten en el trabajo entre los diferentes nodos del cluster*), así como **aumentar la disponibilidad** de los datos al comportarse el cluster como un sistema RAID1 al establecerse una replicación entre varios nodos de la red, de tal forma que si alguno de los nodos cae, siempre se pueden recuperar los datos a través de otro de los nodos donde haya una réplica. Como ya se verá más adelante, por defecto **hadoop** establece una replicación de 3 (*comportamiento equivalente a un sistema RAID1 formado por 3 discos*), aunque puede configurarse a posteriori. Posteriormente configuraremos un servicio de transferencia de archivos (*FTP, SMB, etc.*), el cual hará uso del sistema **HDFS** para almacenar los archivos, de tal forma que podremos hacer pruebas de rendimiento (*velocidad y disponibilidad*).



Aunque la finalidad última de la práctica sería implementar el cluster sobre un entorno real formado por equipos con sistema operativo de escritorio (*p.e. Ubuntu Gnome*), donde los usuarios estarían trabajando con sus equipos (*herramientas ofimáticas, Internet, etc.*), al mismo tiempo que en background estarían formando parte de un sistema de archivos distribuido HDFS, destinando parte de su potencia en ello (*hoy en día la mayor parte de los equipos esta funcionando por debajo de su capacidad de procesamiento*), la actual práctica se hará en un entorno virtualizado compuesto por tres máquinas virtuales con un sistema operativo sin entorno gráfico (*p.e. Ubuntu Server*) con la finalidad de no consumir excesivos recursos en la máquina física bajo la cual se virtualiza el cluster. Tal como se puede observar en la figura anterior, dispondremos de un cluster de 3 nodos (*un master y dos slaves*), donde uno de ellos hará tanto de **namenode** (*controlador del cluster o master*) como de **datanode**, y los otros dos restantes serán únicamente **datanode** (*almacen de datos del cluster, también llamados clientes o slaves*).

Además, para que la puesta en marcha y funcionamiento del cluster sea el detallado durante la práctica, sería conveniente comprobar que la versión del sistema operativo Ubuntu Server con el que se van a implementar los nodos del cluster es 12.04 o superior. Para corroborar la versión de la distribución del sistema operativo con el que estamos trabajando, podemos ejecutar el siguiente comando:

```
[usuario@hadoop1|2|3]$ lsb_release -a
```

A continuación se detallan los pasos para la implementación del cluster:

(0) Como paso previo a la instalación de **hadoop** prepararemos las máquinas o nodos que van a formar el cluster. Resaltar que las máquinas dispondrás de dos discos independientes, ya que en el primero estará instalado el sistema, y el segundo, lo usaremos como unidad de almacenamiento para el sistema de archivos distribuido HDFS:

Nombre Equipo *more /etc/hostname*	**Dirección IP (*p.e.*)**	**Namenode / Datanode**	**Nº Discos**	
hadoop1	192.168.1.101	Namenode / Datanode	2	/dev/sda /dev/sdb
hadoop2	192.168.1.102	Datanode	2	/dev/sda /dev/sdb
hadoop3	192.168.1.103	Datanode	2	/dev/sda /dev/sdb

Comenzaremos asignando una password a la cuenta de root o administrador de nuestros Ubuntu Server, ya que por defecto no dispone de contraseña. Su asignación nos permitirá conectarnos desde la máquina física a los nodos del cluster vía ssh como root y ahorrarnos tiempo a la larga:

```
[usuario@hadoop1|2|3]$ sudo su
[root@hadoop1|2|3]# passwd root
```

En cuanto a la configuración de red (dirección IP, puerta de enlace y servidor DNS), haremos uso de un script (*p.e. /etc/init.d/conf-red*) que se encargue de ello por su versatilidad y facilidad de modificación, al cual asignaremos permiso de ejecución e incluiremos en el script **rc.local** para que la configuración surta efecto al iniciarse la máquina. Por ejemplo, para el nodo master el script a crear tendría el siguiente aspecto (*en el resto de nodos crearíamos el mismo script donde únicamente modificaríamos la IP asignada*):

```
[root@hadoop1]# nano /etc/init.d/conf-red
#!/bin/bash
/etc/init.d/networking stop
ifconfig eth0 down
ifconfig eth0 192.168.1.101
route add default gw 192.168.1.1
echo "nameserver 8.8.8.8" > /etc/resolv.conf
```

```
[root@hadoop1]# chmod +x /etc/init.d/conf-red
[root@hadoop1]# bash /etc/init.d/conf-red
```

```
[root@hadoop1]# echo "bash /etc/init.d/conf-red" >> /etc/init.d/rc.local
```

Una vez realizada la configuración IP, configuraremos uno de los equipos como servidor DNS, o lo que es más sencillo modificaremos el archivo **/etc/hosts** del **master** para que pueda resolver nos nombres de las máquinas que componen el cluster:

```
[root@hadoop1]# nano /etc/hosts
# 127.0.0.1         hadoop1
192.168.1.101       hadoop1
192.168.1.102       hadoop2
192.168.1.103       hadoop3
```

(1) Instalamos los repositorios necesarios para instalar **java**, **hadoop**, y **hadoop-fuse-dfs** (*nos permitirá montar el sistema HDFS en nuestro sistema*):

¡¡Importante!! Si estas trabajando bajo una distribución GNU/Linux Ubuntu Server o Debian es posible que no dispongas de la herramienta de gestión de repositorios "**add-apt-repository**" necesaria para agregar repositorios PPA. En ese caso, será necesario que instales los paquetes "**software-properties-common**" y "**python-software-properties**":

```
[root@hadoop1|2|3]# apt-get install software-properties-common python-software-properties
```

```
[root@hadoop1|2|3]# add-apt-repository ppa:webupd8team/java
[root@hadoop1|2|3]# wget \
        http://archive.cloudera.com/one-click-install/maverick/cdh3-repository_1.0_all.deb
[root@hadoop1|2|3]# dpkg -i cdh3-repository_1.0_all.deb
[root@hadoop1|2|3]# apt-get update
```

(2) Una vez tengamos los repositorios actualizados instalaremos todos los paquetes necesarios para la creación y gestión del cluster, siendo necesario reiniciar los equipos al terminar toda la instalación:

Máquina Virtual / Dirección IP	Tipo Nodo	Software a Instalar
hadoop1 / 192.168.1.101	Namenode / Datanode	**oracle-java7-installer** **oracle-java7-set-default** **hadoop-0.20** **hadoop-0.20-namenode** **hadoop-0.20-datanode** **hadoop-0.20-fuse**
hadoop2 / 192.168.1.102	Datanode	**oracle-java7-installer** **oracle-java7-set-default** **hadoop-0.20** **hadoop-0.20-datanode**
hadoop3 / 192.168.1.103		

```
[root@hadoop1|2|3]# apt-get install oracle-java7-installer
[root@hadoop1|2|3]# update-java-alternatives -s java-7-oracle
```

```
[root@hadoop1|2|3]# apt-get install oracle-java7-set-default
[root@hadoop1]# apt-get install hadoop-0.20 hadoop-0.20-namenode \
                        hadoop-0.20-datanode hadoop-0.20-fuse
[root@hadoop2|3]# apt-get install hadoop-0.20 hadoop-0.20-datanode
[root@hadoop1|2|3]# init 6
```

(3) Configuramos las principales propiedades de configuración de **hadoop** a través de los ficheros de configuración "**core-site.xml**" y "**hdfs-site.xml**" ubicados en "**/etc/hadoop/conf/**" (*al editarlo advertiremos que el archivo no esta pendiente de configurar*), en todos los equipos o nodos del cluster, **master** o **namenode** y en los **datanode**. En concreto, será necesario ajustar al menos la propiedad "**fs.default.name**", cuyo valor se corresponderá con la dirección IP y puerto de servicio del equipo **master** del cluster, el **namenode** (*p.e. 192.168.1.101:9000*), y "**dfs.name.dir**" y "**dfs.data.dir**" para indicar la ubicación de los datos en el **namenode** y los **datanode**:

```
[root@hadoop1|2|3]# nano /etc/hadoop/conf/core-site.xml
...
<configuration>
<property>
<name>fs.default.name</name>
  <value>hdfs://192.168.1.101:9000</value>
  <description>URI del NameNode</description>
</property>
</configuration>
```

```
[root@hadoop1|2|3]# nano /etc/hadoop/conf/hdfs-site.xml
...
<configuration>
<property>
  <name>dfs.name.dir</name>
  <value>/cluster/dfs/name</value>
  <description>Determina donde se ubica el sistema de archivos DFS del namenode.</description>
</property>

<property>
  <name>dfs.data.dir</name>
  <value>/cluster/dfs/data</value>
   <description>Determina donde se ubica el sistema de archivos DFS donde el datanode almacena sus bloques. Puede indicarse una lista de directorios separados por comas.
  </description>
</property>
</configuration>
```

¡¡Importante!! Los directorios indicados en la configuración anterior para las propiedades **dfs.name.dir** y **dfs.data.dir** pueden ser cualesquiera dentro del sistema de archivos del nodo, teniendo que cumplir únicamente que el propietario sea el usuario **hdfs** bajo el cual funciona el cluster. No obstante, es recomendable que la partición o volumen del disco donde se almacenen los datos del cluster sea independiente a la del sistema, para evitar problemas de espacio. Por ello, tal

como se ha planteado al comienzo de la práctica, para su implementación haremos uso de un segundo disco (*/dev/sdb*) que se dedicará expresamente para ello: **(1)** Tras añadir un segundo disco a la cada una de las máquinas virtuales que implementan el cluster, crearemos una partición **/dev/sdb1** en el disco /dev/sdb mediante **fdisk** (*opción n*), **(2)** la formatearemos en **ext4**, **(3)** la montaremos en **/cluster** y por último **(4)** crearemos la estructura de directorios para el servicio hdfs.

```
[root@hadoop1|2|3]# fdisk /dev/sdb
[root@hadoop1|2|3]# mkfs.ext4 /dev/sdb1
[root@hadoop1|2|3]# mkdir /cluster
[root@hadoop1|2|3]# nano /etc/fstab
…
/dev/sdb1 /cluster ext4 defaults 0 0
[root@hadoop1|2|3]# mkdir -p /cluster/dfs/name /cluster/dfs/data
[root@hadoop1|2|3]# chown -R hdfs /cluster
[root@hadoop1|2|3]# chmod 700 /cluster/dfs/data
```

Antes de pasar al siguiente apartado de la práctica, sería importante señalar que aunque sólo es necesario personalizar las propiedades del cluster "**fs.default.name**", "**dfs.name.dir**" y "**dfs.data.dir**" para su puesta en marcha, existen otras muchas propiedades muy interesantes que podríamos personalizar entre las cuales cabría destaca la siguiente (*aquellas a las que no les asignemos un valor, adoptarán un valor por defecto, que más tarde podremos consultar*):

Propiedad	Archivo Configuración	Explicación / Utilidad
dfs.replication	**hdfs-site.xml**	Su valor determina el número de nodos datanode donde se replicará cada uno de los archivos que se almacenen en el cluster. Su valor por defecto es 3, lo que equivales a haber implementado un volumen RAID1 con 3 discos distribuidos.
file.replication	**core-site.xml**	

(4) Una vez configurado mínimamente el cluster hadoop, pasaremos a iniciar tanto el nodo principal **master**, también llamado **namenode**, como los nodos clientes o esclavos, también llamados **datanodes**. Comenzaremos por el equipo **namenode**, hadoop1/192.168.1.101: **(a)** suplantaremos a la cuenta de usuario del equipo **master** "**hdfs**", creada para gestionar el servicio, **(b)** daremos formato al nodo **master** y (c) lo iniciaremos.

```
[root@linux]# ssh root@192.168.1.101
[root@hadoop1]# su hdfs
[hdfs@hadoop1]$ hadoop namenode -format
[hdfs@hadoop1]$ hadoop namenode
```

¡¡Advertencia!! En el caso de que al ejecutar alguno de los comandos anteriores nos aparezca un error asociado a Java, relativo a que el sistema no sabe donde esta instalado Java (*variable JAVA_HOME*), deberemos crear una variable del sistema expresamente para ello:

```
[hdfs@hadoop1]$ hadoop namenode
...
|     Error: JAVA_HOME is not set and Java could not be found        |
```

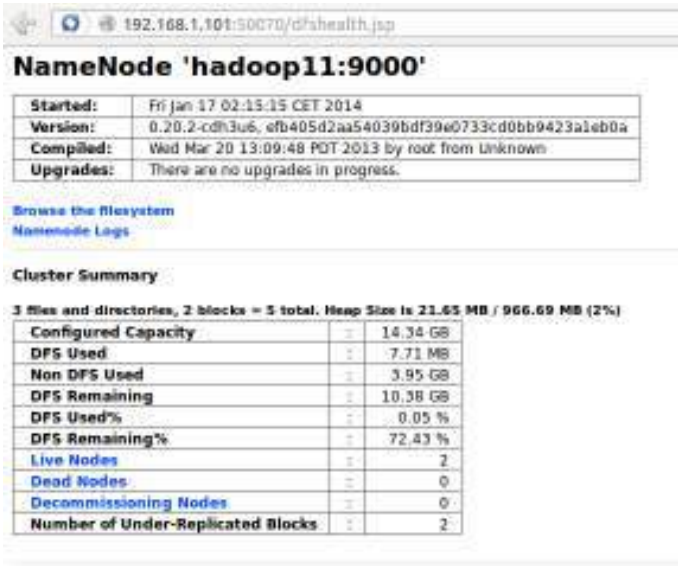
…

[root@hadoop1|2|3]# **export JAVA_HOME=/usr/lib/jvm/java-7-oracle**

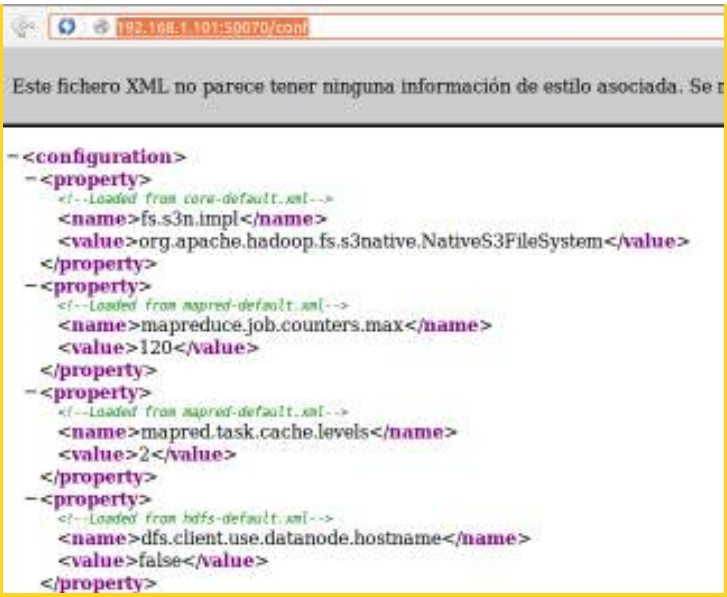
Para que la declaración de la variable del sistema surta efecto en los siguientes reinicios de los nodos del cluster, deberemos declararla en algún archivo que se ejecute al iniciarse la máquina o crear un archivo de definición de variables de sistema (*/etc/default*) expresamente para ello:

[root@hadoop1|2|3]# **echo "export JAVA_HOME=/usr/lib/jvm/java-7-oracle" > /etc/default/java**

Sabremos que se ha iniciado correctamente ya que deberíamos poder acceder al sitio web servido por el nodo **master namenode** que nos permite comprobar el estado del cluster. Para ello, iniciar un navegador web en la máquina física y escribir en la barra de direcciones la URL **http://192.168.1.101:50070**.



¡¡Importante!! Hadoop nos ofrece la posibilidad de monitorizar vía web los valores que reciben las propiedades de configuración del **cluster hadoop** una vez iniciado (*valores por defecto y asignados en core-site.xml y hdfs-site.xml*), escribiendo en la barra de direcciones de un navegador la URL siguiente "**http://192.168.1.101:50070/conf**", compuesta por la ip del **namenode** y el puerto de servicio por defecto, seguido del alias "**/conf**".



¡¡Observación!! Para comprobar a través de que puertos esta dando servicio el nodo **namenode** puede hacerse un escaneo de puertos mediante **nmap**. Esto es importante, ya que dependiendo de la implementación y versión de **hadoop** que estemos probando pueden cambiar.

[root@linux]# **nmap -A 192.168.1.101 -T5 -p1-65535**

(5) Una vez que hayamos comprobado que el equipo **namenode** funciona correctamente activaremos los **datanode**. Recordar que en la realización de la práctica el equipo master hará las veces de **namenode** y **datanode**, por lo que los pasos que se describen a continuación habrá que ejecutarlos en todos los nodos del cluster.

¡¡Advertencia!! Habrás advertido que una vez iniciado el **namenode** en nombre del usuario **hdfs** la terminal se queda inutilizable. Por esa razón, habrá que establecer una nueva conexión ssh (en el caso de que lo gestionemos remotamente desde la máquina física) o abrir una nueva terminal (*ALT+F2*, *ALT+F3*, *etc.*) para poder iniciar en el equipo master (hadoop1) el servicio **datanode**. No es aconsejable lanzar el comando "**hadoop namenode &**" en modo background ya que en caso de aviso o error este se muestra por pantalla.
Además, veremos que esta no es la única ocasión donde nos va a ocurrir esto en el equipo master namenode, ya que al ejecutar el comando que le hace trabajar como datanode, "hadoop datanode", nuevamente nos volverá a dejar la terminal inutilizable, por lo que tendremos que abrir una nueva terminal de conexión para montar posteriormente el sistema de archivos, lo cual nos volverá nuevamente a dejar inutilizable la última terminal abierta. En definitiva, al menos habrá que abrir cuatro terminales o conexiones ssh sobre el equipo master: **(1)** para hacerlo trabajar como namenode "hadoop namenode", **(2)** para hacerlo trabajar como datanode "hadoop datanode", **(3)** para montar el sistema de archivos, "**hadoop-fuse-dfs -d dfs://***ip_namenode***:9000** */punto_montaje* **-o rw,allow_other**" y **(4)** para configurar algún servicio que funcione bajo el sistema de archivos implementado.

[root@linux]# **ssh root@192.168.1.10*X***

```
[root@hadoop1|2|3]# su hdfs
[hdfs@hadoop1|2|3]$ hadoop datanode
```

¡¡Importante!! Puede comprobarse a través del sitio web "**http://192.168.1.101:50070**" como ha medida que se van agregando los nodos datanode al cluster queda reflejado en la fila de la tabla HTML "Live Nodes", y que el espacio del cluster "**Configured Capacity**" va aumentando (*pulsar F5 para refrescar los contenidos el sitio web*).

(6) A continuación montaremos el sistema de archivos distribuido HDFS en el nodo **master** mediante la ayuda de la herramienta **hadoop-fuse-dfs** instalada previamente, con la finalidad de que pueda ser utilizado como una unidad de almacenamiento más por el sistema:

¡¡Advertencia!! Es aconsejable descomentar del archivo de configuración de **fuse** la directiva "**user_allow_other**" con la finalidad de que usuarios que no sean el root puedan tener opciones de montaje:

```
[root@hadoop1]# more /etc/fuse.conf
...
user_allow_other
```

```
[root@hadoop1]# mkdir /mnt/hdfs
[root@hadoop1]# hadoop-fuse-dfs -d dfs://ip_namenode:9000 /mnt/hdfs -o rw,allow_other
```

Para comprobar que el montaje se ha realizado exitosamente listaremos los dispositivos de almacenamiento del **master** mediante "**df -h**":

```
[root@hadoop1]# df -h
S.ficheros      Tamaño  Usado  Disp  Uso%  Montado en
…
fuse_dfs        15G     0      15G   0%    /mnt/hdfs
```

En caso de querer desmontarlo deberemos recurrir a la herramienta **fusermount** de **fuse** (*opción -u, umount, seguido del punto de montaje*):

```
[root@hadoop1]# fusermount -u /mnt/hdfs
```

También existe la opción de montar el sistema HDFS de manera automática a través del archivo /etc/fstab, aunque deberemos tener en cuenta que no podrá hacerse en el propio nodo master, ya que al iniciarse el nodo trataría de automontarlo, antes de que el servicio se hubiera iniciado, lo cual daría un error de montaje:

```
[root@otroequipo]# nano /etc/fstab
...
hadoop-fuse-dfs#dfs://ip_namenode:9000 /mnt/hdfs fuse allow_other,rw 2 0
```

¡¡Importante!! Si ejecutamos un "**ls -l /mnt**" advertiremos que el único usuario con permisos suficientes para guardar información en la unidad de almacenamiento montada es el usuario "**hdfs**",

por lo que deberemos suplantarlo para guardar información en el cluster. Si lo suplantamos, y guardamos información dentro del punto de montaje, podremos monitorizar a través del sitio web "**http://192.168.1.101:50070**" como el campo "**DFS Used**" aumenta (*pulsar F5 para refrescar los contenidos el sitio web*).

Ej. Práctico 5.2.1: Servicio FTP bajo un sistema de archivos HDFS

Configurar un servicio FTP anónimo suplantando al usuario **hdfs** mediante el software servidor **proftpd** en el nodo **master** del cluster, de tal forma que la raíz de los documentos del sitio FTP sea el punto de montaje del cluster. De esta forma, podremos realizar pruebas de rendimiento y de alta disponibilidad del cluster.

Solución Ej. Pr. 5.2.1.I.- Implementación del Servicio FTP bajo HDFS

Dado por hecho que se han seguido los pasos especificados a lo largo de la práctica, aquí lo único que habrá que hacer es instalar y configurar el **proftpd** en el nodo **master**, y hacer las pruebas correspondientes:

```
[root@hadoop1]# apt-get install proftpd
[root@hadoop1]# nano /etc/proftpd/proftpd.conf
# Introducimos la siguiente configuración al comienzo del fichero proftpd.conf:
<Anonymous /mnt/hdfs>
        User hdfs
        UserAlias anonymous hdfs
        <Limit LOGIN>
                AllowAll
        </Limit>
</Anonymous>
...
```

¡¡Curiosidades!! Aunque se podrían enumerar multitud de curiosidades, señalar las siguientes:

(1) Deberíamos advertir que como la replicación por defecto del cluster es 3 (*propiedad **dfs.replication** comentada anteriormente*), al almacenar un archivo en cluster, si este es de 1MB, el espacio total ocupado en el cluster que podremos visualizar desde "http://*ip_namenode*:50070" (**DFS Used**) será de 1MB si hay un nodo activo, 2MB si hay dos nodos activos, y 3MB si hay 3 o más nodos activos.

(2) El usuario del sistema bajo el cual funciona y es gestionado el sistema de archivos HDFS es **hdfs**. Por esa razón, no tiene sentido tratar de realizar operaciones de lectura y escritura sobre este sistema de archivos a través de otra cuenta de usuario diferente a **hdfs**.

(3) El sistema de archivos HDFS de hadoop nos esta pensado para ser usado como un sistema de archivos clásico, sino que esta pensado como almacén de grandes bases de datos, con la finalidad de aumentar la disponibilidad de los datos y conseguir una mayor rapidez en las operaciones de lectura y escritura. Por esta razón, carece de sentido pretender establecer listas de control de acceso (*ACLs*)

al sistema de archivos o establecer cuotas de espacio en disco. Por esta razón, carece de sentido hacer uso de los comandos chmod, chown, chgrp, setfacl, quotatool, etc. con la finalidad de controlar permisos. El sistema de archivos HDFS esta pensado para que el único que lo manipule sea la cuenta de usuario **hdfs**.

(3) El sistema de archivos HDFS no es más que una capa de abstracción sobre el propio sistema de archivos ext4 con el cual se ha formateado el disco donde almacena los datos el **datanode**. Por tanto, **hadoop** y su sistema de archivos **HDFS** tan sólo es un nueva estrategia que se encarga de distribuir los datos por el sistema de archivos convencional.

(4) En la práctica que aquí se plantea, el nodo **namenode** haces las veces de **datanode**, lo cual puede ocasionar problemas de consumo de memoria RAM, ya que si el servicio **namenode** observa que el **datanode** no le deja suficiente memoria RAM directamente dejará de dar servicio (*kill*). Para que esto no suceda podemos decidir de antemano la cantidad de memoria RAM y memoria Virtual o SWAP que consumirá cada uno de ellos configurando las variables **HADOOP_NAMENODE_OPTS** y **HADOOP_CLIENT_OPTS**. La primera se encarga de establecer opciones del **namenode**, y la segunda del **datanode**. Ambas las encontraremos en el script **hadoop-env.sh** que podremos encontrar en **/etc/hadoop/conf/**. Por ejemplo, si quisiéramos establecer que la cantidad de memoria virtual a utilizar por el servicio **namenode** fuera de 128MB añadiríamos al principio de la variable anterior "**-Xmx128m -Dhadoop.security.lo**", y si quisiéramos asignar 256MB al **datanode** añadiríamos "-Xmx256m" al principio de la segunda variable indicada:

```
[root@hadoop1]# cd /etc/hadoop/conf
[root@hadoop1:hadoop/conf]# nano  hadoop-env.sh
…
export HADOOP_NAMENODE_OPTS="-Xmx128m -Dhadoop.security.lo
        -Dhadoop.security.logger=${HADOOP_SECURITY_LOGGER:-INFO,RFAS}
        -Dhdfs.audit.logger=${HDFS_AUDIT_LOGGER:-INFO,NullAppender
        $HADOOP_NAMENODE_OPTS"
export HADOOP_CLIENT_OPTS="-Xmx256m $HADOOP_CLIENT_OPTS"
...
```

Ej. Práctico 5.2.2: Servicio SMB bajo un sistema de archivos HDFS

Configurar el servicio Samba para ofrecer un directorio compartido llamado "smb-hdfs" con acceso público con permisos de lectura y escritura:

Directorio	Permisos	Usuario bajo el cual se Ofrece el Servicio	Acceso Visible en Entorno de red
/mnt/hdfs/serviciosamba	**Lectura/Escritura**	**hdfs**	**Público / Invitados** **Sí** *(browseable)*

Solución Ej. Pr. 5.2.2.I.- *Como Ofrecer Recurso Compartido bajo HDFS*

Al igual que en el ejercicio práctico anterior, se supondrá que se han seguido los pasos sugeridos en la práctica y que ya disponemos de un sistema de archivos HDFS montado bajo /mnt/hdfs sobre el cual el único usuario que tiene permisos de gestión es la cuenta de usuario hdfs. Por otro lado supondremos que no tenemos instalado samba, por lo simplemente lo instalaremos y lo configuraremos para cumplir el cometido indicado en el enunciado de la práctica:

```
[root@hadoop1]# apt-get install samba
[root@hadoop1]# nano /etc/samba/smb.conf
# Añadir el siguiente recurso compartido al comienzo del archivo smb.conf
[smb-hdfs]
path = /mnt/hdfs
writable = yes
browseable = yes
public = yes
guest ok = yes
force user = hdfs
...
```

```
[root@hadoop1]# testparm
[root@hadoop1]# /etc/init.d/smbd restart
```

Práctica Nº6.- Uso de la Criptografía en los actuales Sistemas Informáticos. Claves Simétricas y Asimétricas. Firma Digital. Entidades Certificadoras

La seguridad no era algo que preocupara excesivamente a los primeros usuarios de las redes de comunicaciones. En efecto, usando éstas para poco más que enviarse correo electrónico y compartir alguna impresora dentro de una universidad u organización cerrada, pocos problemas de seguridad podrían producirse.

En concreto en los primeros sistemas informáticos que aparecieron era impensable plantearse problemas en la seguridad, cuando ni siquiera existían los usuarios malintencionados, ya que normalmente los únicos usuarios experimentados en sistemas informáticos, solían coincidir con los responsables del propio sistema informático, y por tanto, de su seguridad. Con lo cual, no tenía mucho sentido ponerse barreras uno a si mismo.

Por tanto, en los comienzos de la informática, los pocos equipos informáticos existentes solían utilizarlos una única persona (*o un grupo muy reducido de personas*), donde básicamente el único problema de seguridad que existía era, que un usuario con desconocimiento total del sistema lo "tocase" y pudiera provocar efectos indeseados. De hay que las primeras medidas de seguridad que se dieron fueron el uso de claves de acceso (*medida de seguridad software*) y el encerrar el equipo informático en cuartos encerrados por llave (*medida de seguridad hardware*).

Los problemas de seguridad como hoy los entendemos, surgen a raíz del nacimiento de los sistemas informáticos multiusuario, donde varios usuarios diferentes compartían la misma máquina, con lo que para garantizar una confidencialidad y una integridad de los datos de cada usuario, se requería de unos determinados niveles de seguridad, que evitase que cualquier usuario pudiese hacer lo que quisiese de forma malintencionada. Ante esta situación, además de las medidas anteriores, aparecieron los permisos de lectura, escritura y de ejecución, los cuales garantizaban mediante su asignación o no asignación, que las acciones de todos los usuarios que iniciasen sesión en el sistema informático, tuvieran unos limites definidos en sus actuaciones: sólo podían leer, escribir, modificar y borrar dentro del sistema de ficheros del sistema informático donde tuvieran permisos para ello (*no podían borrar información que no les perteneciese, no podían acceder al interior de carpetas que no les perteneciese y sobre las cuales no se les había concedido tal permiso, no podían alterar más que aquella información que les perteneciese, y tan sólo podían leer aquella información sobre la que se les había dado permiso*).

Tras los sistemas informáticos multiusuario, y bajo la necesidad de interconexión de los equipos informáticos con la finalidad de compartir recursos, y gracias al abaratamiento de los sistemas informáticos, surgen las redes informáticas, con los grandes riesgos de seguridad que ello ha supuesto. Aplicaciones como son el correo electrónico (*smtp/pop/imap, etc.*), el control remoto (*telnet, ssh, etc.*), o cualquier otro servicio de red, ha conllevado indiscutibles ventajas para los usuarios, pero al mismo tiempo, impresionantes agujeros en la seguridad de los sistemas informáticos.

Es decir, hoy en día, el panorama ha cambiado radicalmente en relación a los principios de la informática. El mundo de la seguridad en las redes informáticas va adquiriendo una complejidad

inquietante. Caminamos hacia la interconexión total y cualquier aspecto de un sistema, irrelevante para su seguridad cuando se encuentra aislado, se convierte, si el sistema se conecta al exterior mediante una red, en un factor determinante del éxito o fracaso de la organización. De este modo, las organizaciones que desean utilizar el correo electrónico y los demás servicios de Internet, bien internamente, bien para relacionarse con el exterior, lo que hoy por hoy significa ya prácticamente todas las organizaciones, deben tomar medidas que permitan garantizar la confidencialidad, la integridad y al mismo tiempo la disponibilidad de la información. Para la consecución de todo ello se deben seguir una serie de pasos entre los cuales cabría recalcar los siguientes:

I) Identificación de las amenazas en la seguridad que se ciernen sobre los servicios que incluyen el uso de redes de comunicaciones.
II) Determinación de los requerimientos de los usuarios finales.
III) Identificación de los servicios de seguridad que dan una posible solución a las amenazas identificadas y las técnicas o mecanismos de seguridad necesarios para implementar dichos servicios de seguridad.
IV) Localizar los productos comerciales que proporcionan los citados servicios.

6.1.- Características de un Sistema Seguro. Tipos de Amenazas informáticas

Para una comprensión adecuada de lo que va a ser tratado a lo largo de la presente práctica, cabría recalcar una serie de conceptos y términos utilizados en el mundo de la seguridad informática:

(a) Confidencialidad: característica de un sistema que asegura que la información no sea visible por personas indeseadas. Para garantizarla se pueden llegar a establecer medidas de seguridad extremas que hagan casi inviable cualquier ataque (*p.e. complejas claves de cifrado, que se renuevan periódicamente*).

En los ejercicios prácticos que se propondrán a lo largo del presente capítulo o práctica garantizaremos la **confidencialidad** mediante el uso de técnicas de cifrado, de tal forma que una vez cifrada la información tan sólo podrá leerla o descifrarla aquel que posea la clave de descifrado necesaria.

(b) Integridad: característica de un sistema que nos garantiza que la información se recupera tal y como fue guardada, o lo que es lo mismo, que no ha sufrido ninguna manipulación intermedia.

En los ejercicios prácticos que se propondrán a lo largo del presente capítulo o práctica comprobaremos la **integridad** mediante el uso de la firma digital, ya que si el resultado de la comprobación de la firma es exitoso es indicativo de que la información recibida es integra.

(c) Disponibilidad: facilidad con la que la información puede ser recuperada, consultada y manejada por personas de confianza. Esta característica esta reñida con la primera, "**confidencialidad**", ya que es complicado llegar a un compromiso entre la cantidad de medidas de seguridad a colocar para preservar la "**confidencialidad**", y al mismo tiempo, presentar una alta "**disponibilidad**".

(d) Autenticación: característica que posee en la actualidad cualquier sistema de comunicaciones,

que nos permite confirmar que una persona es quien dice ser. Como veremos a lo largo de la presente práctica, actuales técnicas como puede ser la **firma digital**, nos permite asegurarnos de que quien dice ser un usuario, lo es en verdad. No obstante, como comentaremos al final de la práctica, siempre corremos el riesgo de sufrir el conocido ataque del intermediario, *Man in the Middle*.

En cuanto a las distintas amenazas informáticas que existen en la actualidad cabría recalcar las siguientes:

1) **Virus**: Según "**Fred B. Cohen**", un virus informático es un trozo de código ejecutable dentro de un ordenador que puede infectar a otros programas modificándolos para incluir una copia de sí mismo. Este virus necesita obligatoriamente un programa (*"**host**"*) donde insertarse para poderse ejecutar, por lo que no se puede considerar un programa o proceso independiente. Al ejecutarse el programa "**host**" infectado, el código del virus queda almacenado (*residente*) en la memoria RAM del equipo, aun cuando el programa que lo contenía haya terminado de ejecutarse. Esto provoca, que el virus pueda tomar el control de los servicios básicos del sistema operativo, infectando a los posteriores ficheros ejecutables que sean abiertos y ejecutados, añadiendo a su propio código al del programa infectado y grabándolo en disco, con lo cual el proceso de replicado se completa.

2) **Gusano**: Hace referencia a programas capaces de viajar por sí mismos a través de redes de computadores para realizar cualquier actividad una vez alcanzada la máquina. Aunque esta actividad no tiene por qué entrañar peligro, los gusanos pueden instalar en el sistema un virus, atacar al sistema como lo haría un intruso, o simplemente consumir excesivas cantidades de ancho de banda en la red afectada.

3) **Caballo de Troya**: De la misma forma que el antiguo caballo de Troya de la mitología griega escondía en su interior algo que los troyanos desconocían, y que tenía una función muy diferente a la que ellos pensaban, un troyano o caballo de Troya actual es un programa que aparentemente realiza una función útil para quién lo ejecuta, pero que en realidad realiza una función que el usuario desconoce, generalmente dañina. Estos suelen ser muy comunes en situaciones del falsa autenticación: se nos pide un "login" y "password" administrativos para llevar a cabo una tarea que realmente no lo requería, o en el caso de requerirlo, ha pasado desapercibido, habiendo capturado una información extremadamente valiosa para cualquier atacante.

4) **Conejos**: También llamados bacterias, son programas que de forma directa no dañan al sistema, sino que se limitan a reproducirse, generalmente de forma exponencial, hasta que la cantidad de recursos consumidos (*procesador, memoria, disco, etc.*) no son capaces de aceptar más peticiones de servicio.

5) **Sniffing**: Es el arte de capturar todos los paquetes que pasan por una red. En concreto, se trata de programas que se dedican a captar y analizar todo el trafico que pasa por la red, con serio peligro de que puedan ser interceptadas contraseñas u otro tipo de información confidencial.

6) **Spoofing**: Este ataque consiste en una suplantación de identidad. Es decir, es el nombre que recibe la utilización de direcciones MAC e IP falsas en comunicaciones TCP/IP. Su finalidad es la de simular la identidad de otra máquina de la red para conseguir accesos a recursos de un tercer sistema que ha establecido algún tipo de confianza basada en el nombre o la dirección IP del host suplantado.

7) **Hijacking**: Se produce cuando un atacante consigue interceptar una sesión ya establecida. El atacante espera a que la víctima se identifique ante el sistema y tras ello le suplanta como usuario autorizado.

8) **Applets**: Se trata de pequeños programas implementados mediante el uso del lenguaje de

programación Java, e incrustados en sitios WEB. Esto provoca que al navegar sin darnos cuenta pinchemos en algún pulsador de un "**applet**" (o cualquier otra acción), y ello pueda desencadenar una serie de acciones terribles para nuestro sistema. Debemos tener en cuenta que Java es un lenguaje muy potente, pudiendo provocar todo tipo de acciones.

9) Puertas Traseras: Son trozos de código en un programa que permiten a quién conoce su funcionamiento saltarse los métodos usuales de autenticación para realizar ciertas tareas. Habitualmente son insertados por los programadores para agilizar la tarea de probar su código durante la fase de desarrollo del mismo y se eliminan en el producto final, pero en ciertas situaciones el programador puede mantener estar puertas traseras el programa funcional, ya sea deliberada o involuntariamente.

6.2.- Técnicas y Estrategias de Cifrado de la Información.

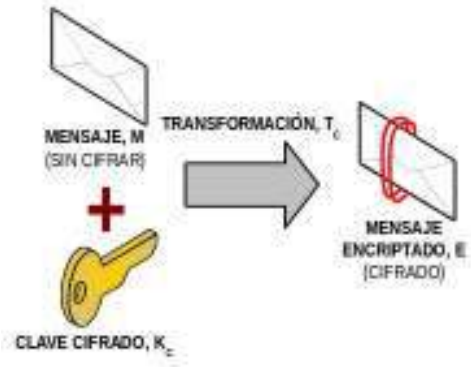
Una de las técnicas utilizadas para combatir alguna de las amenazas de seguridad comentadas anteriormente y que mejores resultados están dando, es la ocultación de la información. Es decir, asumiendo que la mayoría de los ataques que se realizan sobre las redes de comunicaciones, son con la finalidad de conocer información privilegiada que a posteriori permitiría a los atacantes provocar males mayores, la solución es bastante evidente, tratar de ocultarla a los atacantes, pero pudiendo seguir siendo visible para nosotros. De esto es lo que se encarga una de las ramas de las matemáticas más conocidas, como es la **criptología**.

La **criptología** se divide en dos ciencias antagonistas: al **criptografía** y el **criptoanálisis**. Todos estos términos tienen sus raíces etimológicas en el griego clásico. Así, criptografía se puede traducir como "escritura oculta" (*"criptos", oculto, y "graphos", escritura*). Según el DRAE, la criptografía es el "arte de escribir con clave secreta o de un modo enigmático". Evidentemente, hoy en día más que un arte se trata de una ciencia, cuyo cometido principal es conseguir que un mensaje sea sólo comprensible para sus legítimos destinatarios e ininteligible para cualquier extraño. Para ello, la criptografía se sirve de procedimientos matemáticos o claves que convierten el texto en claro en criptogramas.

En cuanto al criptoanálisis es la ciencia cuyo objetivo es quebrantar el cifrado obtenido por la criptografía. Esto se puede hacer descifrando un mensaje sin conocer la llave, o bien obteniendo a partir de uno o más criptogramas la clave que ha sido empleada en su codificación.

En concreto, con la finalidad de garantizar la confidencialidad, e integridad de la información, el mecanismo básico de actuación es el denominado criptosistema o algoritmo de cifrado, que especifica dos transformaciones: **(1) cifrado y (2) descifrado**.

(1) El cifrado. Conversión del texto en claro (*plain text*, *texto plano*) en el texto cifrado o criptograma (*Chipre text*) mediante el empleo de una función parametrizada y una clave (*Key*) de codificación. Desde un punto de vista más matemático, sería equivalente a decir que si al mensaje a enviar, M, se le aplica una transformación ó algoritmo de cifrado, T (*transformación*), se convierte en un mensaje encriptado E: $\mathbf{E=Tc_{kc}(M)}$.

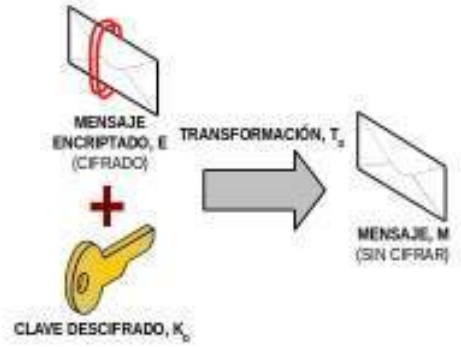


Este cifrado se puede llevar a cabo por software o por hardware, dependiendo a que nivel dentro de la estructura TCP/IP se haga.

En relación a el cifrado por software, podríamos decir que es aquel que se realiza como paso previo al paso de conversión del mensaje a binario (*0's y 1's*). En concreto, este tipo de cifrado se lleva a cabo en las capas de aplicación ó transporte.

En cambio el cifrado por hardware, es aquel que se realiza mediante el uso de un chip adecuado, donde al mensaje ya en formato binario se le aplica un algoritmo de cifrado. Este cifrado es llevado a cabo en la capa de enlace, y suele ser llevado a cabo por el propio interfaz de red (*tarjeta de red*), donde por medio de los drivers se le cargan los posibles algoritmos de cifrado.

(2) El descifrado. Proceso inverso al anterior y para el que se emplea la función o transformación inversa, la cual también suele necesitar como parámetro su correspondiente clave de descifrado. Matemáticamente la operación de descifrado se podría describir de la siguiente forma: $\mathbf{M=Td_{Kd}(E)=Td_{Kd}(Tc_{Kc}(M))}$.



Tal como se puede apreciar a través de las dos operaciones anteriores, existe un parámetro fundamental en ambas que es la clave. La clave no es más que una serie de dígitos que son utilizados en los algoritmos de cifrado de tal forma, que se ha comprobado matemáticamente que haciendo uso de una clave concreta para el cifrado, es necesario conocer su correspondiente clave para su descifrado, donde no necesariamente tienen que coincidir tales claves. Según esto, la seguridad de un sistema de cifrado radica casi por entero en la privacidad de las claves secretas, ya que el conocimiento de estas echaría al traste toda la supuesta seguridad que se trata de garantizar. Por ello, los ataques que puede realizar un criptoanalista enemigo están orientados a descubrir dichas claves.

Dicho de otra forma, el algoritmo o conjunto de operaciones que se siguen para encriptar o cifrar un mensaje es conocido por todo el mundo (*se dice que es un algoritmo público*), siendo la

clave secreta el único elemento utilizado en esos algoritmos que es privado, y por tanto, lo que garantiza la seguridad en la comunicación.

Los métodos clásicos de cifrados hacían énfasis en el secreto del algoritmo de cifrado y de hecho en muchos de ellos no existía el concepto de clave. Sin embargo, en los sistemas criptográficos modernos la seguridad no se basa en el secreto del sistema, sino en la robustez de sus operaciones (*algoritmos empleados*) y sus protocolos (*forma de usar los operadores*), siendo el único secreto la clave ya que los operadores y los protocolos son públicos.

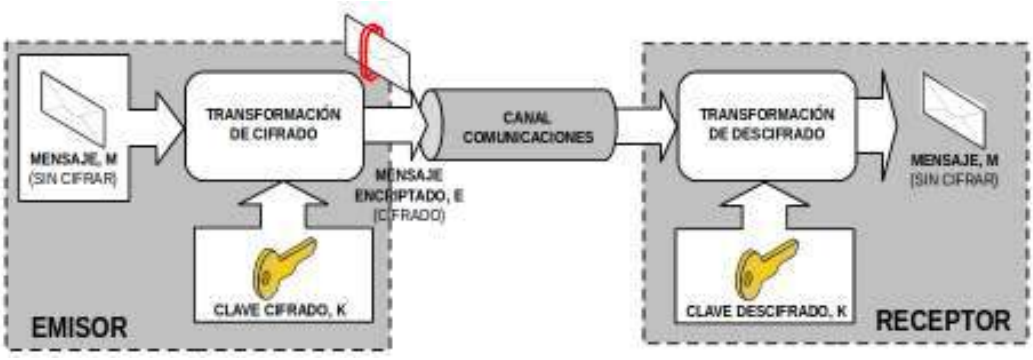
En definitiva, los elementos que intervienen en un criptosistema, podrían citarse los siguientes:

A) El conjunto de todos los mensajes sin cifrar que pueden ser generados y cifrados. Lo denominaremos M.
B) El conjunto de todos los posibles mensajes cifrados, o criptogramas. Lo denominaremos E.
C) El conjunto de claves que se pueden emplear en el criptosistema. Lo denominaremos K. Teniendo en cuenta que las claves utilizadas no tinen porque ser iguales en las transformaciones de cifrado y de descifrado, podrá distinguirse entre K_C y K_D.
D) El conjunto de transfomaciones de cifrado o algoritmos que se aplican a cada elemento de M para obtener un elemento de E. Lo denominaremos T_C. Como es lógico, el resultado de aplicar una transformación a un mensaje M, proporciona mensajes cifrados E totalmente diferentes para cada posible valor de la clave de cifrado K_C.
E) El conjunto de transformaciones de descifrado, al cual denominaremos T_D.

A continuación, se detallarán los algoritmos de cifrado que se utilizan en la actualidad, pudiéndose dividir en dos categorías: simétricos o de clave privada, y asimétricos o de clave pública.

6.3.- Cifrado de Información mediante Claves Simétricas (*Clave Secreta*)

La criptografía simétrica usa la misma clave para cifrar y para descifrar un mensaje por lo que su seguridad se basa en el secreto de dicha clave. Generalmente se utilizan dos funciones, una para realizar la codificación o cifrado del mensaje, y otra para la descodificación o descifrado del mensaje. Su principal desventaja es que hace falta que el emisor y el receptor compartan la clave, y para ello, en sistemas distribuidos es preciso que la clave viaje de algún modo desde el origen al destino: correo electrónico, telefónicamente, correo certificado, etc. Si este secreto fuese enviado por un canal inseguro, la seguridad del sistema sería bastante pobre, dado que cualquiera podría interceptarla y comprometer todo el sistema. También hay que tener en cuenta la frecuencia con la que esta clave debe ser renovada para evitar que sea desvelada.



Por tanto, el funcionamiento es muy sencillo: el emisor cifra el mensaje con la clave K, y se lo envía al receptor. Este último, que conoce dicha clave, la utiliza para desencriptar la información.

Uno de los métodos de clave secreta más conocido, todavía utilizado en aplicaciones de tipo comercial y otras donde no se requiere un elevado nivel de seguridad, es "DES" (*Data Encryption Standard*). "DES" fue el primer algoritmo desarrollado comercialmente (1977) y surgió como resultado de un acuerdo entre la NSA (*Nacional Security Agency*) de EEUU e IBM, siendo esta última la empresa informática a la que se encomendó su desarrollo.

"DES" puede ser implementado tanto por software como por hardware, alcanzando en hardware una velocidad de hasta 50 Mbps. Un ejemplo de implantación hardware puede ser PC-Encryptor, de Eracom, y un ejemplo de implantación software es DES-LOCK, de la empresa Oceanics.

El mecanismo de ataque más práctico contra DES es el de fuerza bruta, que consiste en intentar descifrar el texto codificado probando con todas las claves posibles ($2^{56}=7.206x10^{16}$ *posibilidades, al ser la clave de 56 bits*) y buscar resultados coherentes, y en el caso en que se disponga de algún segmento de texto en claro, comparar con el resultado. Respecto a esto, hay que resaltar que el tamaño inicial de la clave, en el diseño de IBM, era de 128 bits, aunque la razón de la disminución a 56 bits no se ha hecho pública hasta el momento.

El algoritmo es teóricamente robusto y no presenta debilidades significativas (*existen unas pocas claves, perfectamente identificadas, que no se deben utilizar*). No obstante, debido al actual desarrollo tecnológico, la seguridad proporcionada por el "DES" estándar (*clave de 56 bits*) es bastante limitada y de hecho ya se ha conseguido violar. Todo ello ha llevado a la búsqueda de otros sistemas simétricos alternativos: Triple-DES, IDEA, etc.

Por último cabría resaltar otro gran problema que conlleva el uso de la criptografía simétrica mediante el uso de claves secretas, ya que en una red de "n" usuarios, para asegurar confidencialidad entre todos ellos, cada pareja necesita tener su clave secreta particular, lo que requiere un total de n·(n-1)/2 claves para esa red, es decir, combinaciones de n usuarios tomadas de 2 en 2. Esto supone unas cinco mil claves en una red de sólo cien usuarios, medio millón en una de mil usuarios, etc., lo que supone que sea económicamente inaceptable el que se pueda distribuir todas estas claves por anticipado, e indeseable el tener que posponer las comunicaciones seguras mientras las claves están siendo trasladadas de una a otra parte.

Para dar solución a los problemas que surgen del uso de claves simétricas o secretas, aparecen los criptosistemas de clave pública o asimétrica.

Ej. Práctico 6.3.1: Cifrado Simétrico mediante GPG

Haciendo uso del software **gpg** (*GNU Privacy Guard, OpenPGP encryption and signing tool*) cifra un documento (*p.e. documento.pdf, archivo.dat, etc.*) haciendo uso de claves simétricas. Después envía dicho documento a un compañero (*vía email, smb, ftp, etc.*), y comprobad que la clave para el descifrado es la misma que la usada al cifrar. Para resolver el ejercicio práctico planteado deberemos conocer antes como cifrar documentos con **gpg** haciendo uso de claves simétricas (*man gpg*):

Comando	Opciones / Parámetros	Ejemplo / Descripción
gpg	**[-a] [-o] -c / --symmetric**	[usuario@linux]$ **gpg --symmetric** *doc.pdf*
		[usuario@linux]$ **gpg -a** [**-o** *doc.pdf.asc*] **-c** *doc.pdf*
Cifra el documento indicado haciendo uso de la contraseña que se indica por parte del usuario en el momento de su ejecución. Con la opción "**-a**" genera un documento cifrado haciendo uso de codificación ASCII. La opción "**-o**" nos permite indicar un nombre especifico para el archivo cifrado de salida. En caso de no indicar un nombre con "**-o**" generará un archivo cifrado igual de nombre igual que el original con extensión *.gpg o *.asc dependiendo de si se ha usado la opción "**-a**" o no.		
gpg	**[-o] [-d / --decrypt]**	[usuario@linux]$ **gpg** *doc.pdf.gpg*
		[usuario@linux]$ **gpg --decrypt** *doc.pdf.gpg*
		[usuario@linux]$ **gpg -o doc.pdf --decrypt** *doc.pdf.gpg*
El resultado de ejecutar el comando **gpg** utilizando como único parámetro el nombre del archivo previamente cifrado, lo descifra y nos proporciona el archivo original. Para ello, nos solicitará la **passphrase** o contraseña introducida durante el cifrado. En el caso de que usemos la opción "**-d**" lo descifrará mostrando el resultado por la salida estándar, a no ser que especifiquemos con la opción "**-o**" un archivo concreto de salida donde se almacenará el resultado.		

***Solución Ej. Pr. 6.3.1.I.**- Cifrado con GPG mediante claves simétricas*

Para realizar el ejercicio práctico seguiremos los siguientes pasos:

(1) El usuario emisor cifra el documento (*p.e. documento.pdf*) que desea enviar al usuario destinatario haciendo uso de claves simétricas en **gpg** (*si añadimos la opción "-a" gpg generará un archivo cifrado usando codificación ASCII*). Al ejecutar la orden de cifrado se nos solicitará el **passphrase** o contraseña que se usará tanto para lo operación de cifrado como de descifrado:

```
[usuario1@linux]$ gpg --symmetric documento.pdf
[usuario1@linux]$ gpg -a --symmetric documento.pdf
Enter passphrase:
```

Esto genera un documento cifrado con extensión *.gpg, "**documento.pdf.gpg**", o *.asc, "**documento.pdf.asc**" dependiendo de si se ha hecho uso de la opción "-a" durante el cifrado. Podemos ver el resultado del cifrado, y comprobar que es ilegible completamente.

```
[usuario1@linux]$ more documento.pdf.asc
```

(2) El usuario emisor envía el documento cifrado "**documento.pdf.asc**" (*ó el "**documento.pdf.gpg**"*) al destinatario mediante el canal que crea más conveniente: email, smb, ftp, etc. Por ejemplo, en si suponemos que el destinatario comparte un directorio por Samba de manera pública con permisos de escritura, podríamos enviarlo vía smb mediante **smbclient**:

```
[usuario1@linux]$ smbclient //192.168.1.100/recurso1 \
                        -c "put doumento.pdf.asc subdirectorio1/documento.pdf.asc"
```

En el comando anterior se supone que la ip del equipo del destinatario es "192.168.1.100", que comparte un recurso compartido llamado "recurso1", y que en dicho recurso hay un subdirectorio que se llama "subdirectorio1" que es donde se almacenará el archivo resultante (*la opción "-**c**" del smbclient ejecuta el comando que le indicamos*).

¡¡Aclaración!! Para compartir un directorio por samba de manera publica tan sólo habría que añadir las siguientes líneas en su archivo de configuración "**/etc/samba/smb.conf**":

```
[root@linux]# apt-get install samba
[root@linux]# nano /etc/samba/smb.conf
# Escribir lo siguiente al comienzo del archivo smb.conf:
[recurso1]
path = /home/usuario2/nombre-directorio-a-compartir
writable = yes
public = yes
guest ok = yes
browseable = yes
…

[root@linux]# chmod -R 777 /home/usuario2/nombre-directorio-a-compartir
[root@linux]# /etc/init.d/smbd restart
```

(3) El usuario destinatario comprobará que la única forma de leer el archivo es descifrarlo previamente (*al ejecutar el siguiente comando, se nos solicitará la password que uso el emisor en*

el cifrado):

```
[usuario2@linux]$ gpg  documento.pdf.asc
```

Con esto comprobamos que en este tipo de sistemas criptográficos las claves de cifrado y descifrado son las mismas, y que la seguridad del sistema radica en como enviar la clave entre emisor y receptor, y como renovarla periódicamente para que no sea obtenida por estrategias de fuerza bruta.

6.4.- Cifrado de Información mediante Claves Asimétricas (*Clave Pública y Secreta*)

Uno de los mayores problemas de seguridad con los que se enfrenta la criptografía es cómo conseguir un método de distribución de claves seguro. Tal como se ha descrito en el apartado anterior, si una organización usa un algoritmo de cifrado simétrico para encriptar los mensajes que envía, tanto los remitentes como los destinatarios deberán conocer la clave para poder cifrar/descifrar el mensaje. Evidentemente, dicha clave deberá ser renovada periódicamente, tras lo cual se tendrá que enviar a todos los usuarios para que puedan seguir usando el algoritmo. El quid del asunto está en como enviar dichas claves de manera segura y efectiva.

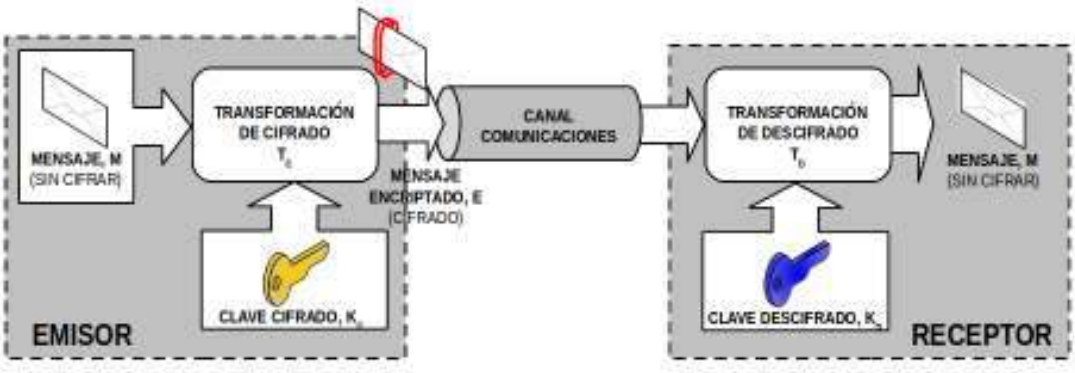
Para solucionar el problema, en 1976 Diffie y Hellman describieron el primer criptosistema de clave pública, en el que las claves de cifrado y descifrado son diferentes. Estos criptosistemas están basados en propiedades matemáticas de los números primos, que permite que cada interlocutor tenga una pareja de claves propias. De esta pareja de claves, una se denomina privada o secreta y la otra se denomina pública. La clave privada no se transmite nunca y se mantiene secreta. La clave pública, por el contrario, se puede y se debe poner a disposición de cualquiera, dado que es imposible deducir la clave privada a partir de la pública.

La propiedad fundamental de esta pareja de claves es que lo que se cifra con una de estas claves, se puede descifrar únicamente con la otra clave. Esta potente característica asimétrica es la que permite a esta tecnología servir de base para el diseño de sistemas de comunicación segura (*SSH, SFTP, HTTPS, etc.*).

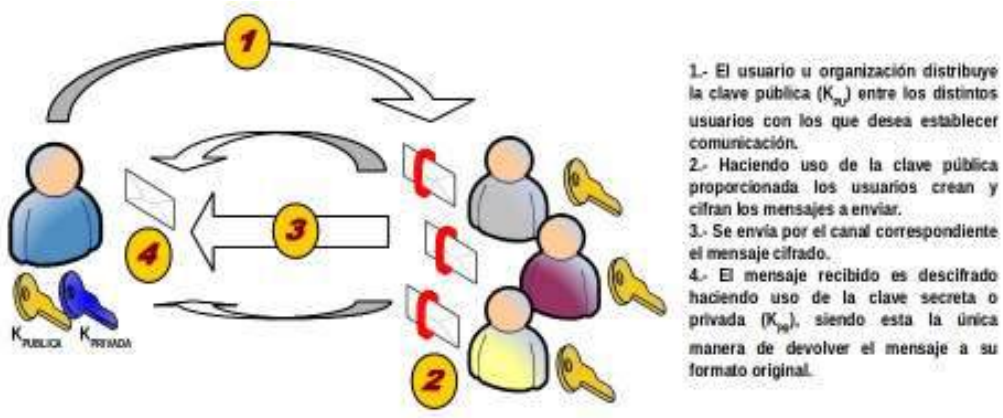
Un criptosistema, para que sea denominado de clave pública o asimétrica, debe cumplir los siguientes tres requisitos:

Requisito Nº1.- La ecuación matemática que debe definir al sistema criptográfico es la siguiente:

$E=Tc_{Kc}(M)$, el mensaje cifrado, E, es el resultado de aplicar la transformación de cifrado T_C al mensaje original M haciendo uso de la clave de cifrado K_C.
$M=Td_{Kd}(E)$, el mensaje original, M, es recuperado tras aplicar la transformación de descifrado T_D al mensaje cifrado E haciendo uso de la clave de descifrado K_D.



O lo que es lo mismo, $\mathbf{M=Td_{Kd}(Tc_{Kc}(M))}$, donde K_C es diferente de K_D ($K_C \neq K_D$). Uno de los usos más comunes de las claves asimétricas es garantizar que cierta información sólo sea leída por quien deseamos que únicamente pueda leerla. Es decir, tal como se muestra en la siguiente figura, supongamos que un conjunto de usuarios (*p.e. usuB*) envían archivos a un usuario (*p.e. usuA*), y estos quieren asegurarse de que los archivos que se envíen sólo los pueda leer él, el ***usuA***. Para dar solución a este problema, el ***usuA*** debería generar un par de claves asimétricas, $\mathbf{K_C}$ (*clave de cifrado o clave pública*) y $\mathbf{K_D}$ (*clave de descifrado o secreta*), y distribuir la clave pública entre todos aquellos que desean enviarle archivos. De esta forma, cuando los usuarios ***usuB*** hagan uso de la clave pública de ***usuA*** para el cifrado, se están asegurando que el único que podrá descifrar lo que han cifrado será él, ya que es el único que tiene la clave privada para ello.



Requisito Nº2.- Al contrario de lo que ocurre en el caso de cifrado con clave simétrica, la transformación o algoritmo a aplicar para el descifrado, T_D, no es el resultado de invertir las operaciones llevadas a cabo en la transformación de cifrado, T_C, lo que provoca, que conocer K_C no implica conocer K_D. Es decir, que a partir de la clave pública de un usuario, independientemente que la usemos para cifrar o descifrar, no puede obtenerse la clave asimétrica secreta, por lo que no debe existir ningún temor en su distribución.

En cuanto a los algoritmos que en la actualidad hacen uso de clave asimétrica, pueden diferenciarse aquellos que hacen uso de la cable pública para el cifrado o para el descifrado:

(1) Cifrado con Clave Pública:

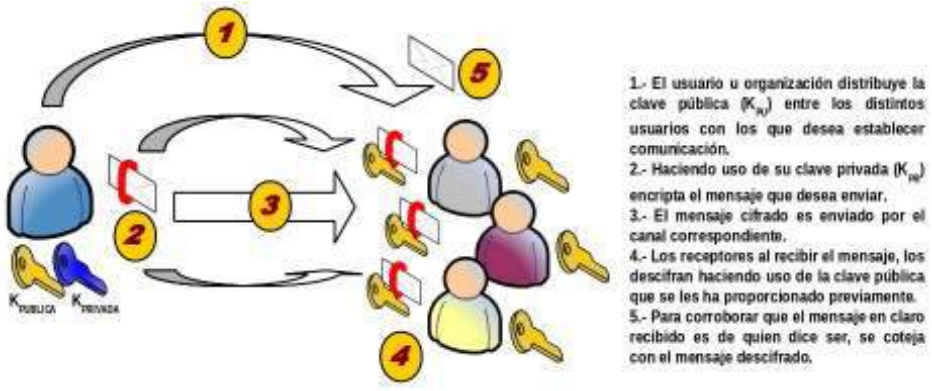
Una primera posible situación de utilización de clave asimétrica, sería la mostrada en la figura anterior, ya explicada. Tal situación se corresponde con la de un usuario, A, el cual quiere enviar información a otro usuario, B, para lo cual hace uso de la clave pública, K_{puB}, previamente suministrada por B para encriptar los datos. Por su lado, el usuario B al recibir el mensaje mandado por A utilizará su clave privada, K_{prB}, para obtener el texto en claro a partir del texto cifrado. De esta forma, cualquier usuario que desee enviar información al usuario B, deberá solicitarle previamente la clave pública, K_{puB}.

Esta modalidad de cifrado puede emplearse para proporcionar el servicio de confidencialidad, pues una vez cifrado el mensaje, el único usuario que tiene capacidad de descifrarlo es el usuario B, ya que se supone que es el único poseedor de la clave privada, K_{prB}.

La única desventaja de este esquema consiste en que cuando es preciso distribuir un mensaje cifrado entre "n" usuarios es preciso manejar "n" claves públicas: K_{puUsu1}, K_{puUsu2}, ..., K_{puUsuN}. Es decir, el emisor del mensaje debe recoger previamente las claves públicas de los "n" usuarios a los que desea enviar información, y cifrar el mensaje por separado "n" veces con cada una de las claves, para posteriormente ya enviarlos. A pesar de ello, las aplicaciones software disponibles con esta opción actualmente permiten hacerlo con relativa facilidad.

(2) Cifrado con Clave Privada o Secreta:

De manera contrapuesta a la anterior, el propietario de las claves pública (K_{PU}) y privada (K_{PR}), puede ser el interesado de enviar la información, para lo cual hace uso de la clave privada (K_{PR}) para su cifrado, de tal forma que cualquiera que conozca la clave pública correspondiente a dicho usuario podrá descifrar la información transmitida.



Tal como se puede advertir, en esta modalidad de cifrado **no se proporciona confidencialidad**, ya que supuestamente al ser la clave de descifrado de ámbito público cualquiera puede conocer el contenido del mensaje. Sin embargo, sí se puede emplear para proporcionar el servicio de **autenticación**, ya que la obtención del texto en claro sin cifrar a partir del texto cifrado es garantía de que el emisor del mensaje es el propietario de la clave pública, K_{PU}, que le corresponde.

Para saber si el mensaje obtenido a partir del texto cifrado es el texto en claro original, se ha de comparar con una copia fidedigna de éste, copia que se puede obtener por medios diferentes. Esta es la base de la técnica conocida como **firma digital**.

Uno de los algoritmos de cifrado asimétrico más conocido y usado es "RSA". Este algoritmo fue presentado por sus creadores Ron Rivest, Adi Shamir y Len Adleman en 1978, dando nombre al algoritmo las iniciales de sus apellidos, pertenecientes a un grupo de investigación del MIT.

"RSA" emplea las ventajas proporcionadas por las propiedades de los números primos cuando se aplican sobre ellos operaciones matemáticas basadas en la función módulo. En concreto, emplea la función exponencial discreta para cifrar y descifrar, y cuya inversa, el logaritmo discreto, es muy complejo de calcular.

Los cálculos matemáticos de este algoritmo emplean un número denominado el módulo público, n, que forma parte de la clave pública y que se obtiene a partir de la multiplicación de dos números primos, p y q, diferentes y grandes (*del orden de 1024 bits*) y que forman parte de la calve privada. RSA permite hacer "n" público, sin que los valores p y q pierdan su carácter secreto, debido a la dificultad que entraña la factorización de un número grande.

También cabe destacar que "RSA" es un algoritmo reversible, proporcionando las dos variantes comentadas anteriormente: cifrar con la clave pública y descifrar con la clave privada, o cifrar con la privada y descifrar con la pública.

Por tanto, la robustez del algoritmo se basa en la facilidad para encontrar dos número primos grandes y adecuados para la operación, frente a la enorme dificultad que presenta la factorización de su producto. Aunque el avance tecnológico hace cada vez más viable un posible ataque, el simple hecho de aumentar la longitud de las claves empleadas, supone un incremento en la carga de cómputo inherente lo suficientemente grande para que este tipo de ataque sea inviable.

Según todo lo anterior, la principal ventaja de este tipo de criptosistemas es que la clave secreta ya no tiene que transmitirse entre los interlocutores y tampoco es necesario tener claves diferentes para cada pareja de interlocutores, es suficiente con que cada usuario tenga su clave doble con componente pública y privada. Por lo tanto el número total de claves asimétricas en una red se reduce a "2·n", lo que se traduce en tan sólo 200 claves en una red de cien usuarios, 2000 en una de mil (*si se comparar con un sistema de claves simétricas es ridículo*), etc.

Cualquier intruso que intercepte la transmisión de un mensaje encriptado con la clave pública no podrá descifrar el contenido de la misma al no poseer la clave privada, por lo que no supone ningún peligro el enviar dicha clave por un canal inseguro, aunque esto no es del todo cierto, ya que como veremos más tarde en el apartado que trata la forma en que se distribuyen las claves a través de la red, el ataque "*Man in the Middle*" atenta contra la integridad de la clave pública.

Por contra, estos algoritmos tienen la desventaja de que no son tan eficientes a nivel de velocidad como pueden ser los basados en criptografía simétrica (*y resultan inviables en transmisiones de cantidades de datos considerables*) debido al coste computacional que conlleva el cifrado y descifrado mediante claves tan largas, aunque el mayor inconveniente que tienen se refiere a la autenticidad de las claves públicas, es decir, ¿quién nos garantiza que la clave pública de un interlocutor, que se obtiene libremente en la red, es realmente de él? ¿Qué ocurriría si alguien nos envía su clave pública afirmando ser alguien que realmente no es?

Este último problema es resuelto mediante las Autoridades de Certificación (*CA, "Certification Authority"*), que emiten certificados de las claves públicas de los usuarios firmando con su clave secreta un documento, válido por un período determinado de tiempo, que asocia el nombre distintivo de un usuario con su clave pública.

¡¡Importante!! Tal como se habrá podido advertir a través de las explicaciones de este apartado teórico las claves asimétricas sirven para dos cosas:

(1) Cifrar mediante la **clave pública** del usuario al que deseo enviarle un archivo o información de forma **confidencial**. La **confidencialidad** esta asegurada en esta situación ya que al hacer uso de la clave pública del destinatario nos estamos asegurando de que el único que va a poder descifrarlo es él, ya que él es el único que tiene la clave secreta o privada asociada a la clave pública que podría descifrarlo. Recordar que la pareja de claves asimétricas se caracterizan porque lo que se cifra con una de ellas, tan sólo puede ser descrifrado por la otra.
En relación a esta operación de cifrado u ocultación de información con la finalidad de asegurar una confidencialidad siempre debe realizarse mediante la clave pública y no con la clave privada o secreta, ya que si cifráramos con esta última, no estaría asegurada la confidencialidad al poder descifrarlo cualquiera que tuviera la parte pública.

(2) Firmar mediante la **clave secreta o privada** con la finalidad de enviar un archivo o información a otro usuario para que éste pueda comprobar su **autenticidad** e **integridad** haciendo uso de la clave pública.

Ej. Práctico 6.4.1: Cifrado de Archivos con Claves Asimétricas con GPG

Haciendo uso de gpg (*GNU Privacy Guard*), indicar paso a paso, todos los pasos que serían necesarios llevar a cabo para poder transferir un documento cifrado (*p.e. documento.doc*) entre tres usuarios llamados "**usulinux1**", "**usulinux2**" y "**usulinux3**", de tal forma que el usuario "usulinux1" pueda enviárselo a los otros dos de manera segura haciendo uso de claves asimétricas, garantizando la **confidencialidad de la información**. Deberán crearse los tres usuarios en el caso de que no existan.

Emisor	Receptores	Distribución de claves Públicas	Documento a Enviar Cifrado	Tipo Criptografía
usulinux1	**usulinux2** **usulinux3**	**Exportar** **Importar**	**documento.doc**	**Asimétrica - GPG**

Para poder resolver el ejercicio práctico propuesto deberán tenerse en cuenta las siguientes opciones del comando "**gpg**":

Comando y Opciones	Ejemplo / Descripción
gpg --gen-key	[usuario@linux]$ **gpg --gen-key**
Genera un par de claves asimétricas (*clave pública y privada*).	
gpg [-a] [-o] --export	[usuario@linux]$ **gpg -a** [**-o** *clave.asc*] \ **--export** *ID_K*$_{Pub}$
	[usuario@linux]$ **gpg -a --export** *ID_K*$_{Pub}$ > *clave.asc*
Nos permite exportar la clave pública para poderla distribuir entre otros usuarios. Esto permitirá a	

<table>
<tr><td colspan="3">esos usuarios poder mandarnos archivos cifrados garantizando la confidencialidad, o enviarles nosotros archivos garantizando autenticidad e integridad, tal como veremos en el apartado de firma digital. Si le añadimos la opción "-a" la generará haciendo uso de codificación ASCII. Con "-o" podremos especificar un archivo de salida con el resultado de la exportación, aunque puede resultar más cómodo redireccionar la salida (>).</td></tr>
<tr><td colspan="2">gpg --import</td><td>[usuario@linux]$ gpg --import clave.asc</td></tr>
<tr><td colspan="3">Nos permite importar la clave pública de otro usuario que previamente él haya exportado. Una vez importada podremos cifrar archivos para que el propietario de la clave pública sea él único que pueda descifrarlos (confidencialidad), o corroborar la autenticidad e integridad de los archivos que hayan sido cifrados con su correspondiente clave privada.</td></tr>
<tr><td colspan="2">gpg [-a] [-o] --encrypt \
-r/--recipient ID_K_{Pub}</td><td>[usuario@linux]$ gpg -a [-o doc.odt.asc] \
-r ID_K_{Pub} -r ID_K_{Pub ...} --encrypt doc1.odt</td></tr>
<tr><td colspan="3">Nos permite cifrar un documento destinado a uno o varios usuarios (--recipient) haciendo uso de las claves públicas que previamente hayamos importado.</td></tr>
<tr><td>gpg</td><td>--list-keys
-k
--list-public-keys
-K
--list-secret-keys</td><td>[usuario@linux]$ gpg --list-public-keys
[usuario@linux]$ gpg -k
[usuario@linux]$ gpg --list-secret-keys
[usuario@linux]$ gpg -K
[usuario@linux]$ gpg --list-keys</td></tr>
<tr><td colspan="3">Nos permiten listar las claves disponibles en nuestro keyring o anillo de claves (todas, solamente las claves públicas o solamente las privadas). Mediante la opción -k o --list-public-keys podemos listar nuestras claves públicas, y mediante -K o --list-secret-keys podemos listar nuestras claves secretas.</td></tr>
<tr><td>gpg</td><td>--delete-key
--delete-secret-key
--delete-secret-and-public-key</td><td>[usuario@linux]$ gpg --delete-key ID_K_{Pub}
[usuario@linux]$ gpg --delete-secret-key ID_K_{Priv}</td></tr>
<tr><td colspan="3">Nos permiten eliminar las claves de nuestro almacén de claves: keyring o anillo de claves.</td></tr>
<tr><td>gpg</td><td>--edit-key</td><td>[usuario@linux]$ gpg --edit-key ID_Key
>fpr (fingerprint)
>sign (firmar y validar la clave)
>check (comprobación clave)
>...</td></tr>
<tr><td colspan="3">Nos permite editar una clave haciendo referencia al identificador, email o nombre de usuario introducidos durante la generación de las claves asimétricas. Por ejemplo, nos puede resultar útil si queremos autofirmar una clave de otro usuario con la finalidad de avalarla y confiar en ella.</td></tr>
<tr><td colspan="2" rowspan="3">gpg [-o] [-d/--decrypt]</td><td>[usuario@linux]$ gpg documento.pdf.gpg</td></tr>
<tr><td>[usuario@linux]$ gpg --decrypt documento.pdf.gpg</td></tr>
<tr><td>[usuario@linux]$ gpg -o doc-original.pdf \
--decrypt documento.pdf.gpg</td></tr>
<tr><td colspan="3">El resultado de ejecutar el comando gpg utilizando como único parámetro el nombre del archivo previamente cifrado, lo descifra y nos proporciona el archivo original, siempre y cuando gpg</td></tr>
</table>

detecte que disponemos de la clave secreta asociada a la clave pública con la que se cifró el documento. Para hacer uso de la parte secreta o privada de la pareja de claves asimétricas, **gpg** nos solicitará el **passphrase** o contraseña introducida durante la generación de las claves asimétricas (*gpg –gen-key*).
En el caso de que usemos la opción "**-d**" lo descifrará mostrando el resultado por la salida estándar, a no ser que especifiquemos con la opción "**-o**" un archivo concreto de salida donde se almacenará el resultado.

Solución Ej. Pr. 6.4.1.I.- Cifrado de Archivos mediante la Clave Pública

Suponiendo que los usuarios ya existen, seguiremos los siguientes pasos para la implementación de la práctica:

(1) El usuario "usulinux2" y "usulinux3" deberán generar un pareja de claves asimétricas, exportar la parte pública y hacerla llegar a "usulinux1". De esta forma "usuliux1" podrá utilizar las claves públicas de "usulinux2" y "usulinux3" para cifrar el documento a enviar y asegurar de esta forma la confidencialidad, ya que los únicos que podrán descifrarlo serán "usulinux2" y "usulinux3" al ser los únicos que disponen de la clave privada o parte asimétrica necesaria para poderlo descifrar. En concreto, generaremos claves de tipo RSA con una longitud de 1024 bits (*para evitar costes computacionales*) y sin caducidad, como características más importantes (*se muestra a continuación como hacerlo para el usuario "usulinux2", teniendo que hacer lo mismo con "usulinux3"*):

```
[usulinux2|usulinux3@linux]$ gpg --gen-key
Por favor seleccione tipo de clave deseado:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sólo firmar)
   (4) RSA (sólo firmar)
Su elección: 1
las claves RSA pueden tener entre 1024 y 4096 bits de longitud.
¿De qué tamaño quiere la clave? (2048) 1024
Por favor, especifique el período de validez de la clave.
         0 = la clave nunca caduca
      <n>  = la clave caduca en n días
      <n>w = la clave caduca en n semanas
      <n>m = la clave caduca en n meses
      <n>y = la clave caduca en n años
¿Validez de la clave (0)? 0
...
Necesita un identificador de usuario para identificar su clave. El programa
construye el identificador a partir del Nombre Real, Comentario y Dirección
de Correo Electrónico de esta forma:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"
```

```
Nombre y apellidos: UsuLinux2
Dirección de correo electrónico: usulinux2@gmail.com
Comentario: El usuario 2
Ha seleccionado este ID de usuario:
   " UsuLinux2 (El usuario 2) <usulinux2@gmail.com>"

¿Cambia (N)ombre, (C)omentario, (D)irección o (V)ale/(S)alir? V
Introduzca una contraseña: ---
Repita frase contraseña: ---

gpg: /home/usulinux2/.gnupg/trustdb.gpg: se ha creado base de datos de confianza
gpg: clave 8F5DE799 marcada como de confianza absoluta
claves pública y secreta creadas y firmadas.

pub     1024R/8F5DE799 2014-01-27
        Huella de clave = C21E D55A F899 29AB 91A7  2247 D25C 03A0 8F5D E799
uid             UsuLinux2 (El usuario 2) <usulinux2@gmail.com>
sub     1024R/36C73123 2014-01-27
```

¡¡Aclaración!! La contraseña solicitada al final del proceso será requerida cada vez que el usuario vaya a hacer uso de la clave privada. De esta forma, aunque el usuario haya dejado una sesión abierta, otro usuario no podrá suplantarlo y hacer uso de su clave privada al no conocer dicha clave. También cabría señalar que durante la generación de las claves podemos encontrarnos con el siguiente mensaje:

Es necesario generar muchos bytes aleatorios. Es una buena idea realizar alguna otra tarea (trabajar en otra ventana/consola, mover el ratón, usar la red y los discos) durante la generación de números primos. Esto da al generador de números aleatorios mayor oportunidad de recoger suficiente entropía.

No hay suficientes bytes aleatorios disponibles. Por favor, haga algún otro trabajo para que el sistema pueda recolectar más entropía (se necesitan 284 bytes más).

Éste nos indica que para que la generación de las claves sea lo más aleatoria posible hay que aumentar la entropía del sistema, o lo que es lo mismo, aumentar el trabajo realizado por el sistema, y de esta forma al encontrarse el equipo más estresado la clave será más enrevesada. Para ello podemos hacer búsquedas mediante el comando **find**, instalar algún programa mediante **apt-get install**, o cualquier otra cosa que se nos ocurra.

Para comprobar que se ha generado correctamente la clave, podemos listar las claves disponibles en nuestro keyring o almacén de claves (*-k, lista las claves públicas y -K las secretas*):

```
[usulinux2@linux]$ gpg -k
[usulinux2@linux]$ gpg --list-public-keys
[usulinux2@linux]$ gpg -K
```

```
[usulinux2@linux]$ gpg --list-secret-keys
```

Tras generar la pareja de claves asimétricas, exportaremos la clave publica haciendo referencia al identificador de la clave generada: nombre y apellidos indicados durante la generación, "**UsuLinux2**", mediante el identificador propiamente dicho, "**8F5DE799**", o mediante la cuenta de email introducidos al generar las claves asimétricas "**usulinux2@gmail.com**" (*la opción "-a" es opcional, permitiendo exportar la clave utilizando codificación ASCII*).

```
[usulinux2@linux]$ gpg --export "UsuLinux2" > clave_publica_usulinux2.gpg
[usulinux2@linux]$ gpg --export 8F5DE799 > clave_publica_usulinux2.gpg
[usulinux2@linux]$ gpg -a --export "UsuLinux2" > clave_publica_usulinux2.asc
[usulinux2@linux]$ gpg -a --export 8F5DE799 > clave_publica_usulinux2.asc
```

Por último, la haremos llegar al usuario "usulinux1" mediante algún canal de confianza (*email, ftp, smb, scp, etc.*). Al igual que en el ejercicio práctico anterior, suponiendo que se la hacemos llegar vía smb (*repasarlo dicho ejercicio para más aclaraciones*):

```
[usulinux2@linux]$ smbclient //192.168.1.100/recurso1 \
                        -c "put clave_publica_usulinux2.asc"
```

(2) El usuario "usulinux1" importará las claves públicas que han sido distribuidas por los usuarios "usulinux2" y "usulinux3". Tras importarlas listaremos las claves públicas disponibles (*opción -k o --list-public-keys*):

```
[usulinux1@linux]$ gpg --import clave_publica_usulinux2.asc
gpg: clave 8F5DE799: clave pública " UsuLinux2
 <usulinux2@gmail.com>" importada
gpg: Cantidad total procesada: 1
gpg: importadas: 1  (RSA: 1)
[usulinux1@linux]$ gpg --import clave_publica_usulinux3.asc
...
```

```
[usulinux1@linux]$ gpg -k
[usulinux1@linux]$ gpg --list-public-keys
```

¡¡Advertencia!! Aunque es posible distribuir las claves públicas mediante las opciones **--export** e **--import** del comando **gpg**, suele ser mucho más cómodo y fácil de gestionarlas mediante el uso de alguno de los servidores públicos PGP, tal como se mostrará en el siguiente ejercicio práctico. La única pega que presenta el uso de servidores PGP como almacén de claves públicas es que requerimos de una conexión con Internet para subir y descargar las claves.

(3) Una vez que el usuario "usulinux1" dispone de las claves públicas de los usuarios a los que desea enviar una información de manera confidencial, cifrará (*--encrypt*) el documento a enviar (*p.e. documento.doc*) haciendo uso de esas claves públicas importadas, haciendo referencia a los identificadores (*nombres de usuarios, identificador o email*) de la claves públicas (*p.e. -r UsuLinux2 y -r UsuLinux3*). El resultado del siguiente comando generará el documento cifrado "**documento.doc.asc**" el cual tan sólo podrá ser descifrado por aquellos usuarios que dispongan de

las claves privadas asociadas a la claves públicas que se han usado durante el cifrado (*pareja de claves asimétricas*), garantizando de esta forma la confidencialidad en la comunicación, al no poder descifrarlo nadie más:

```
[usulinux1@linux]$ gpg -a -r UsuLinux2 -r  UsuLinux3 --encrypt documento.doc
[usulinux1@linux]$ more documento.doc.asc
```

¡¡Aclaración!! Al cifrar o encriptar el documento original nos aparecerá el siguiente mensaje advirtiendonos de que la clave pública que estamos usando no es seguro que sea del usuario que dice ser quien es (ver al final del capítulo el ataque Man in the middle), y que es posible que pueda pertenecer a un intermediario:

```
[usulinux1@linux]$ gpg -a -r UsuLinux2 -r  UsuLinux3 --encrypt documento.doc
...
No es seguro que la clave pertenezca a la persona que se nombra en el
identificador de usuario. Si *realmente* sabe lo que está haciendo,
puede contestar sí a la siguiente pregunta.
...
```

Para evitar la aparición de ese mensaje de desconfianza podemos validar las claves públicas anteriores correspondientes a los usuarios usulinux2 y usulinux3 editando la clave pública. Por ejemplo, para validar y autofirmar la clave pública del usuario usulinux2:

```
[usulinux1@linux]$ gpg --edit-key UsuLinux2
gpg> check
gpg> sign
gpg> quit
```

A continuación podemos volver a encriptar el documento y comprobar que la desconfianza a desaparecido.

(4) El usuario "uslinux1" hará llegar el documento cifrado a los destinatarios "usulinux2" y "usulinux3" (*email, smb, ftp, etc.*). Para ello puede hacerse uso del mismo canal que se uso para enviar las claves públicas.

(5) Los usuarios "usulinux2" y "usulinux3" destinatarios comprobarán que pueden descifrar el documento recibido:

```
[usulinux2@linux]$ gpg documento.doc.asc
```

¡¡Aclaración!! El software **gpg** (GNU Privacy Guard) almacena de manera independiente para cada usuario las claves pertenecientes a ese usuario (*públicas y privadas*) en un almacén de claves llamado comúnmente anillo de claves o "**keyring**". De esta forma, durante la operación de descifrado el software **gpg** comprueba si el usuario dispone de la parte privada asociada a alguna de las claves públicas (*claves asimétricas*) con las que el emisor cifró el documento analizando su **keyring**. En caso de que sí que dispongamos de la parte secreta de la pareja de claves asimétricas, **gpg** nos lo descifrará.

También podría comprobarse que otro usuario que recibiera el documento cifrado no podría descifrarlo al no disponer en su anillo de claves de la clave privada o secreta emparejada con alguna de las claves públicas que se usó durante el cifrado:

```
[usulinux4@linux]$ gpg documento.doc.asc
gpg: ...
gpg: descifrado fallido: clave secreta no disponible
```

¡¡Observación!! Como ha podido observarse, para la realización del ejercicio práctico anterior se ha hecho uso de un método clásico de transferencia de archivos (*smb, ftp, email, etc.*) para la distribución de la clave o parte pública de la pareja de claves asimétricas. Más adelante, en varios ejercicios prácticos veremos como distribuir esa clave pública de una manera más comoda mediante el uso de del servicio ofrecido por multitud de servidores públicos PGP (*Pretty Good Privacy*) de claves públicas.

Ej. Práctico 6.4.2: Distribución de Claves Públicas mediante un Servidor

Si en el ejercicio anterior hemos visto como generar un par de claves asimétricas y distribuir la parte pública a otros usuarios para que pueda ser utilizada por estos mediante el uso de el software habitual de transferencia de archivos (*email, smb, etc.*), en el presente ejercicio práctico veremos otra posible estrategia para su distribución mediante servidores públicos encargados de almacenar claves públicas.

Es decir, con la finalidad de facilitar la búsqueda y aumentar la confianza en el uso de claves públicas para el cifrado (*confidencialidad*) y comprobación de firmas digitales (*autenticación, integridad y no repudio*), surgieron una red de servidores públicos PGP (*Pretty Good Privacy*) distribuidos por todo el mundo con la finalidad de almacenar toda clave pública de todo aquel usuario que lo desee. Al respecto indicar que todos los servidores públicos PGP son clones, lo que garantiza que subida la clave a uno de ellos, puede obtenerse a partir de cualquier otro.

Con la finalidad de comprobar su facilidad de uso, distribuiremos la clave pública de un usuario que disponga de una pareja de claves asimétricas (*p.e. usulinux2*). Aunque puede subirse la clave pública a distribuir a través de un formulario HTML disponible en la web de los servidores PGP, también es posible realizarse la distribución mediante comandos **gpg** específicos, tal como veremos a continuación.

En relación al servidor PGP encargado de almacenar las claves públicas, haremos uso del servidor de claves públicas PGP español **pgp.rediris.es** (*http://www.rediris.es/keyserver/*), pero tenemos la opción de usar otros muchos otros servidores públicos (*p.e. http://pgp.mit.edu/, http://pgp.zdv.uni-mainz.de, http://keyserver.pgp.com, certserver.pgp.com, etc.*).

Comando y Opciones	Ejemplo / Descripción
gpg --keyserver *Servidor* **--send-keys** *ID_Key*	[usuario@linux]$ **gpg --keyserver pgp.rediris.es \ --send-keys** *ID_Key*
Envía la clave pública especifica al servidor de claves para que sea almacenada. También nos permite enviar la revocación de una clave que queramos dar de baja, previamente revocada.	

gpg --keyserver *Servidor* **--recv-keys** *ID_Key*	[usuario@linux]$ **gpg --keyserver pgp.rediris.es** \ **--recv-keys** *ID_Key*
Nos permite obtener del servidor de claves la clave pública especificada.	
gpg [-o] --gen-revoke *ID_Key*	[usuario@linux]$ **gpg -o** *revoc.out* **--gen-revoke** *ID_Key*
gpg --import	[usuario@linux]$ **gpg --import** *revoc.out*
La opción "**--gen-revoke**" nos permite revocar o dar de baja una clave que queramos dejar de usar. Mediante "**--import**" podremos importar el archivo de revocación. Una vez revocada la clave, para darla definitivamente de baja en un servidor público de claves públicas será necesario volver a enviar la clave revocada (*--send-keys*).	

Solución Ej. Pr. 6.4.2.I.- Distribución de Claves Públicas mediante un Servidor

A modo de ejemplo, siguiendo con los usuarios utilizados en los problemas anteriores, supondremos que el usuario "usulinux2" quiere distribuir su clave pública para que otros usuarios (*p.e. "usulinux1"*) pueda tanto enviarle archivos confidencialmente como corroborar la firma de documentos realizada con la parte secreta de la pareja de claves asimétricas. Para ello el usuario "usulinux2" listará las claves públicas que posee en su keyring, y haciendo uso del ID de la clave pública que se desea distribuir, la enviará al servidor de claves públicas de **pgp.rediris.es**:

```
[usulinux2@linux]$ gpg --list-public-keys
/home/usulinux1/.gnupg/pubring.gpg
...
pub   1024R/28E91A59 2014-02-05
uid                  Usuario Linux 2 (El usuario 2) <usulinux2@gmail.com>
sub   1024R/34B5DA57 2014-02-05
```

```
[usulinux2@linux]$ gpg --keyserver pgp.rediris.es --send-keys 28E91A59
```

En el caso de que el usuario "usulinux1" quiera obtenerla tan sólo tendrá que ejecutar el siguiente comando:

```
[usulinux1@linux]$ gpg --keyserver pgp.rediris.es --recv-keys 28E91A59
[usulinux1@linux]$ gpg -k
[usulinux1@linux]$ gpg --list-public-keys
```

¡¡Observación!! Una vez realizado el ejercicio práctico podrás comprobar accediendo a la página Web de alguno de los servidor PGP (*p.e.* http://pgp.rediris.es, http://pgp.mit.edu, *etc.*) que la clave pública subida al servidor se encuentra allí disponible para cualquiera que quiera obtenerla. Advertir entonces que una vez subida la clave pública a uno de los servidores PGP públicos existentes, la clave es distribuida entre el resto de servidores PGP que existen en el mundo, para que se accesible fácilmente desde cualquier lugar.

Destacar, que el uso de este tipo de servidores PGP es muy habitual en el mundo del software libre para poder corroborar a través de la firma digital que el software que me instalo fue desarrollado por quien dice ser que es, y que no me estoy instando un software malicioso. Este aspecto referente a la firma digital lo comprenderemos más en detalle en el siguiente apartado de la práctica. De igual forma podrían indicarse otros ejemplos prácticos reales de utilidad.

Ej. Práctico 6.4.3: Cifrar con Claves Distribuidas mediante Servidores PGP

Indica **qué comandos tendrá que ejecutar cada uno de los usuarios que intervienen en la comunicación** (*empleado1*, *empleado2 y empleado3*) en el supuesto caso de que el **empleado3** quiera enviar de manera **confidencial** un archivo llamado "**datos-privados-3.pdf**" a los otros dos usuarios **empleado1** y **empleado2**. Ninguno de los tres usurarios dispone de claves asimétricas, por lo que los usuarios que las requieran, y sólo aquellos que las requieran deberán generarlas. Para la distribución de las claves haremos uso de el servidor de claves públicas PGP "**pgp.rediris.es**".

Emisor	Receptores	Distribución de claves Públicas	Documento a Enviar Cifrado	Tipo Criptografía
empleado3	**empleado1** **empleado2**	**pgp.rediris.es**	**datos-privados-3.pdf**	**Asimétrica - GPG**

Solución Ej. Pr. 6.4.3.I.- Cifrado mediante Claves Públicas de un Servidor PGP

Si repasamos los ejercicios prácticos anteriores advertiremos que los pasos a seguir para la resolución del ejercicio práctico propuesto serían los siguientes:

¡¡Observación!! En el caso de no disponer de las cuentas de usuario **empleado1**, **empleado2** y **empleado3** deberemos crearlas en el sistema haciendo uso de los comando **useradd** y **passwd**.

```
[root@linux]# useradd -m -d /home/empleadoX -s /bin/bash empleadoX
[root@linux]# passwd empleadoX
```

(1) Los empleados **empleado1** y **empleado2** generarán un par de claves asimétricas GPG (*clave*

pública y clave privada), y subirán al servidor de claves públicas **pgp.rediris.es** su correspondiente clave pública. Estas claves públicas le permitirán al **empleado3** cifrar documentos que tan sólo podrán descifrar ellos al ser ellos quien tienen la clave privada necesaria para ello, garantizando de esta forma la confidencialidad en la comunicación. Por contra, el empleado **empleado3** no necesita generar las claves asimétricas, a no ser que quiera autofirmar con una clave privada las claves públicas que recibirá de los otros dos usuarios. Supondremos que los identificadores de las claves asimétricas generadas por los empleados **empleado1** y **empleado2** tras ejecutar el comando **gpg --gen-key** son **1111111A** y **2222222B** respectivamente.

```
[empleado1|empleado2@linux]$ gpg --gen-key
[empleado1@linux]$ gpg --keyserver pgp.rediris.es --send-keys 1111111A
[empleado2@linux]$ gpg --keyserver pgp.rediris.es --send-keys 2222222B
```

(2) El empleado **empleado3** se descargará las claves públicas anteriores y cifrará el documento a enviar a esos usuarios.

```
[empleado3@linux]$ gpg --keyserver pgp.rediris.es --recv-keys 1111111A
[empleado3@linux]$ gpg --keyserver pgp.rediris.es --recv-keys 2222222B
[empleado3@linux]$ gpg --list-public-keys
[empleado3@linux]$ gpg -a -r 1111111A -r 2222222B --encrypt datos-privados-3.pdf
```

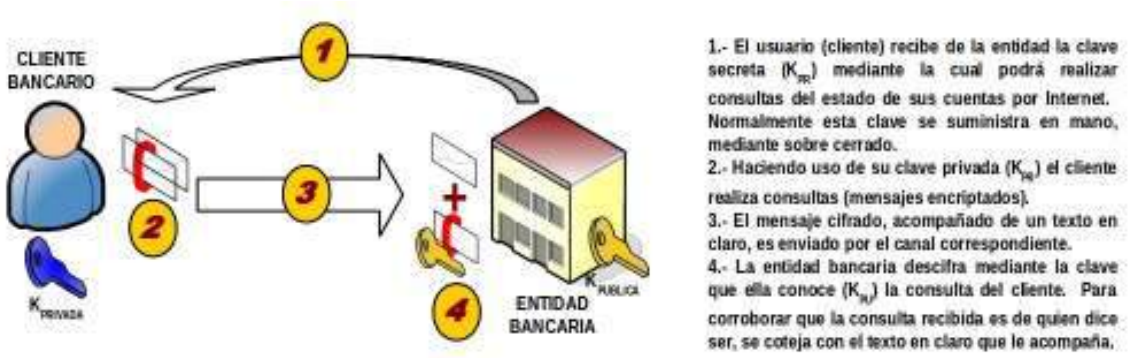
(3) El archivo cifrado resultante, **datos-privados-3.pdf.asc**, se le hará llegar a los empleados empleado1 y empleado2. Estos lo descifrarán mediante **gpg** obteniendo como resultado el archivo original **datos-privados-3.pdf**:

```
[empleado1@linux]$ gpg datos-privados-3.pdf.asc
[empleado2@linux]$ gpg datos-privados-3.pdf.asc
```

6.5.- Firma Digital

Tal como se ha expuesto en el punto anterior, el uso de algoritmos de cifrado asimétrico o de clave pública, hoy en día esta muy implantado bajo el nombre de firma digital o electrónica.

Un ejemplo muy común son las consultas y operaciones bancarias que pueden realizarse hoy en día mediante el uso de Internet, sin la necesidad de tener que acercarse a la entidad bancaria más cercana, garantizando una comodidad para el usuario indiscutible, claro esta, a no ser que la operación que se desee realizar sea extraer dinero en efectivo (*en la siguiente figura se muestra de manera esquemática cual es el proceso*).



Es decir, el banco hace uso de la firma digital para corroborar que quien esta realizando la consulta u operación es quien dice ser (*autenticación*), ya que da por hecho, que quien tiene la clave privada con la que se realiza la firma (*cifrado del extracto*) es el usuario a quién se le suministro.

De forma similar, en la actualidad se dan multitud de situaciones donde se hace uso de esta metodología, como puede ser la declaración de la renta por Internet, contratación de seguros, etc.

Tal como se puede advertir a través de la figura anterior, un paso crucial para poder llevar a cabo todas estas operaciones desde Internet, es la forma en que se generan las claves y se distribuyen. En la realidad, la generación de las claves se lleva a cabo haciendo uso de un software informático, el cual no deja rastro de la clave creada, lo cual es vital, sobre todo en el caso de la clave privada o secreta. Una vez generada, se suele suministrar al usuario en mano, mediante el uso de sobre cerrado, o vía telefónica, mediante el uso de teleoperadoras virtuales. Debido a la importancia de este aspecto, en el siguiente punto siguiente se detallan los problemas que pueden darse, siendo el tipo de ataque más extendido el "*Man in the Middle*".

Concretando más, una firma o huella digital no son más que un bloque de caracteres que acompañan a un documento, y que sirven para:

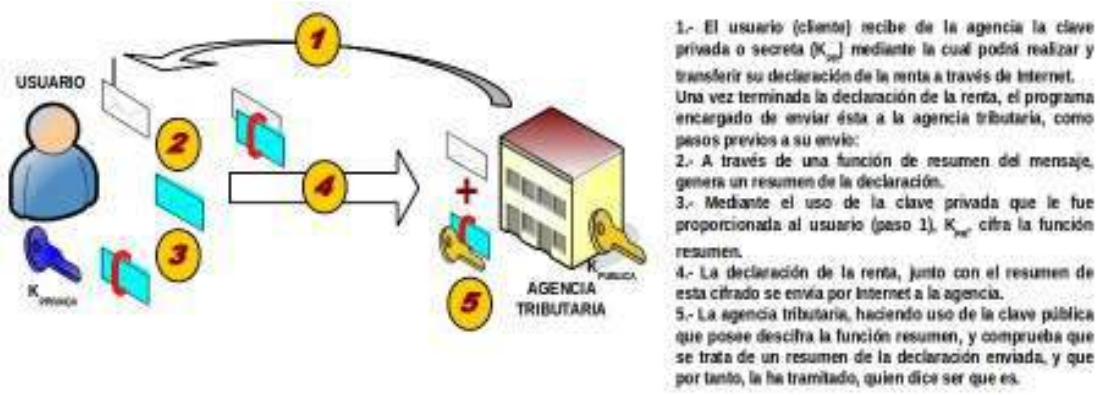
(a) Comprobación de la autenticidad. El receptor pueda acreditar la identidad del emisor. En el caso anterior, a la entidad bancaria le permite corroborar que el que llevo a cabo la consulta es quien dice ser que es.
(b) Evitar el repudio. Que el transmisor no pueda repudiar a posteriori el contenido del mensaje. La firma digital permite desbaratar cual tipo de reclamación o contradicción en un proceso. Por ejemplo, si un usuario tras hacer su declaración de la renta vía Internet, luego quiere alegar que él no fue quien envío dicha declaración, se puede comprobar si esta diciendo la verdad o no mediante el uso de la firma digital. Es decir, si firmó el usuario, luego no puede ser que diga que el no fue.
(c) Integridad de la información. Mediante la comprobación de la firma digital el receptor puede comprobar que la información que recibe es realmente la misma información integra que fue enviada por el emisor. En caso de que se haya producido una manipulación o alteración intermedia de la información, la comprobación de la firma nos avisará de ello.
(d) Por último, al basarse la firma digital en el uso de claves asimétricas, es decir, que la clave para el cifrado no es la misma que la de descifrado, el receptor no puede suplantar en ningún caso al emisor y falsificar el mensaje.

No obstante, el método de cifrar el mensaje completo con la firma digital a veces es innecesario, pues lo único que se requiere es poder enviar documentos normales pero firmados para

acreditar su procedencia. A este fin se aplica el uso de las conocidas como funciones "Hash", o **funciones de resumen del mensaje**, que proporcionan un extracto del mensaje sobre el que se aplica la clave privada antes de enviar el texto en claro, junto con el resumen cifrado. Una función de resumen, "**fr**", debe cumplir una serie de requisitos:

I) Dado un mensaje, "m", debe resultar fácil calcular su función de resumen, fr(m).
II) Sea cual sea el tamaño del mensaje, "m", el tamaño resultante de su función de resumen será constante.
III) Dado fr(m) es computacionalmente inviable hallar "m".
IV) Dado m es computacionalmente inviable hallar una fr' tal que fr'(m)=fr(m).

Cuando el receptor recoge el mensaje, pasa a descifra con la clave pública del emisor el resumen. Tras ello aplica la función de resumen al texto en claro y comprueba que coincide con el resumen recibido.



Dos de los algoritmos de resumen de mensaje más usados son MD5 y SHA. El primero fue diseñado en 1992 por Ron Rivest, uno de los mayores expertos en criptografía actuales, y se usa, entre otros, en el conocido protector de correo electrónico PGP (*Private Good Privacy*). El algoritmo SHA fue diseñado por la NSA (*Nacional Security Agency*) en 1993, para ser incluido en el estándar DSS (*Digital Signatura Standard*). Al contrario que los algoritmos de cifrado propuestos por esta organización (*por ejemplo, DES, Data Encryption Standard*), SHA se considera seguro y libre de puertas traseras, ya que favorece a los propios intereses de la NSA que esto sea así. Produce firmas de 160 bits, a partir de bloques de 512 bits del mensaje original. También se utiliza en las versiones recientes de PGP y en clientes de correo como Microsoft Outlook Express.

El proceso de la firma digital lo realiza un programa (*por ejemplo, un cliente de correo electrónico*) que aplica un algoritmo resumen, MD5 o SHA, sobre el texto a firmar, obteniendo el extracto. Este, cuya longitud suele oscilar entre 128 y 160 bits, se somete a continuación al cifrado, RSA ó DSS, mediante la clave secreta del autor.

En resumen, el propósito de la firma digital es emular, superando sus prestaciones, a su antecesora en el cargo, es decir, a la firma manual de toda la vida.

Ej. Práctico 6.5.1: Firma de Documentos con Texto en Claro mediante GPG

Indica **qué comandos tendrá que ejecutar cada uno de los usuarios que intervienen en la comunicación** (*empleado1, empleado2 y empleado3*) en el supuesto caso de que el **empleado2** quisiera enviar un documento llamado "**datos-publicos-2.pdf**" **firmado con el texto en claro sin cifrar**. Ninguno de los tres usurarios dispone de claves asimétricas GPG, por lo que los usuarios que las requieran, y sólo aquellos que las requieran deberán generarlas. Para la distribución de las claves públicas haremos uso un servidor de claves públicas PGP, p.e. **pgp.rediris.es**.

Advertir durante el ejercicio práctico, que al hacer uso de la clave privada para la realización de la firma, se nos solicitará el **passphrase** o contraseña que se introdujo durante su generación (*gpg --gen-key*).

Emisor	Receptores	Distribución de claves Públicas	Documento a Enviar Firmado	Tipo Criptografía
empleado2	**empleado1** **empleado3**	**pgp.rediris.es**	**datos-publicos-2.pdf**	**Asimétrica - GPG**

Para la resolución del ejercicio práctico haremos uso de las siguientes opciones del software **gpg**:

Comando y Opciones	Ejemplo / Descripción
gpg [-a] [-o] [-u *ID_K*$_{Priv}$**]** **--clearsign**	[usuario@linux]$ **gpg -a --clearsign** *fichero.odt*
La opción "**--clearsign**" nos permite firmar un documento sin cifrar. Esta opción nos puede resultar útil cuando nuestro único interés sea el de comprobar la **autenticación** del emisor y la **integridad** de la información recibida, no siendo importante la **confidencialidad**. En el caso de que el usuario disponga de más de una pareja de claves asimétricas, y por tanto, más de una posible clave secreta para la realización de la firma, deberemos indicar con "**-u**" el identificador de clave a usar. En caso contrario, elegirá la primera que se creó.	
gpg [--verify]	[usuario@linux]$ **gpg** *fichero.odt.asc*
	[usuario@linux]$ **gpg --verify** *fichero.odt.asc*
El uso del comando **gpg** pasándole como único parámetro el archivo firmado (*p.e. firmado en claro mediante --clearsign*) nos permite verificar la firma y obtener el archivo original. Si usamos la opción "**--verify**" únicamente comprueba la firma digital, pero no proporciona el archivo original.	
gpg [-o] [-d/--decrypt]	[usuario@linux]$ **gpg** *fichero.odt.asc*
	[usuario@linux]$ **gpg --decrypt** *fichero.odt.asc*
El uso del comando **gpg** pasándole como único parámetro el archivo firmado y cifrado (*p.e. firmado en claro mediante --sign y cifrado con --encrypt*) nos permite verificar la firma (*se necesita la clave pública del emisor*) y descifrar el documento para obtener el archivo original (*debemos poseer la clave secreta asociada a la clave pública con la que se cifró*).	

Comprueba la firma digital y lo descifra si es necesario (*--sign --encrypt*)

Solución Ej. Pr. 6.5.1.I.- Cómo Firmar Documentos con el Texto en Claro

Antes de dar la solución a este ejercicio práctico aclarar que su pretensión no es garantizar una confidencialidad en la comunicación, ya que el archivo a enviar se envía con el texto en claro, sin cifrar, sino que lo que se pretende al firmar es permitir a los receptores el poder comprobar quien es el autor del mensaje (autenticidad), además de verificar su integridad. Por tanto, la firma digital nos va a permitir garantizar la **autenticación** e **integridad** en la comunicación.

¡¡Observación!! En el caso de no disponer de las cuentas de usuario **empleado1**, **empleado2** y **empleado3** deberemos crearlas en el sistema haciendo uso de los comando **useradd** y **passwd**.

```
[root@linux]# useradd -m -d /home/empleadoX -s /bin/bash empleadoX
[root@linux]# passwd empleadoX
```

Los pasos a seguir para la resolución serían los siguientes:

(1) El empleado **empleado2** será el único que requerirá de un par de claves asimétricas (*clave pública y clave privada*). La clave privada la usará para firmar el documento a enviar, y la clave pública les permitirá a los otros dos empleados, **empleado1** y **empleado3**, comprobar la firma (*autenticación e integridad*). Por tanto, el empleado2 generará las claves asimétricas, subirá la parte pública al servidor PGP para su fácil distribución y firmará el documento. Supondremos que el identificador de las claves asimétricas generadas por empleado2 tras ejecutar **gpg --gen-key** es **2222222B**:

```
[empleado2@linux]$ gpg --gen-key
[empleado2@linux]$ gpg --keyserver pgp.rediris.es --send-keys 2222222B
[empleado2@linux]$ gpg -a -u 2222222B --clearsign datos-publicos-2.pdf
```

(2) Los empleados **empleado1** y **empleado3** se descargarán la clave pública anterior del servidor PGP. Esta clave les permitirá comprobar la firma (*autenticación e integridad*) y eliminar la firma digital del documento recibido obteniendo como resultado el fichero original sin firmar. Antes de nada también puede comprobarse que el documento recibido no esta cifrado, y que contiene la firma al final (*p.e. tail -20*):

```
[empleado1|empleado3@linux]$ more datos-publicos-2.pdf.asc | tail -20
[empleado1|empleado3@linux]$ gpg --keyserver pgp.rediris.es --recv-keys 2222222B
[empleado1|empleado3@linux]$ gpg --list-public-keys
[empleado1|empleado3@linux]$ gpg --verify datos-publicos-2.pdf.asc
[empleado1|empleado3@linux]$ gpg datos-publicos-2.pdf.asc
[empleado1|empleado3@linux]$ evince datos-publicos-2.pdf
```

¡¡Observación!! Puede comprobarse que si el fichero recibido por los receptores fuera modificado, al comprobar la firma se nos advertiría de una fallo de autenticación e integridad en la información

recibida. Para ello, puede firmarse un simple archivo de texto "datos.txt" y modificarse antes de la comprobación de la firma.

Ej. Práctico 6.5.2: Firmar un Documento con la Firma Separada

En la misma línea que el ejercicio práctico anterior, indica **qué comandos tendrá que ejecutar cada uno de los usuarios que intervienen en la comunicación** (*empleado1*, *empleado2 y empleado3*) en el supuesto caso de que el **empleado2** quisiera enviar un documento llamado "**datos-publicos-2.pdf**" **firmado con el texto en claro sin cifrar**, pero con **la firma por separado**. Ninguno de los tres usurarios dispone de claves asimétricas GPG, por lo que los usuarios que las requieran, y sólo aquellos que las requieran deberán generarlas. Para la distribución de las claves públicas haremos uso un servidor de claves públicas PGP, p.e. **pgp.rediris.es**.

Advertir durante el ejercicio práctico, que al hacer uso de la clave privada para la realización de la firma, se nos solicitará el **passphrase** o contraseña que se introdujo durante su generación (*gpg --gen-key*).

Emisor	Receptores	Distribución de claves Públicas	Documento a Enviar Firmado	Tipo Criptografía
empleado2	**empleado1** **empleado3**	**pgp.rediris.es**	**datos-publicos-2.pdf**	**Asimétrica - GPG**

Para la resolución del ejercicio práctico haremos uso de las siguientes opciones del software **gpg**:

Comando y Opciones	Ejemplo / Descripción
gpg [-a] [-o] [-u *ID_*K_{Priv}**] [-s] -b/--detach-sign**	[usuario@linux]$ **gpg -a --detach-sign** *fichero.odt*
	[usuario@linux]$ **gpg -a --sign --detach-sign** *fichero.odt*
La opción "**-b**" o "**--detach-sign**" genera una función resumen firmada por separado. De esta forma, haciendo uso de esta opción, al receptor habría que enviarle dos archivos: el fichero en claro, más la función resumen firmada mediante la cual el receptor hará la comprobación correspondiente de **autenticación** e **integridad**.	
gpg [--verify]	[usuario@linux]$ **gpg** *fichero.odt.asc*
	[usuario@linux]$ **gpg --verify** *fichero.odt.asc*
El uso del comando **gpg** pasándole como único parámetro el archivo firmado (*p.e. firmado en claro mediante --clearsign*) nos permite verificar la firma y obtener el archivo original. Si usamos la opción "**--verify**" únicamente comprueba la firma digital, pero no proporciona el archivo original.	
gpg [-o] [-d/--decrypt]	[usuario@linux]$ **gpg** *fichero.odt.asc*
	[usuario@linux]$ **gpg --decrypt** *fichero.odt.asc*
El uso del comando **gpg** pasándole como único parámetro el archivo firmado y cifrado (*p.e. firmado en claro mediante --sign y cifrado con --encrypt*) nos permite verificar la firma (*se necesita la clave pública del emisor*) y descifrar el documento para obtener el archivo original	

(*debemos poseer la clave secreta asociada a la clave pública con la que se cifró*).
Comprueba la firma digital y lo descifra si es necesario (*--sign --encrypt*)

Solución Ej. Pr. 6.5.2.I.- Cómo Firmar un Documento con la Firma Separada

Al igual que en el ejercicio práctico anterior aclarar que la pretensión de la firma digital no es garantizar una confidencialidad en la comunicación, ya que el archivo se envía con el texto en claro, sin cifrar, sino que lo que se pretende al firmar es permitir a los receptores puedan comprobar quien es el autor del mensaje (*autenticidad*), al mismo tiempo que nos permite verificar su integridad. Por tanto, la firma digital nos va a permitir garantizar la **autenticación** e **integridad** en la comunicación.

A diferencia del ejercicio práctico anterior donde el documento enviado ya integraba la firma al final, en este caso práctico la firma la enviaremos por separado. Por esta razón, a los receptores habrá que entregarles dos documentos: el documento original y el documento que contiene la firma. Este último documento no es más que un resumen del documento original firmado con la clave secreta o privada del emisor. De esta forma, el receptor al comprobar la firma estará comprobando que el resumen que se obtenga del documento original recibido coincide con el resumen firmado enviado, verificando de esta forma la autenticidad e integridad de la información.

¡¡Observación!! En el caso de no disponer de las cuentas de usuario **empleado1**, **empleado2** y **empleado3** deberemos crearlas en el sistema haciendo uso de los comando **useradd** y **passwd**.

```
[root@linux]# useradd -m -d /home/empleadoX -s /bin/bash empleadoX
[root@linux]# passwd empleadoX
```

Los pasos a seguir para la resolución serían los siguientes:

(1) El empleado **empleado2** será el único que requerirá de un par de claves asimétricas (*clave pública y clave privada*). La clave privada la usará para firmar el documento a enviar, y la clave pública les permitirá a los otros dos empleados, **empleado1** y **empleado3**, comprobar la firma (*autenticación e integridad*). Por tanto, el empleado2 generará las claves asimétricas, subirá la parte pública al servidor PGP para su fácil distribución y firmará el documento. Supondremos que el identificador de las claves asimétricas generadas por **empleado2** tras ejecutar **gpg --gen-key** es **2222222B**:

```
[empleado2@linux]$ gpg --gen-key
[empleado2@linux]$ gpg --keyserver pgp.rediris.es --send-keys 2222222B
[empleado2@linux]$ gpg -a -u 2222222B --sign --detach-sign datos-publicos-2.pdf
[empleado2@linux]$ more datos-publicos-2.pdf.asc
```

(2) El empleado2 deberá hacer llegar dos documentos a los otros dos empleados: **datos-publicos-2.pdf** y **datos-publicos-2.pdf.asc**. Los empleados **empleado1** y **empleado3** se descargarán la clave pública anterior del servidor PGP. Esta clave les permitirá comprobar la firma (*autenticación e integridad*):

```
[empleado1|empleado3@linux]$ gpg --keyserver pgp.rediris.es --recv-keys 2222222B
[empleado1|empleado3@linux]$ gpg --list-public-keys
[empleado1|empleado3@linux]$ gpg --verify datos-publicos-2.pdf.asc
[empleado1|empleado3@linux]$ gpg datos-publicos-2.pdf.asc
[empleado1|empleado3@linux]$ evince datos-publicos-2.pdf
```

¡¡Observación!! Puede comprobarse que si el fichero recibido por los receptores fuera modificado, al comprobar la firma se nos advertiría de una fallo de autenticación e integridad en la información recibida. Para ello, puede firmarse un simple archivo de texto "datos.txt" y modificarse antes de la comprobación de la firma.

Ej. Práctico 6.5.3: Cifrar y Firmar Simultáneamente un Documento

Indica **qué comandos tendrá que ejecutar cada uno de los usuarios que intervienen en la comunicación** (*empleado1, empleado2 y empleado3*) en el supuesto caso de que el **empleado1** quisiera enviar un documento llamado "**datos-1.pdf**" **cifrado y firmado simultáneamente**. Ninguno de los tres usurarios dispone de claves asimétricas, por lo que los usuarios que las requieran, y sólo aquellos que las requieran deberán generarlas. Para la distribución de las claves públicas haremos uso del servidor de claves públicas "**pgp.rediris.es**". Advertir durante el ejercicio práctico, que al hacer uso de la clave privada para la realización de la firma, se nos solicitará el **passphrase** o contraseña que se introdujo durante su generación (*gpg --gen-key*).

Emisor	Receptores	Distribución de claves Públicas	Documento a Enviar Cifrado y Firmado	Tipo Criptografía
empleado1	**empleado2** **empleado3**	**pgp.rediris.es**	**datos-1.pdf**	**Asimétrica - GPG**

Para poder resolver el ejercicio práctico propuesto deberemos conocer las siguientes opciones del comando **gpg**:

Comando y Opciones	Ejemplo / Descripción
gpg [-a] [-o] [-u *ID_K*$_{Priv}$**]** **[-r** *ID_K*$_{Pub}$**] -s/--sign [--encrypt]**	[usuario@linux]$ **gpg -a -o** *fichero.doc.gpg* **-s** *fichero.doc*
	[usuario@linux]$ **gpg -a -r** *ID_K*$_{Pub}$ \ **--sign --encrypt** *fichero.doc*
La opción "**--sign**" firma un documento, aunque combinada con **--symmetric** (*claves simétricas*) o **--encrypt** (*claves asimétricas*) nos permite cifrar y firmar un documento. La opción "**-u**" nos permite especificar una clave secreta diferente de la configurada por defecto para la realizar la firma, y con "**-o**" podemos indicar el fichero que almacenará el resultado de salida.	
gpg [--verify]	[usuario@linux]$ **gpg** *fichero.odt.asc*
	[usuario@linux]$ **gpg --verify** *fichero.odt.asc*
El uso del comando **gpg** pasándole como único parámetro el archivo firmado (*p.e. firmado en*	

claro mediante --clearsign) nos permite verificar la firma y obtener el archivo original. Si usamos la opción "**--verify**" únicamente comprueba la firma digital, pero no proporciona el archivo original.	
gpg [-o] [-d/--decrypt]	[usuario@linux]$ **gpg** *fichero.odt.asc*
	[usuario@linux]$ **gpg --decrypt** *fichero.odt.asc*
El uso del comando **gpg** pasándole como único parámetro el archivo firmado y cifrado (*p.e. firmado en claro mediante --sign y cifrado con --encrypt*) nos permite verificar la firma (*se necesita la clave pública del emisor*) y descifrar el documento para obtener el archivo original (*debemos poseer la clave secreta asociada a la clave pública con la que se cifró*). Comprueba la firma digital y lo descifra si es necesario (*--sign --encrypt*)	

Solución Ej. Pr. 6.5.3.I.- Cómo Cifrar y Firmar un Documento Simultáneamente

Mediante este ejercicio práctico pretendemos mostrar como garantizar la **confidencialidad** (*mediante el cifrado de la información*), la **autenticidad** y la **integridad** (*mediante la firma digital*) en una comunicación.

¡¡Observación!! En el caso de no disponer de las cuentas de usuario **empleado1**, **empleado2** y **empleado3** deberemos crearlas en el sistema haciendo uso de los comando **useradd** y **passwd**.

```
[root@linux]# useradd -m -d /home/empleadoX -s /bin/bash empleadoX
[root@linux]# passwd empleadoX
```

Basándonos en todos los ejercicios prácticos anteriores, los pasos a seguir para la resolución serían los siguientes:

(1) Todos los empleados (*empleado1, empleado2 y empleado3*) requerirán de un par de claves asimétricas (*clave pública y clave privada*). El **empleado1** utilizará su clave privada para firmar el documento a enviar, y las claves públicas de los otros dos usuarios para cifrarlo. En cambio, los otros dos empleados **empleado2** y **empleado3**, utilizarán su clave privada para descifrar el documento que reciban, y la clave pública del **empleado1** para poder comprobar la firma (*autenticación e integridad*). Por tanto, los tres empleados generarán las parejas de claves asimétricas y subirán la parte pública al servidor PGP para su fácil distribución. Supondremos que los identificadores de las claves asimétricas generadas por los usuarios anteriores tras ejecutar **gpg --gen-key** son **1111111A**, **2222222B** y **3333333C** respectivamente:

```
[empleado1|empleado2|empleado3@linux]$ gpg --gen-key
[empleado1@linux]$ gpg --keyserver pgp.rediris.es --send-keys 1111111A
[empleado2@linux]$ gpg --keyserver pgp.rediris.es --send-keys 2222222B
[empleado3@linux]$ gpg --keyserver pgp.rediris.es --send-keys 3333333C
[empleado1@linux]$ gpg --keyserver pgp.rediris.es --recv-keys 2222222B
[empleado1@linux]$ gpg --keyserver pgp.rediris.es --recv-keys 3333333C
[empleado2@linux]$ gpg --keyserver pgp.rediris.es --recv-keys 1111111A
[empleado3@linux]$ gpg --keyserver pgp.rediris.es --recv-keys 1111111A
```

```
[empleado1|empleado2|empleado3@linux]$ gpg --list-secret-keys
[empleado1|empleado2|empleado3@linux]$ gpg --list-public-keys
```

(2) El **empleado1** cifrará y firmará el documento a enviar **datos-1.pdf**. El resultado **datos-1.pdf.asc** lo hará llegar a los destinatarios vía email, ftp, smb, etc.

```
[empleado1@linux]$ gpg -a -u 1111111A -r 2222222B -r 3333333C --sign --encrypt  datos-1.pdf
```

(3) Los empleados **empleado2** y **empleado3** descifrarán el archivo recibido (*confidencialidad*) y comprobarán la firma (*autenticación e integridad*):

```
[empleado2|empleado3@linux]$ gpg datos-1.pdf.asc
[empleado2|empleado3@linux]$ evince datos-1.pdf
```

Ej. Práctico 6.5.4: Repaso de Formas de Firmar Digitalmente un Documento

A modo de repaso de todos los ejercicios prácticos vistos en este capítulo, haciendo uso de la herramienta **gpg** haz que un usuario que disponga de una pareja de claves asimétricas (*gpg --gen-key*) firme un documento (*p.e. midoc.pdf*) con su clave secreta y se lo haga llegar a un usuario que disponga de la parte pública, con la cual verificará la firma, y descifrará en caso de ser necesario. Advertir durante el ejercicio práctico, que al hacer uso de la clave privada para la realización de la firma, se nos solicitará el **passphrase** o contraseña que se introdujo durante su generación (*gpg --gen-key*).

Para ello, haremos uso de las cuentas de usuario creadas en los ejercicios anteriores (*usulinux1, usulinux2, etc.*) ya que usulinux2 debería disponer de una pareja de claves asimétricas (clave *secreta y clave pública*), lo cual le permite firmar un documento mediante su clave secreta. Después usulinux1 comprobará la firma mediante el uso de la clave pública del usuario usulinux2 que previamente importó (*usulinux2 distribuyo las claves haciendo uso de un servidor PGP de claves públicas*).

En concreto firmaremos un documento (*p.e. "**declaracion-usu2.txt**"*) para los siguientes casos:

(1) El usuario "usulinux2" firmará un documento (*p.e. declaracion-usu2.txt*) con el texto en claro (*--clearsign*), y se lo hará llegar al usuario "usulinux1". Éste último, "usulinux1", con la finalidad de comprobar la **autenticidad** del emisor, la **integridad** del archivo recibido y evitar el posible **repudio** del emisor verificará la firma. Advierte que el documento recibido por "usulinux1" es un texto legible compuesto por el contenido del fichero original seguido de la firma digital. Comprueba igualmente que si el texto recibido fuera modificado, la comprobación de la firma nos advertiría de la falta de integridad y autenticidad.

(2) El usuario "usulinux2" creará una función resumen firmada del documento a enviar (*--sign -b*), de tal forma que al usuario "usulinux1" deberán llegarle dos documentos para poder comprobar la firma: I) el documento original con el texto en claro (*p.e. declaracion-usu2.txt*), y II) el resumen firmado (*la firma va por separado*), mediante los cuales podrá comprobar la **autenticidad** e **integridad** del archivo, además de evitar el **repudio** del emisor.

(3) El usuario "usulinux2" enviará el documento (*p.e. declaracion-usu2.txt*) al usuario "usulinux1"

firmado (*--sign*) con su clave secreta y cifrado (*--encrypt*) con la clave pública del usuario usulinux1. De esta forma se garantiza una comunicación segura donde se cumplen las premisas de **confidencialidad**, **autenticidad**, **integridad** y **no repudio**.

Resumiendo todas las opciones vistas en los ejercicios prácticos anteriores, la sintaxis del comando **gpg** a utilizar para la realización de la firma será la siguiente:

Comando y Opciones	Ejemplo / Descripción
gpg [-a] [-o] [-u *ID_K*$_{Priv}$**] [-r** *ID_K*$_{Pub}$**] -s/--sign**	[usuario@linux]$ **gpg -a -o** *fichero.doc.gpg* **-s** *fichero.doc*
	[usuario@linux]$ **gpg -a -r** *ID_K*$_{Pub}$ \ **--sign --encrypt** *fichero.doc*
La opción "**--sign**" firma un documento, aunque combinada con **--symmetric** (*claves simétricas*) o **--encrypt** (*claves asimétricas*) nos permite cifrar y firmar un documento. La opción "**-u**" nos permite especificar una clave secreta diferente de la configurada por defecto para la realizar la firma, y con "**-o**" podemos indicar el fichero que almacenará el resultado de salida.	
gpg [-a] [-o] [-u *ID_K*$_{Priv}$**] --clearsign**	[usuario@linux]$ **gpg -a --clearsign** *fichero.odt*
La opción "**--clearsign**" nos permite firmar un documento sin cifrar. Esta opción nos puede resultar útil cuando nuestro único interés sea el de comprobar la **autenticación** del emisor y la **integridad** de la información recibida, no siendo importante la **confidencialidad**. En el caso de que el usuario disponga de más de una pareja de claves asimétricas, y por tanto, más de una posible clave secreta para la realización de la firma, deberemos indicar con "**-u**" el identificador de clave a usar. En caso contrario, elegirá la primera que se creó.	
gpg [-a] [-o] [-u *ID_K*$_{Priv}$**] [-s] -b/--detach-sign**	[usuario@linux]$ **gpg -a --detach-sign** *fichero.odt*
	[usuario@linux]$ **gpg -a --sign --detach-sign** *fichero.odt*
La opción "**-b**" o "**--detach-sign**" genera una función resumen firmada por separado. De esta forma, haciendo uso de esta opción, al receptor habría que enviarle dos archivos: el fichero en claro, más la función resumen firmada mediante la cual el receptor hará la comprobación correspondiente de **autenticación** e **integridad**.	
gpg [--verify]	[usuario@linux]$ **gpg** *fichero.odt.asc*
	[usuario@linux]$ **gpg --verify** *fichero.odt.asc*
El uso del comando **gpg** pasándole como único parámetro el archivo firmado (*p.e. firmado en claro mediante --clearsign*) nos permite verificar la firma y obtener el archivo original. Si usamos la opción "**--verify**" únicamente comprueba la firma digital, pero no proporciona el archivo original.	
gpg [-o] [-d/--decrypt]	[usuario@linux]$ **gpg** *fichero.odt.asc*
	[usuario@linux]$ **gpg --decrypt** *fichero.odt.asc*
El uso del comando **gpg** pasándole como único parámetro el archivo firmado y cifrado (*p.e. firmado en claro mediante --sign y cifrado con --encrypt*) nos permite verificar la firma (*se necesita la clave pública del emisor*) y descifrar el documento para obtener el archivo original (*debemos poseer la clave secreta asociada a la clave pública con la que se cifró*).	

Comprueba la firma digital y lo descifra si es necesario (*--sign --encrypt*)

Solución Ej. Pr. 6.5.4.I.- Repaso de Cómo Firmar Digitalmente un Documento

Para resolver el ejercicio propuesto, se describirán paso a paso los tres casos solicitados de manera independientemente:

(1) Para el uso de la firma con el texto en claro.

1a) El usuario "usulinux2" creará el documento a enviar y lo firmará. Después se lo hará llegar al usuario "usulinux1" (*ftp*, *smb*, *email*, *etc.*):

```
[usulinux2@linux]$ echo "Yo usulinux2 declaro que ..." >  declaracion-usu2.txt
[usulinux2@linux]$ gpg -a -o dec-usu2.txt.asc -u usulinux2@gmail.com \
                          --clearsign declaracion-usu2.txt
```

1b) Una vez que el usuario "usulinux1" posee el documento firmado, aunque puede ver el contenido ya que se encuentra sin cifrar, con la finalidad de comprobar la autenticidad e integridad del archivo deberá comprobar la firma:

```
[usulinux1@linux]$ more  dec-usu2.txt.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Yo usulinux2 declaro que ...
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.12 (GNU/Linux)

iJwEAQECAAYFAlL5bqgACgkQEaa4bijpGlknEgQAqVCBGwaPwa1CWEwE2y9pBl6d
BzcdWhe5or5y6ox25GmCSCapvl4Vgc6Hp3OkFaGTu2taEyqlkGEiCklMbQ1jTim1
SWyVKyfdR+UnVaQ036TYTkghkbQsBxXURn3eLQuSjly16Lk2EMrdcAMGKfwGUuCf
7PpySg5dmSNRNcnyp/E=
=vtKR
-----END PGP SIGNATURE-----
```

```
[usulinux1@linux]$ gpg --verify  dec-usu2.txt.asc
...
gpg: Firma correcta de ...
```

```
[usulinux1@linux]$ gpg  dec-usu2.txt.asc
[usulinux1@linux]$ more dec-usu2.txt
Yo usulinux2 declaro que ...
```

¡¡Sugerencia!! Prueba a modificar el texto en claro que contiene el archivo firmado, y comprueba que al verificar la firma se detecta la falta de integridad.

(2) También tenemos la opción de firmar por separado con la clave secreta una función resumen del documento original, la cual servirá al destinatario (*tras generar él también la función resumen*), la autenticidad e integridad del archivo recibido:

2a) El usuario "usulinux2" creará el documento a enviar y creara la firma por separado (*--sign -b*). Al no hacer uso de la opción "-o" gpg generará el archivo resumen con la firma separada con el nombre del archivo seguido de la extensión *.asc (*opción -a*). Después le hará llegar al usuario "usulinux1" (*ftp, smb, email, etc.*) tanto el archivo orginal (p.e. *declaracion-usu2.txt*) como la función resumen firmada (p.e. *declaracion-usu2.txt.asc*):

```
[usulinux2@linux]$ echo "Yo usulinux2 declaro que ..." >  declaracion-usu2.txt
[usulinux2@linux]$ gpg -a -u usulinux2@gmail.com --sign -b declaracion-usu2.txt
```

2b) Una vez que el usuario "usulinux1" posee los dos archivos, tan sólo le queda comprobar la firma para corroborar que el archivo original que ha recibido es de quien dice ser que es, y que además se encuentra integro:

```
[usulinux1@linux]$ gpg declaracion-usu2.txt.asc
...
gpg: Firma correcta de ...
```

¡¡Sugerencia!! Prueba a modificar el texto en claro que contiene el archivo firmado, y comprueba que al verificar la firma se detecta la falta de integridad.

(3) Por último veremos la posibilidad de firmar y cifrar el documento simultáneamente. Para ello, suponemos que el usuario "usulinux1" se ha creado un par de claves asimétricas, ha exportado su clave pública, se la ha hecho llegar al usuario "usulinux2" y éste la ha importado.

3a) El usuario "usulinux2" creará el documento a enviar, lo firmará haciendo uso de la clave secreta seleccionada con la opción "-u" (*suponemos que puede tener más de una clave secreta, sino no hace falta indicar dicha opción*) y la cifrará haciendo uso de la clave pública importada del usuario "usulinux1" (*la opción -r nos permite indicar el identificador de la clave pública, el nombre o email del destinatario*). Después le hará llegar al usuario "usulinux1" (*ftp, smb, email, etc.*) el archivo resultante:

```
[usulinux2@linux]$ echo "Yo usulinux2 declaro que ..." >  declaracion-usu2.txt
[usulinux2@linux]$ gpg -a -o declaracion-usu2.txt.asc \
-u usulinux2@gmail.com -r usulinux1@gmail.com --sign --encrypt declaracion-usu2.txt
```

3b) Una vez que el usuario "usulinux1" posee el archivo firmado y cifrado, tan sólo le queda comprobar la firma para corroborar que el archivo original que ha recibido es de quien dice ser que es, y que además se encuentra integro. El uso directo de gpg nos verifica la autenticidad e integridad, al mismo tiempo que nos descifra el documento (*para descifrarlo se hará uso de la clave secreta por lo que se nos solicitará el passphrase introducido por el usuario usulinux1 en la generación de la pareja de claves asimétrica*):

```
[usulinux1@linux]$ gpg declaracion-usu2.txt.asc
...
gpg: Firma correcta de ...
```

```
[usulinux1@linux]$ more  declaracion-usu2.txt
```

¡¡Importante!! Al verificar la firma los ejemplos anteriores habremos podido advertir que gpg nos avisa de que la clave pública que usamos para comprobar la firma no es seguro que sea de quien dice ser que es.

```
[usulinux1@linux]$ gpg --verify declaracion-usu2.txt.asc
gpg: Firmado el mar 11 feb 2014 17:31:53 CET usando clave RSA ID 28E91A59
gpg: Firma correcta de "Usuario Linux 2 (El usuario 2) <usulinux2@gmail.com>"
gpg: ATENCIÓN: ¡Esta clave no está certificada por una firma de confianza!
...
```

Es decir, que para gpg la clave pública no es de confianza hasta que no se demuestre lo contrario (*ver ataque Man in the Middle al final del capítulo para entender la desconfianza*). Una posibilidad para configurar las claves públicas como confiables, es que uno mismo las edite, y las autofirme:

```
[usulinux1@linux]$ gpg --edit-key usulinux2@gmail.com
gpg> check
uid  Usuario Linux 2 (El usuario 2) <usulinux2@gmail.com>
sig!3        28E91A59 2014-02-05  [autofirma]

gpg> sign
...
¿Firmar de verdad? (s/N) s
gpg> quit
```

Ahora ya podremos comprobar que para **gpg** la clave de usulinux2 ha pasado a ser una clave de confianza:

```
[usulinux1@linux]$ gpg --verify declaracion-usu2.txt.asc
gpg: Firmado el mar 11 feb 2014 17:31:53 CET usando clave RSA ID 28E91A59
gpg: comprobando base de datos de confianza
gpg: 3 dudosa(s) necesarias, 1 completa(s) necesarias,
modelo de confianza PGP
gpg: nivel: 0  validez:   1  firmada:   1  confianza: 0-, 0q, 0n, 0m, 0f, 1u
gpg: nivel: 1  validez:   1  firmada:   0  confianza: 1-, 0q, 0n, 0m, 0f, 0u
gpg: Firma correcta de "Usuario Linux 2 (El usuario 2) <usulinux2@gmail.com>"
```

¡¡Conclusión!! Por lo que hemos visto a través del ejercicio práctico existen tres posibles a la hora de firmar un documento:

(I) Firmar el documento dejando el contenido que contiene en claro (*opción –clearsign*). Nos permite garantizar autenticidad, integridad y no repudio en la comunicación.
(II) Realizar la firma por separado. Al receptor se le envía el documento original más el fichero resumen firmado resultante (*opción --detach-sign*). Nos permite garantizar autenticidad, integridad y no repudio en la comunicación.
(III) Firmar y cifrar simultáneamente (*opción --sign --encrypt*). Nos permite garantizar autenticación, integridad, no repudio y confidencialidad en la comunicación.

6.6.- Entidades de Certificación (CA): Certificado de Clave Pública

Tal como veremos en el siguiente apartado de la práctica, uno de los riesgos de seguridad al que estamos expuestos al utilizar este tipo de tecnología es que a consecuencia de un ataque "Man in the Middle", recibamos una clave pública de un usuario diferente del esperado, siendo todo ello transparente para nosotros, y con el grave problema de que todo lo que cifremos con ella, creyendo que lo va a leer el usuario deseado, podrá ser descifrado previamente por un desconocido. Con la finalidad de minimizar este grave problema surgen las entidades certificadoras, o también conocidas como CAs. Su labor consiste en certificar que la cable pública que recibidos es realmente de quien dice ser que es. Para ello, las CA firman mediante su clave privada la clave pública de los usuarios, y haciendo uso de la clave pública de la CA comprobamos su autenticidad.

Por tanto, dentro del actual esquema de redes abiertas (*Internet*), las autoridades de certificación (**CA**) asumen el papel de fedatarios públicos, verificando la identidad de usuarios y entidades. Para ello proporcionan un certificado digital. En la actualidad, la certificación en Internet usa el estándar X.509 v3 de certificados. Esta norma permite lo siguiente:

1.- Firmar digitalmente los mensajes de tal forma que el receptor pueda descifrarlos y tener acceso a su contenido, garantizando la autenticidad y el no repudio.
2.- Cifrar la información del certificado de tal forma que sólo el receptor pueda descifrarlos y tener acceso a su contenido, garantizando su integridad y confidencialidad.
3.- Autenticar la identidad de acceso de los usuarios de redes.

Un certificado digital contiene, básicamente, la clave pública de la persona o entidad para la que se emite, junto con información propia, y todo ello firmado electrónicamente por la autoridad de certificación. En España la autoridad de certificación más extendida es la llamada Autoridad de Certificación Española, **ACE**.

¡¡Observación!! En este capítulo o práctica no implementaremos una Autoridad de Certificación (CA) ya que su implementación la dejamos para la práctica correspondiente a la implementación de protocolos en modo seguro (*p.e. HTTPS*) donde comprenderemos más en detalle su importancia.

6.7.- Problema de Distribución de Claves

Un punto débil de todas las técnicas criptográficas modernas es la gestión de las claves. Por gestión se entiende el conjunto de procesos encaminados a la generación, implantación, distribución y revocación de claves, procesos que deberán responder a los criterios de oportunidad, eficacia y

discreción propios de tan delicada cuestión.

El tipo de método empleado para llevar a cabo la gestión de las claves es diferente según el tipo de criptografía utilizada: simétrica o asimétrica.

(1) Distribución de Claves en Algoritmos de Cifrado Simétricos:

A la hora de distribuir las claves privadas o secretas de las que se hace uso para el cifrado simétrico, se puede optar por una distribución simétrica o asimétrica:

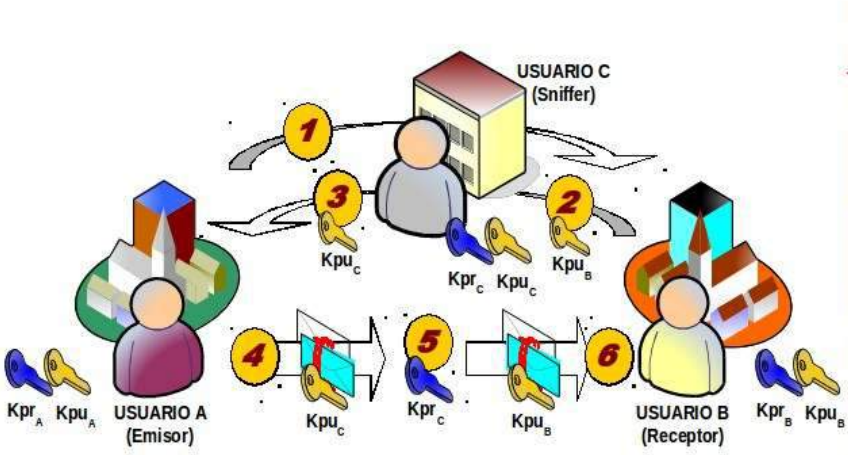
a) Distribución Simétrica. Para permitir una cierta agilidad a la par que seguridad en el proceso de utilización de estas claves, es preciso establecer un mecanismo de varios niveles. Así, será normal que en cada sesión de comunicación la clave de cifrado se modifique. Para permitir su envío por el canal de comunicación, pues la otra opción es el envío manual por un canal seguro, se utilizará otra clave para cifrar estas claves de sesión. Estas claves de nivel superior sí que se pueden transmitir manualmente por canales seguros (*por carta certificada, en mano, mediante el uso de valija diplomática, etc.*).

b) Distribución Asimétrica. En este caso, se utiliza un mecanismo de clave pública para cifrar las claves que se envían. La razón de no usar simplemente cifrado asimétrico radica en que éste consume bastantes más recursos de cómputo que el simétrico. Uno de los métodos más usados son los llamados sobres digitales. Dichos receptáculos incorporan una clave pública. De este modo lo primero que hará el emisor será cifrar el texto en claro con una clave simétrica privada. Seguidamente se cifrará la clave privada con la clave pública del sobre y se enviará el texto cifrado y el sobre al destinatario. El receptor procederá a descifrar el sobre con su clave asimétrica privada y obtendrá la clave simétrica con la que podrá descodificar el texto cifrado. Este tipo de sistemas, se denomina también sistema de comunicación seguro híbrido, al hacer uso durante la comunicación tanto de claves simétricas (*usadas durante la comunicación*) como asimétricas (*para pasarse las claves simétricas que utilizarán durante la comunicación*). Actuales algoritmos como HTTPS, SSH, … hacen todos ellos uso de este tipo de estrategias, con la finalidad de minimizar el tiempo computacional que implicaría el uso de claves asimétricas durante toda la comunicación.

(2) Distribución de Claves en Algoritmos de Cifrado Asiméticos:

En los métodos asimétricos, cada interlocutor sólo a de poseer un par de claves (*privada y pública*) independientemente del número de sistemas con los que se comunique. La gestión de este tipo de claves presenta la debilidad conocida como "*el ataque del intermediario*". Supongamos un supuesto escenario compuesto por tres usuarios (A, B y C): un primer usuario, A, interesado en establecer una comunicación con un segundo usuario, B, mientras un tercer usuario, C, trata de espiarles.

Cuando A le solicite a B su clave pública, Kpu_B, C se interpone, obteniendo la clave de B, y remitiendo a A su propia clave pública, Kpu_C, aspecto que pasa desapercibido para A. Cuando A codifique el mensaje, C lo interceptará de nuevo, descodificándolo con su clave privada propia, y empleando Kpu_B lo recodificará y lo reenviará a B.



1.- El usuario o entidad A, se pone en contacto con la B, para que le ceda su clave pública, Kpu_B, con la finalidad de establecer una comunicación confidencial.
2.- El usuario o entidad B, le remite su clave publica, pero es interceptada por el sniffer, usuario C.
3.- El usuario C, intercambia la clave pública de B, Kpu_B, por la suya, Kpu_C, y se la reenvía a A, sin que este se de cuenta.
4.- Creyendo A, que esta en posesión de la clave pública de B, cifra el mensaje y se lo envía a B.
5.- El mensaje enviado por A a B, es interceptado por C, y descifrado con su clave privada, Kpr_C. Una vez que ha husmeado la información, la vuelve a cifrar, esta vez, con la clave Kpu_B.
6.- El usuario o entidad B recibe el mensaje cifrado por su clave publica, Kpu_B, que posee C, y pasa a descifrarlo con su clave privada Kpr_B, sin percartarse en ningún momento que el mensaje ha sido interceptado.

Ni A, ni B, pueden percatarse de que sus mensajes están siendo interceptados por un sniffer (*husmeador, fisgón*). La única manera de evitar esto consiste en que A pueda asegurar de alguna forma que la clave pública que recibe de B es auténtica. Para ello, se puede hacer que ésta esté firmada digitalmente por un amigo común, que de este modo certifica la autenticidad de la clave. Una forma industrial de este mecanismo es lo que se conoce como certificación de clave pública, que son emitidos por unas entidades de confianza llamadas autoridades certificadoras (*CA, Certification Authorities*), comentadas ya en el apartado anterior, entre las que podría recalcarse VeriSign, y que garantizan que una determinada clave pública es realmente de quien dice ser su poseedor.

Por último, un factor fundamental de las claves es su caducidad. Todas las claves han de tener un tiempo determinado de validez, denominado criptoperiodo, pasado el cual dejan de ser útiles. Esta caducidad de las claves tiene como único fin evitar que, mediante técnicas de criptoanálisis, los fisgones tengan el suficiente tiempo e información para descifrarlas.

Práctica Nº7.- Control Remoto en modo Seguro: SSH. Sistema de Ficheros SSHFS. Control Remoto de Escritorios con Guacamole

Si en la práctica anterior comprendimos todo lo necesario acerca del uso de la criptografía en las comunicaciones informáticas, en esta práctica veremos un caso muy habitual de su utilización: **SSH** (*Secure Shell*). Mediante SSH podremos conectarnos remotamente al equipo que deseemos y enviarle ordenes o comandos en modo seguro, ya que mediante el uso de la criptografía los paquetes TCP/IP que formen parte de la comunicación viajarán de manera cifrada sin que ningún sniffer intermedio pueda adivinar ni quien esta accediendo, ni que password ha utilizado, ni que ordenes esta enviando.

En concreto, SSH hace uso de criptografía híbrida ya que hace uso de claves asimétricas durante el establecimiento de la comunicación con la finalidad de intercambiarse los equipos emisor y receptor la clave simétrica que usarán durante el resto de la comunicación. Como ya se explico en la práctica anterior, el motivo por el cual se usa una estrategia híbrida en lugar de usar claves asimétricas durante toda la comunicación es por evitar el gran consumo computacional que ello conllevaría.

El único requisito para poder llevar a cabo la práctica, es que el equipo emisor cliente disponga de una aplicación SSH cliente (*en GNU/Linux esta preinstalado, y en Windows es aconsejable usar el software putty*), y que el equipo servidor SSH remoto a controlar tenga instalada la aplicación servidora SSH (*openssh-server*).

A modo de ejemplo, la sintaxis más básica de uso de SSH podría ser la siguiente:

```
[usuario@cliente]$ ssh usuario_remoto@ip_servidor_SSH Comando
[usuario@cliente]$ ssh root@192.168.1.100 'apt-get install bind9'
```

7.1.- Configuración Básica del Servidor SSH

Para poder controlar un equipo remotamente vía SSH tan sólo necesitaremos instalar el software servidor "**openssh-server**":

```
[root@servidor]# apt-get install openssh-server
```

¡¡Aclaración!! Como es obvio, para poder acceder remotamente a nuestro servidor será necesario ponerlo al alcance del equipo o equipos clientes. En el caso de que el equipo cliente se encuentre en la misma red lógica (*o en la misma Intranet, con una correcta configuración de las reglas de enrutamiento*) tan sólo será necesario hacer referencia a la dirección IP que tenga el servidor SSH en la conexión, pero si el equipo cliente se encuentra en la Extranet (Internet) será necesario configurar un redireccionamiento de puertos en el router ADSL hacia el servidor SSH de la Intranet

y hacer referencia en el cliente a la dirección IP pública del router ADSL durante la conexión.

Aunque una vez instalado el servicio SSH éste ya esta operativo a la escucha de posibles solicitudes de conexión en modo seguro, puede ser necesaria una modificación de la configuración por defecto del servicio SSH. Para ello se editará el fichero "**/etc/ssh/sshd_config**".

¡¡Observación!! Dentro del directorio **/etc/ssh** podremos encontrar dos ficheros de configuración, con nombre muy parecido, pero con finalidad totalmente diferente que no deben confundirse:

- **/etc/ssh/ssh_config**: fichero de configuración que nos permite establecer el comportamiento de la aplicación cliente SSH.
- **/etc/ssh/sshd_config**: fichero de configuración que nos permite decidir el comportamiento del servicio SSH.

Alguna de las directivas de configuración más interesantes que podremos encontrar son:

Directiva de Configuración SSH	Explicación de Uso
Port	Nos permite indicar el puerto o puertos de escucha. Por defecto es el tcp/22, pero podemos modificarlo o añadir nuevos puertos de escucha.
ListenAddress	Nos permite configurar a través de que direcciones IP de la máquina se escuchará solicitudes de conexión al servicio SSH. Por defecto, por cualquiera.
LogLevel	Nos permite decidir la cantidad de eventos que serán auditados. Su valor puede ser "quiet, fatal, error, info, verbose, debug, debug1, debug2, y debug3", siendo su valor por defecto el nivel intermedio **Verbose**.
PermitRootLogin	Nos permite decidir si queremos que pueda establecerse una conexión remota vía SSH con la cuenta de usuario **root**. Por defecto vale "**yes**".
LoginGraceTime	Establece el tiempo del cual dispone el usuario para indicar una autenticación correcta. Si se le asigna un cero, el tiempo será indefinido.
AllowUsers **DenyUsers** **AllowGroups** **DenyGroups**	Nos permiten decidir sobre que usuarios o grupos de usuarios se podrá establecer una conexión SSH.
ClientAliveInterval **ClientAliveCountMax**	La directiva **ClientAliveInterval** le indica al servicio SSH con que frecuencia el cliente SSH tiene que dar señales de vida con la finalidad de mantener una conexión SSH activa. En caso de no recibir respuesta del cliente el servidor lo volverá a intentar durante el número de veces especificado en **ClientAliveCountMax**.
HostKey	Indica las rutas de las claves privadas RSA y DSA usadas durante el establecimiento de la comunicación.
X11Forwarding **X11UseLocalHost**	Permiten lanzar aplicaciones gráficas remota desde el cliente SSH. Por defecto esta deshabilitada esta posibilidad. Para habilitarlo deberemos

	asignarles el valor "yes". Para que surta efecto esta opción es necesario que en el fichero de configuración del cliente SSH, **/etc/ssh/ssh_config**, la directiva **ForwardX11** tenga asignado el valor "yes" (*por defecto, "no"*).
AllowTcpForwarding	Asignándole el valor "yes" nos va a permitir el redireccionamiento de puertos a través de un túnel SSH: **ssh -L**<puerto_local>:<localhost>:<puerto_remoto> <equipo_remoto> **ssh -R**<puerto_local>:<equipo_remoto>:<puerto_remoto> <localhost> Esta opción es muy interesante cuando queremos establecer una comunicación en modo seguro entre un cliente y un servidor a partir de un protocolo que es inseguro.

Una vez personalizado el servicio SSH tan sólo será necesario reiniciarlo para que surtan efecto los cambios:

```
[root@servidor]# /etc/init.d/ssh restart
```

A partir de ese momento, ya podrá enviarse desde un equipo cliente ordenes vía SSH de ejecución sobre los comandos deseados, o directamente iniciar una sesión remota. Para ello será necesario introducir la password del usuario remoto bajo el cual queremos ejecutar el comando o iniciar la sesión:

```
[usuario@cliente]$ ssh usuario_remoto@ip_servidor_SSH Comando
[usuario@cliente]$ ssh root@192.168.1.100 'apt-get install bind9'
[usuario@cliente]$ ssh arturo@192.168.1.100 'sudo apt-get upgrade'
[usuario@cliente]$ dd if=/dev/sdb | ssh arturo@192.168.1.100 'dd of=/mnt/backup-discob.iso'
[usuario@cliente]$ ssh arturo@192.168.1.100
```

Ej. Práctico 7.1.1: Ejecución de Comandos sobre un Servidor SSH Remoto

Una vez instalado el software servidor **openssh-server** en un equipo comprueba desde un equipo cliente SSH las siguientes acciones en nombre del **root** de la máquina remota:

(A) Listar los archivos y subdirectorios de configuración que existen dentro del directorio /etc del servidor.
(B) Instalar un paquete software en el equipo servidor (*p.e. apache2, bind9, proftpd, etc.*).
(C) Realizar un backup o duplicado de los datos (dd) de un directorio o partición de un disco y almacénalo en el servidor SSH dentro de un subdirectorio que deberás crear expresamente para dicho cometido "/var/backups".

Solución Ej. Pr. 7.1.1.I.- Ejecución de Comandos sobre un Servidor SSH Remoto

Suponiendo a modo de ejemplo que la dirección IP del equipo servidor SSH es 192.168.1.100, los comandos serían los siguientes:

(A) [usuario@cliente]$ **ssh root@192.168.1.100 'ls -l /etc'**
(B) [usuario@cliente]$ **ssh root@192.168.1.100 'apt-get install apache2 bind9 proftpd'**
(C) [usuario@cliente]$ **dd if=/dev/sdb1 | ssh root@192.168.1.100 'dd of=/var/backups/copia.iso'**

¡¡Importante!! Para que la ejecución de los comandos ssh lanzados desde el equipo cliente por la cuenta de usuario "**usuario**@cliente" sobre el equipo servidor ssh en nombre de la cuenta de usuario root@servidor funcionen, debemos tener en cuenta que la cuenta del **root**@servidor debe tener una contraseña habilitada, ya que hay distribuciones GNU/Linux como Ubuntu donde esto no sucede. Para ello:

[usuario@remoto]$ **sudo su**
[root@remoto]# **passwd root**

Ej. Práctico 7.1.2: Configuración del Servicio SSHd

Modifica las directivas de configuración del servicio SSH necesarias para cumplir los requisitos que se establecen en la siguiente tabla, y comprueba que surten efecto:

Usuarios Permitidos / Denegados	Puerto de Escucha	Ejecución de Aplicaciones Gráficas
AllowGroups gruposssh	**22, 22022, 22222**	**X11Forwarding yes** **X11UseLocalHost yes**
PermitRootLogin No		

Solución Ej. Pr. 7.1.2.I.- Configuración del Servicio SSH

Para que surta efecto la configuración solicitada llevaremos a cabo los siguientes pasos:

(1) Comenzaremos creando en el equipo servidor el grupo de usuarios con permiso para poder acceder remotamente a través del servicio SSH (*groupadd*, *useradd*, *usermod*, *etc.*):

[root@servidorSSH]# **groupadd gruposssh**
[root@servidorSSH]# **useradd -m -d /home/ususssh1 -g gruposssh -s /bin/bash ususssh1**
[root@servidorSSH]# **useradd -m -d /home/ususssh2 -g gruposssh -s /bin/bash ususssh2**
[root@servidorSSH]# **useradd -m -d /home/ususssh3 -g gruposssh -s /bin/bash ususssh3**
[root@servidorSSH]# **...**
[root@servidorSSH]# **passwd ususssh1**
[root@servidorSSH]# **passwd ususssh2**
[root@servidorSSH]# **passwd ususssh3**
[root@servidorSSH]# **...**

(2) En el servidor SSH editaremos el archivo **sshd_config** y añadiremos o modificaremos las siguientes directivas de configuración:

[root@servidorSSH]# **nano /etc/ssh/sshd_config**
Port 22
Port 22022

```
Port 22222
AllowGroups gruposssh
PermitRootLogin No
X11Forwarding yes
X11UseLocalHost yes
...
```

(3) En el equipo cliente se modificará la directiva **ForwardX11** en su archivo de configuración con la finalidad de habilitar la ejecución de aplicaciones gráficas (X11) en el lado del servidor vía SSH. Las ventanas de la aplicación que se ejecute en el servidor se redireccionarán hacia el equipo cliente, cediéndole su control.

```
[root@clienteSSH]# nano /etc/ssh/ssh_config
…
ForwardX11 yes
...
```

(4) Por último, comprobaremos desde el equipo cliente que los únicos usuarios que se pueden autenticar vía SSH son los pertenecientes al grupo gruposssh, y que es posible ejecutar aplicaciones gráficas (*se supone que el equipo servidor SSH tiene instaladas aplicaciones gráficas que corren bajo una interfaz gráfica como gnome, kde, etc.*).

```
[usuario@clienteSSH]# ssh root@192.168.1.100
root@192.168.1.100's password:
Permission denied, please try again.
[usuario@clienteSSH]# ssh arturo@192.168.1.100
arturo@192.168.1.100's password:
Permission denied, please try again.
[usuario@clienteSSH]# ssh [-X] ususshl@192.168.1.100
ususshl@192.168.1.100's password:
[ususshl@servidorSSH]# gedit index.php
```

7.2.- Ejecución de Comandos Remotamente de Manera Desatendida

Como se habrá advertido a través de los ejercicio prácticos anteriores, cuando un usuario lanza un comando remotamente sobre el servidor SSH, éste con la finalidad de corroborar que quien lo ordena es quien realmente dice ser que es, nos solicita la contraseña del usuario bajo el cual se ejecuta el comando. Esta característica que a simple vista es idónea para evitar que alguien pueda ejecutar comandos sin permiso, en ocasiones puede ser un gran inconveniente. Por ejemplo, si quisiéramos programar un script que fuera ejecutado de manera automática mediante el servicio Cron o vía Web (*PHP/exec()*), y éste contuviese algún comando a ejecutar remotamente vía SSH, se quedaría "colgado" a la espera de alguien introdujera la password del usuario remoto bajo el cual se trata de ejecutar dicho comando.

Para posibilitar la ejecución de comandos de manera desatendida, sin necesidad de tener que

estar presentes para introducir la correspondiente password solicitada por el servicio SSH, existe la posibilidad de configurar una conexión como confiable. En concreto, dicha configuración se basa en el uso de claves asimétricas y la firma digital tal como se describe a continuación:

(1) El usuario que quiere establecer una conexión confiable con el servidor remoto generará un para de claves asimétricas (*no es necesario ni introducir frase de paso, ni modificar la ubicación de las claves; puede pulsarse al INTRO sin más*):

```
[usuario@cliente]$ ssh-keygen
```

(2) Hacer llegar la parte pública de las claves asimétricas creadas en el paso anterior a la cuenta de usuario del equipo servidor SSH en nombre de la cual se ejecutarán los comandos remotamente. Dicha clave pública será utilizada por el servidor SSH para comprobar la firma digital del usuario que lanza el comando a ejecutar remotamente, y de esta forma comprobar la autenticación del usuario emisor, tal como se explico en la práctica o capítulo anterior referente a la criptografía. Para la distribución de la clave pública el cliente SSH ya dispone de una aplicación que se encarga de ello, "**ssh-copy-id -i**". Al ejecutar dicho comando se nos solicitará la contraseña del usuario remoto, ya que modificaremos el contenido de su archivo "**authorized_keys**" donde se almacenarán las claves públicas de los usuarios de confianza:

```
[usuario@cliente]$ ssh-copy-id -i ~/.ssh/id_rsa.pub  arturo@192.168.1.100
[usuario@cliente]$ ssh-copy-id -i ~/.ssh/id_rsa.pub  root@192.168.1.100
```

¡¡Aclaración!! En el caso de que no exista el subdirectorio ".ssh" dentro del HOME del usuario de la máquina remota "arturo" o "root", y dentro de él, el archivo "**authorized_keys**", el comando "**ssh-copy-id -i**" los creará automáticamente. De esta manera estamos garantizando que el usuario que ha exportado su clave pública pase a ser un usuario de confianza por el usuario del equipo servidor SSH que la ha importado. Es decir, a partir de ese momento, cuando el usuario remoto trate de conectarse vía SSH, en la solicitud de conexión adjunto viajará un extracto firmado por él. El servidor SSH comprobará a través de la clave pública la firma, corroborará que quien trata de acceder es quien dice ser que es, y le dejará entrar sin necesidad de introducir login y password.
Aclarar igualmente que el carácter "~" en GNU/Linux es especial, y hace referencia al directorio HOME del usuario que lo utiliza.

Ej. Práctico 7.2.1: Ejecución de Comandos Remotamente de manera Desatendida

Indica todos los pasos que serían necesarios para que un usuario llamado "**usulocal**" desde un equipo cliente pueda **parar e iniciar el servicio Web apache2** de un **equipo servidor remoto** de manera **desatendida**, en nombre de un usuario remoto "**usuremoto**" (*debe tener privilegios de **root** para gestionar el servicio Apache*), tal como se especifica en la siguiente tabla. Advierte que **(1)** deberás conceder los privilegios necesarios mediante **sudo**, **(2)** deberás asegurarte de que **usulocal** es un usuario de confianza que puede mandar la ejecución de dichos comandos de manera desatendida (*no se le debe solicitar password*), y **(3)** deberás configurar el servicio **Crontab** para que se ejecuten en el momento deseado.

Comandos a Ejecutar	Hora y Dias	IP Servidor *(p.e.)*	Usuario Local	Usuario Remoto
/etc/init.d/apache2 stop	**23:35h L/M/X/J/V**	**192.168.1.100**	**usulocal**	**usuremoto**
/etc/init.d/apache2 start	**6:00h L/M/X/J/V**			

Solución Ej. Pr. 7.2.1.I.- Cómo Ejecutar Comandos Desatendidos Remotamente

Para dar solución al problema que se plantea seguiremos los siguientes pasos:

(1) Crearemos las cuentas de usuario **usulocal** y **usuremoto** en los equipos cliente y servidor respectivamente.

```
[root@cliente]# useradd -m -d /home/usulocal -s /bin/bash usulocal
[root@cliente]# passwd usulocal
[root@servidor]# useradd -m -d /home/usuremoto -s /bin/bash usuremoto
[root@servidor]# passwd usuremoto
```

(2) Configuraremos el **sudo** en el equipo servidor con la finalidad de que la cuenta de usuario **usuremoto** pueda parar e iniciar el servicio Apache. De esta forma, desde el equipo cliente podremos ejecutar los comandos deseados en nombre de ese usuario.

```
[root@servidor]# visudo
# Añadimos la siguiente línea a la configuración del sudo:
usuremoto ALL = (root) NOPASSWD: /etc/init.d/apache2 stop, /etc/init.d/apache2 start
```

¡¡Observación!! La directiva **NOPASSWD:** de configuración del **sudo** es fundamental ya que la ejecución de los comandos anteriores debe ser de manera desatendida. Es decir, al incluirla evitaremos que **sudo** le pida la contraseña a **usuremoto** al ejecutar los comandos indicados.

(3) Configuramos la cuenta de usuario **usulocal** como usuario de confianza en el equipo servidor, con la finalidad de que pueda ejecutar los comandos deseados remotamente vía SSH sin necesidad de autenticación (*de manera desatendida*). Para ello **usulocal** deberá generar una pareja de claves

asimétricas y enviar la parte pública a **usuremoto**. De esta forma **usuremoto** podrá comprobar la firma digital (*autenticación*) de **usulocal** que es enviada junto al comando a ejecutar vía SSH, y así advertir que es quien dice ser que es.

```
[usulocal@cliente]$ ssh-keygen
[usulocal@cliente]$ ssh-copy-id -i ~/.ssh/id_rsa.pub usuremoto@192.168.1.100
```

Ahora el usuario **usulocal** ya es un usuario de confianza para **usuremoto** de tal forma que puede ejecutar comandos en su nombre en la máquina remota.

(4) Por último, el usuario **usulocal** configurará su servicio **Crontab** con la finalidad de programar la ejecución de los comandos indicados en el enunciado a las horas y días deseados.

```
[usulocal@cliente]$ crontab -e
# Añadimos la siguiente línea a la configuración del Crontab del usuario usulocal
# Mínuto Hora Día_del_Mes Mes Día_de_la_Semana Comando_a_ejecutar
35 23 * * 1-5 ssh usuremoto@192.168.1.100 'sudo /etc/init.d/apache2 stop'
00 06 * * 1-5 ssh usuremoto@192.168.1.100 'sudo /etc/init.d/apache2 start'
```

¡¡Observación!! Tal como se puede observar a través del ejercicio anterior, para poder ejecutar comandos remotamente de manera desatendida sin necesidad de tener que introducir ninguna password han sido necesarios dos configuraciones:

- Crear una pareja de claves asimétricas y enviar la clave pública al servidor. Tal como ya se ha explicado anteriormente, cuando iniciamos una conexión SSH/SCP/SFTP/... sin que el usuario lo sepa viaja con la solicitud de conexión un documento firmado con la clave privada del usuario. En el caso de que el usuario del lado del servidor disponga de su clave pública podrá comprobar la firma digital y de esta forma asegurarse de que quien quiere conectarse es quien dice ser que es, autenticándolo y de esta forma evitar que tenga que introducir una password.
- Además, en el caso de que en nombre del usuario remoto queramos ejecutar un comando con privilegios de root, deberemos configurar el sudo y añadir la directiva **NOPASSWD:** para evitar el tener que introducir una password.

Ej. Práctico 7.2.2: Ejecución Desatendida de un Script con Comandos SSH

En caso de que no exista, crea un nuevo usuario en la máquina cliente SSH llamado **ususshl**@cliente y configura su servicio **Crontab** para que ejecute un pequeño script llamado **script.sh** cada minuto, el cual se encargará de crear un directorio en el equipo servidor SSH con la fecha y hora del sistema en /var/remoto/*fecha-hora* en nombre del usuario **root**@servidor (*p.e. 192.168.1.100*).

Usuario Equipo Cliente	Usuario Equipo Servidor	IP Servidor (*p.e.*)	Crontab	Script a Ejecutar
ususshl	**root**	192.168.1.100	**Cada mínuto**	**script.sh**

***Solución Ej. Pr. 7.2.2.I.**- Ejecución Desatendida de un Script con Comandos SSH*

Para realizar el ejercicio práctico propuesto seguiremos los siguientes pasos:

(1) Comenzaremos creando la cuenta de usuario ususssh1 en el equipo cliente SSH, lo configuraremos con usuario de confianza en el servidor SSH y configuraremos el servicio Cron para que ejecute el script cada minuto:

```
[root@cliente]# useradd -m -d /home/ususssh1 -s /bin/bash ususssh1
[root@cliente]# passwd ususssh1
[root@cliente]# su ususssh1
[ususssh1@cliente]$ cd
[ususssh1@cliente]$ ssh-keygen
[ususssh1@cliente]$ ssh-copy-id -i ~/.ssh/id_rsa.pub  root@192.168.1.100
[ususssh1@cliente]$ nano script.sh
#!/bin/bash
FECHAYHORA=`ssh root@192.168.1.100 date +%y-%m-%d-%H:%M`
ssh root@192.168.1.100 mkdir -p /var/remoto/$FECHAYHORA
```

```
[ususssh1@cliente]$ chmod +x script.sh
[ususssh1@cliente]$ crontab -e
# Añadimos la siguiente línea a la configuración del Crontab del usuario usulocal
# Mínuto Hora Día_del_Mes Mes Día_de_la_Semana Comando_a_ejecutar
*/1 * * * * /home/ususssh1/script.sh
```

(2) Comprobaremos que la creación de directorios en el servidor vía SSH de manera desatendida funciona perfectamente listando el contenido del directorio /var/remoto:

```
[ususssh1@cliente]$ ssh  root@192.168.1.100 ls -l /var/remoto
```

7.3.- Transferencia de Archivos en Modo Seguro: SCP

Además de ejecutar comandos en nombre de un usuario remoto en un equipo remoto, el servicio SSH nos permite transferir archivos entre el cliente y el servidor. Para ello se hará uso del comando "**scp**" (*Secure CoPy*).

Por ejemplo, si quisiéramos transferir un archivo desde el cliente a un servidor remoto ejecutaríamos el siguiente comando (*scp origen destino*):

```
[usuario@cliente]$ scp ~/Documentos/fich1.pdf arturo@82.23.103.132:/home/arturo/PDFs
```

Y en el caso de querer realizar la transferencia al revés (*coger un archivo que se localiza en el servidor y traérnoslo a nuestro equipo cliente*):

```
[usuario@cliente]$ scp arturo@82.23.103.132:/home/arturo/PDFs/doc1.pdf /tmp/doc1.pdf
```

7.4.- Unidades de Almacenamiento en Red Seguras: SSHFS

Con la finalidad de complementar al comando scp, el servicio SSH nos ofrece la opción de crear una unidad de red en el equipo cliente SSH asociada a un directorio del servidor SSH, de tal forma que el contenido de dicha unidad de red será el contenga el directorio del servidor.

Para ello, tan sólo será necesario instalar en el equipo cliente el paquete software sshfs (*File System SSH*) y configurar la conexión de manera manual o automática a través del fichero /etc/fstab.

Ej. Práctico 7.4.1: Unidad de Red SSHFS

A modo de ejemplo, a continuación se mostrará como crear una unidad de red SSHFS con las características que se indican en la siguiente tabla. La conexión con la unidad de red se hará manual. Se asumen que los directorios y usuarios ya existen en las máquinas:

IP Servidor / Directorio Remoto	Directorio Local para el Montaje	Usuario Cliente	Usuario Remoto
192.168.1.100 /home/usuremoto1/Documentos	**/home/usucliente1/unidadsshfs**	**usucliente1**	**usuremoto1**

Solución Ej. Pr. 7.4.1.I.- Como Configurar una Unidad de Red SSHFS

A continuación se detallan los comandos que ejecutaríamos en el equipo cliente para poder configurar la unidad de red SSHFS, sin necesidad de password en el momento del establecimiento de la conexión (*la cuenta de usuario usucliente1 deberá ser una cuenta de usuario de confianza para el usuario remoto usuremoto1*):

```
[root@cliente]# apt-get install sshfs
[root@cliente]# su usucliente1
[usucliente1@cliente]$ ssh-keygen
[usucliente1@cliente]$ ssh-copy-id -i ~/.ssh/id_rsa.pub  usuremoto1@192.168.1.100
[usucliente1@cliente]$ mkdir /home/usucliente1/unidadsshfs
[usucliente1@cliente]$ sshfs usuremoto1@192.168.1.100:/home/usuremoto1/Documentos \
                       /home/usucliente1/unidadsshfs
```

Tras ejecutar el comando sshfs el usuario usucliente1@cliente ya dispondrá de una unidad de red, cuyo contenido será el que contenga el servidor SSH en el directorio al que hace referencia "/home/usuremoto1/Documentos":

```
[usucliente1@cliente]$ df -h
[usucliente1@cliente]$ ls -l /home/usucliente1/unidadsshfs
```

En el caso de querer desconectarse de la unidad de red ejecutaríamos el siguiente comando:

```
[usucliente1@cliente]$ fusermount -u ~/unidadsshfs
```

Ej. Práctico 7.4.2: Conexión SSHFS de manera Automática

Siguiendo con el ejercicio anterior, configura el equipo cliente para que el montaje sobre la unidad de red SSHFS se haga de manera automática.

Solución Ej. Pr. 7.4.2.I.- Cómo Configurar la Conexión SSHFS Automáticamente

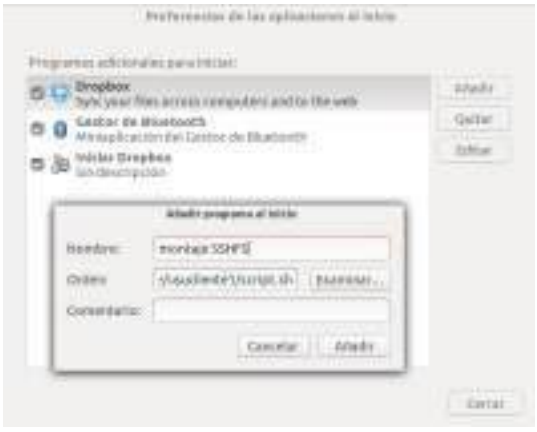
Para realizar el montaje de manera automática podríamos hacer uso del archivo de automontaje del sistema "**/etc/fstab**", pero al ser el directorio de montaje de un subdirectorio del HOME del usuario, es más sencillo crear un pequeño script encargado de hacer la conexión y añadirlo en "**Aplicaciones al Inicio**" en el caso de que el equipo cliente disponga de una interfaz gráfica, o en el "**~/.bashrc**" en el caso de que la interfaz del equipo cliente sea en modo consola:

(1) Editaremos el script que se encargará de la conexión y le daremos permiso de ejecución:

```
[usucliente1@cliente]$ nano /home/usucliente1/script.sh
#!/bin/bash
sshfs usuremoto1@192.168.1.100:/home/usuremoto1/Documentos  \
                /home/usucliente1/unidadsshfs
```

```
[usucliente1@cliente]$ chmod +x /home/usucliente1/script.sh
```

(2) En el caso de que el usuario en el equipo cliente inicie sesión de manera gráfica (*GUI*), editaremos las "**Aplicaciones al Inicio**":



(3) En el caso de que el usuario en el equipo cliente inicie sesión en modo consola editaremos el "**~/.bashrc**":

```
[usucliente1@cliente]$ nano ~/.bashrc
```

```
…
sshfs usuremoto1@192.168.1.100:/home/usuremoto1/Documentos  \
                    /home/usucliente1/unidadsshfs
```

En el caso de querer desconectarse de la unidad de red ejecutaríamos el siguiente comando:

```
[usucliente1@cliente]$ fusermount -u ~/unidadsshfs
```

Ej. Práctico 7.4.3: Conexión SSHFS desde el /etc/fstab

En el caso de querer montar el sistema de archivos SSHFS a través del fichero del sistema de automontaje /etc/fstab, deberemos configurar previamente el servicio fuse, encargado de su gestión. La forma de hacerlo se mostrará en la siguiente solución.

Solución Ej. Pr. 7.4.3.I.- Cómo Configurar una Conexión SSHFS desde el /etc/fstab

Para ello seguiremos los siguientes pasos:

(1) Crearemos el directorio donde se va a llevar a cabo el montaje dentro del equipo cliente:

```
[root@cliente]# mkdir /mnt/montaje
```

(2) Editaremos el /etc/fstab:

```
[root@cliente]# nano /etc/fstab
…
# Añadiremos la siguiente línea:
sshfs#usuremoto1@192.168.1.100:/home/usuremoto1/Documentos /mnt/montaje fuse \
noauto,exec,user,reconnect,allow_other 0 0
```

¡¡Advertencia!! Tal como se puede advertir a través de la línea anterior, el sistema de archivos para el montaje no es ni ext3, ni ext4, ni vfat, ni … sino **fuse**. Éste es un servicio que se encargará del montaje de la unidad de red y de su correcto funcionamiento, pero requerirá de una pequeña configuración previa, tal como se mostrará a continuación.

(3) Editaremos el fichero "**/etc/fuse.conf**", y quitaremos la "**#**" que precede a la directiva "**user_allow_other**" para que sea tenida en cuenta. De esta forma cualquier usuario sin necesidad de ser el root podrá montar unidades de red del fichero "**/etc/fstab**" con sistema de archivos "**fuse**".

(4) Modificaremos a los usuarios que deseemos que puedan llevar a cabo el montaje agrupándolos dentro del grupo de usuarios del sistema **fuse**:

```
[root@cliente]# usermod -G fuse  usucliente1
```

(5) Por último, crearemos un script que se limitará a establecer el montaje, lanzándolo desde "**Aplicaciones al Inicio**" o "**~/.bashrc**", tal como se ha mostrado en el anterior ejercicio práctico.

Por ejemplo, en el caso de no disponer de interfaz gráfica:

```
[usucliente1@cliente]$ nano ~/.bashrc
…
mount /mnt/montaje
```

7.5.- Proxy SOCKS mediante SSH

Si en el capítulo asociado a las prácticas de implementación de un Proxy Cache HTTP mediante Squid vimos algunas de las opciones que éste nos ofrece a la hora de gestionar el tráfico Web, en esta parte práctica aprenderemos a implementar otro tipo de Proxy que puede resultar muy interesante en determinados entornos de red denominado Proxy SOCKS.

Cuando hablamos de Proxy "a secas", lo normal es que estemos haciendo referencia a un Proxy Cache HTTP o Proxy Web, pero tenemos que tener en cuenta que el concepto de Proxy o intermediario es utilizado para muchos otros ámbitos (FTP, DNS, etc.), recibiendo en esos casos otro tipo de adjetivos: Proxy NAT, Proxy Inverso, Proxy SOCKS, etc. En concreto, el Proxy SOCKS (*SOCKetS*) esta pensado no sólo para gestionar el tráfico Web, sino para hacer de intermediario entre equipos clientes y servidores permitiéndonos gestionar cualquier tipo de conexión que sea soportada por el protocolo SOCKS (*RFC 3089*). Es decir, el equipo cliente tiene que ser un cliente SOCKS que implemente este tipo de protocolo específico.

Una forma sencilla de comprender como funciona un Proxy SOCKS es utilizar **ssh** con la opción "**-D número_puerto IP_servidor_SSH**", la cual nos permite hacer una redirección de puertos dinámica, convirtiendo nuestro equipo y el túnel SSH creado con el servidor, en un Proxy SOCKS redireccionando todo el tráfico de la aplicación que configuremos por el canal seguro hacía el servidor SSH que hayamos especificado, redirigiendo éste a su vez el tráfico hacia el equipo que indiquemos.

Un caso típico de uso de un Proxy SOCKS podría ser el siguiente: nos encontramos navegando por Internet en una red insegura, o con dudas relativas a que nuestras comunicaciones puedan ser husmeadas por algún "sniffer", y queremos evitarlo creando un túnel SSH cifrado seguro por el cual viajará todo el tráfico generado por nuestro navegador Web. A través de esta configuración todo el tráfico que generemos será redireccionado hacía el servidor SSH que especifiquemos y este hará de intermediario (*Proxy*) entre yo y los servidores a los que me conecte. Por ejemplo, supongamos que tenemos asociado un nombre de dominio público gratuito a la dirección IP pública del router de nuestra casa (*p.e. micasa.no-ip.org*), y que éste tiene redireccionado el puerto 22 hacía un servidor SSH que tenemos internamente (*p.e. una Raspberry Pi*). Al establecer el tunel SSH con el servidor haciendo uso de la opción **-D**, hacemos que el servidor de nuestra casa haga de Proxy entre nosotros y los servidores a los que nos conectemos durante la navegación Web.

Ej. Práctico 7.5.1: Implementación de un Proxy SOCKS mediante SSH

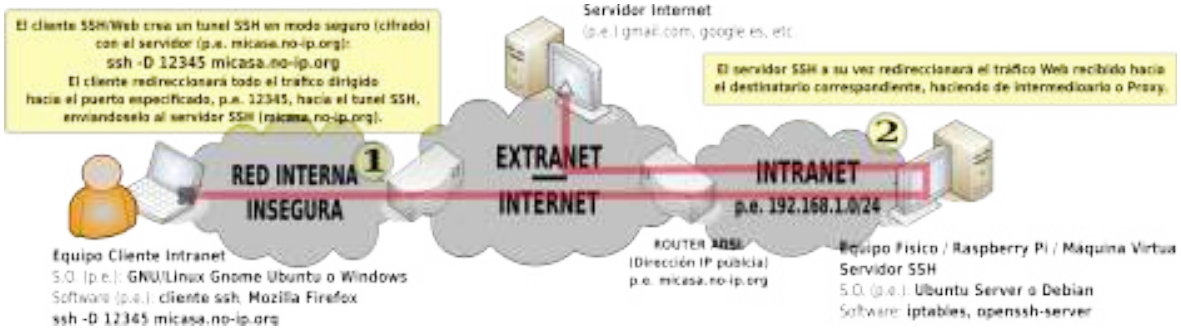
A modo de ejemplo, a continuación se muestra como configurar un equipo cliente para hacer uso de un **Proxy SOCKS** local implementado mediante **SSH** mientras esta navegando en una red supuestamente insegura, de tal forma que todo el tráfico generado por el cliente (*p.e. Mozilla Firefox integra un cliente SOCKS que soporta el protocolo SOCKS*) y dirigido hacía el puerto que especifiquemos (*p.e. 12345*), será encapsulado y redireccionado hacía el servidor SSH, reenviándolo éste a su vez al servidor Web correspondiente. Advertir, que en tal situación, el ancho de banda de la conexión vendrá limitado por la velocidad de upload de la línea ADSL contratada en

el lado del servidor SSH.

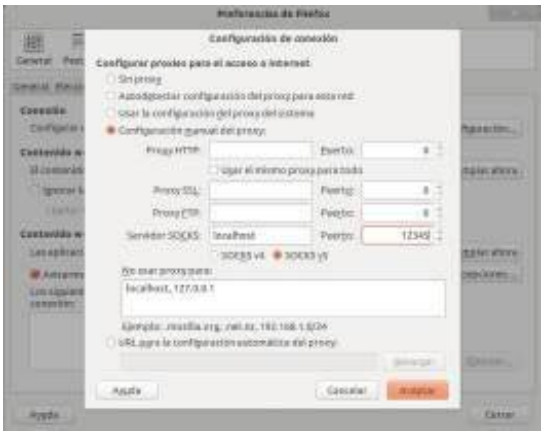
Solución Ej. Pr. 7.5.1.I.- Cómo Implementar un Proxy SOCKS mediante SSH

Para ello, tan sólo necesitaremos crear el tunel SSH (*en Windows haríamos uso de putty*), y posteriormente configurar las opciones de nuestro navegador (*se supone que disponemos de un servidor SSH dentro de una Intranet cualquiera*):

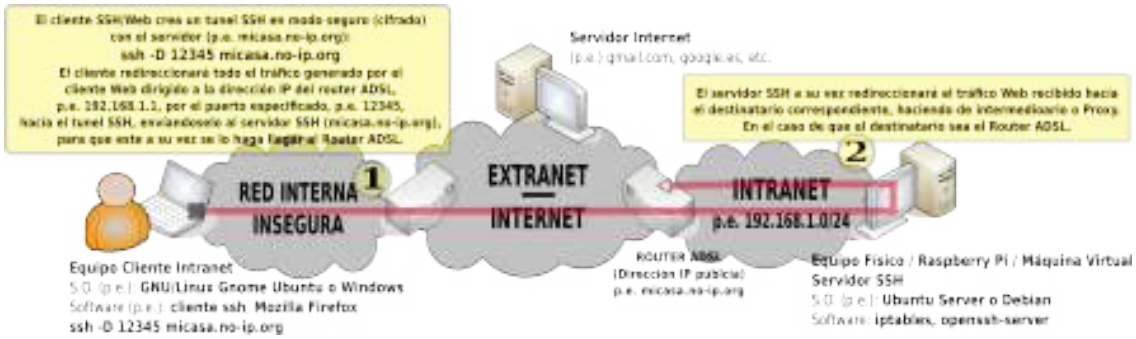
```
[usuario-cliente@linux]$ ssh -D 12345 micasa.no-ip.org
```



En el caso de que hagamos uso de Mozilla Firefox (*Editar → Preferencias → Avanzado → Red → Configuración*), aunque podría cualquier otro tipo de navegador:



Otra opción de uso muy interesante de todo lo anterior, es usar el túnel SSH creado a modo de "VPN", ya que a través del Proxy SOCKS podemos alcanzar cualquier equipo de la Intranet donde se localiza el servidor SSH. Por ejemplo, podríamos configurar el router ADSL haciendo uso de la dirección IP privada de la Intranet que tenga asignada (*p.e. 192.168.1.1*):

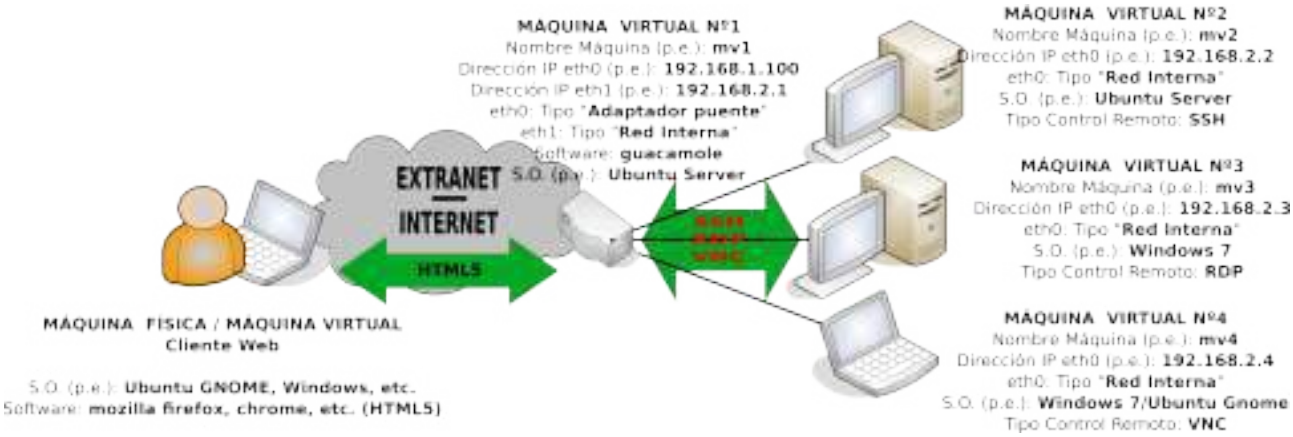


¡¡Observación!! Para que el servidor SSH redireccione las peticiones del cliente SOCKS hacía los destinatarios deseados, deberemos activar el ip_forward y el enmascaramiento de los paquetes (*repasar capítulo del firewall iptables*):

```
[root@servidorssh]# echo "1" > /proc/sys/net/ipv4/ip_forward
[root@servidorssh]# iptables -t nat -A POSTROUTING -j MASQUERADE
```

7.6.- Control Remoto de Múltiples Escritorios desde un Navegador Web: Guacamole

Para terminar con la presente práctica se mostrará el paquete software **guacamole** que nos va permitir conectarnos remotamente a diferentes equipos e iniciar sesión de escritorio en ellos (*GNU/Linux o Windows*) a través de un simple navegador Web (*Intérprete de HTML5*). En concreto, tal como se mostrará, guacamole simplemente hará de intermediario entre el cliente y los equipos a los que se quiere conectar remotamente, suministrándonos un streaming en formato HTML5 del escritorio remoto.



Tal como puede apreciarse en la anterior figura, y tal como advertiremos a través de los ejercicios prácticos propuestos, guacamole soporta los protocolos de gestión remota más afamados: SSH (*Secure Shell*), VNC (*Virtual Network Computing*) y RDP (*Remote Desktop Protocol*).

Para ello, previamente deberemos instalarnos la última versión estable desde los repositorios del grupo de trabajo de **guacamole**:

```
[root@servidor]# apt-get install software-properties-common python-software-properties
[root@servidor]# add-apt-repository ppa:guacamole/stable
[root@servidor]# apt-get update
[root@servidor]# apt-get install guacamole-tomcat libguac-client-rdp0 libguac-client-ssh0 \
                        libguac-client-vnc0
```

¡¡Aclaración!! Según la distribución de Ubuntu que usemos tal vez no dispongamos de la aplicaión preinstalada **add-apt-repository** necesaria para gestionar repositorios PPA (*Personal Package Archive*). Para poder disponer de ella se necesita instalar los paquetes software **software-properties-common python-software-properties**.

¡¡Advertencia!! En el caso de que estemos usando un sistema operativo Debian, Raspbian (*Raspberry Pi*), u otra distribución que no sea Ubuntu, deberemos descargarnos de la Web oficial de **guacamole** (*http://guac-dev.org/*) el código fuente si queremos trabajar con la última versión, o

directamente instalarlo desde los repositorios disponibles aunque teniendo en cuenta que seguramente no será su última versión disponible.
También se ha comprobado en distribuciones diferentes a Ubuntu que una vez instalado **guacamole** desde sus repositorios surgen problemas con el servicio **tomcat** (*servidor Web con soporte para Java*) que le da soporte vía Web (*tomcat nos permite comunicarnos con guacamole a través de un simple navegador Web*). En concreto, puede dar problemas al reiniciar el servicio **tomcat** (*/etc/init.d/tomcat6 restart*) al no detectar la variable del sistema **JAVA_HOME**. Para solucionar el problema será necesario dar el valor correcto a dicha variable del sistema :

```
[root@servidor]# ls /usr/lib/jvm/
[root@servidor]# export JAVA_HOME=/usr/lib/jvm/subdirectorio-jdk-java
[root@servidor]# /etc/init.d/tomcat6 restart
[root@servidor]# /etc/init.d/guacd restart
```

Tras advertir que lo anterior funciona sería conveniente declarar la variable **JAVA_HOME** en el archivo de configuración del sistema "**/etc/default/tomcat6**" para que sea tenida en cuenta al iniciarse la máquina (*hay que descomentar la línea que hace referencia a JAVA_HOME y asignarle el valor correcto*):

```
[root@servidor]# nano /etc/default/tomcat6
JAVA_HOME=/usr/lib/jvm/subdirectorio-jdk-java
```

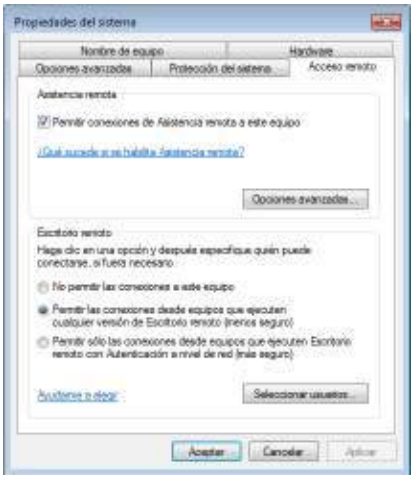
Ej. Práctico 7.6.1: Conexión Remota al Escritorio Windows mediante Guacamole

Configura un equipo Windows (*por ejemplo, un w7*) y guacamole para poder acceder al escritorio Windows remotamente desde el navegador Web de un equipo cliente (*máquina física*).

Solución Ej. Pr. 7.6.1.I.- Conexiones RDP desde Guacamole: Escritorio Windows

Para ello configuraremos por separado Windows y el Ubuntu Server bajo el cual se ha instalado **guacamole**:

(1) Configuramos nuestro equipo Windows para que acepte conexiones al servicio de Escritorio Remoto, e indicaremos las cuentas de usuario del equipo Windows que queremos que puedan ser usadas para la gestión remota del escritorio (*por cuestiones de seguridad de Windows las cuentas de usuario con las que queramos iniciar sesión remotamente deberán tener una contraseña asignada, ya que en caso de no tener password no dejará establecer la conexión*). Para ello debemos acceder a la ventana de configuración "Acceso Remoto" pinchando con el botón derecho del ratón sobre "Equipo" y eligiendo "Propiedades".



¡¡Importante!! Una vez habilitado el acceso al Escritorio Remoto en Windows (*Windows XP, Windows 7, etc*) debes cerrar la sesión con la finalidad de evitar conflictos con la sesión que abriremos a continuación remotamente. Es decir, al no ser Windows un sistema operativo multiusuario (no pueden estar varios usuarios trabajando simultáneamente con la sesión abierta) nos obliga a dejar cerrada la sesión para que el usuario que trate de acceder remotamente pueda hacerlo.

(2) Configuraremos guacamole en el equipo Ubuntu Server. Para ello tan sólo habrá que definir una entrada en el archivo "user-mapping.xml" donde indicaremos los siguientes datos:

- Nombre y contraseña de un usuario con acceso a guacamole. Es decir, cuando nos conectemos desde un navegador Web al servicio guacamole nos aparecerá un formulario de acceso compuesto por un login y una password que nos dará paso al posible control remoto del equipo Windows. El username y password indicados nos validarán en dicho formulario.
- Nombre de la conexión. Le daremos un nombre cualquiera que nos permitirá identificar la conexión con nuestro equipo Windows.
- Indicaremos como protocolo rdp, el puerto de conexión 3389 y la dirección IP del equipo Windows.

```
[root@servidor]# nano /etc/guacamole/user-mapping.xml
<user-mapping>
        <authorize username="arturo" password="1">
        <connection name="w7">
                <protocol>rdp</protocol>
                <param name="hostname">ip_windows</param>
                <param name="port">3389</param>
        </connection>
        </authorize>
</user-mapping>
```

(3) Reiniciaremos el servicio guacamole "**guacd**" para que surtan efectos los cambios realizados en el archivo de configuración **user-mapping.xml** (*aunque no es necesario puede reiniciarse el servicio tomcat encargado de dar el servicio Web/Java de acceso a guacamole*):

```
[root@servidor]# /etc/init.d/tomcat6 restart
[root@servidor]# /etc/init.d/guacd restart
```

(4) Por último, abriremos un navegador en un equipo cliente y nos conectaremos al servicio guacamole haciendo referencia a la dirección IP del equipo servidor Ubuntu Server donde esta instalado guacamole y su puerto de servicio "**8080**", seguido del alias "guacamole": **http://***ip_server***:8080/guacamole**. Desde allí podremos seleccionar sobre la conexión configurada y al poco tiempo veremos el escritorio del equipo Windows (*debe tener la sesión de usuario cerrada*).

¡¡Observación!! Como puede observarse a través de la realización del ejercicio práctico anterior, para acceder desde el navegador Web de un equipo cliente al servicio guacamole, hacemos uso del servicio Web **tomcat** (*soporte para Java*) el cual escucha por defecto por el puerto 8080. En el caso que nos interese modificar el puerto de escucha tan sólo será necesario modificar el valor de dicho puerto en el archivo "/etc/tomcat6/**server.xml**" (*Connector port="8080"*).

Ej. Práctico 7.6.2: Conexión Remota al Escritorio Linux mediante Guacamole

Configura un equipo GNU/Linux (*por ejemplo, un Ubuntu Gnome*) y guacamole para poder acceder a su escritorio remotamente desde el navegador Web de un equipo cliente (*máquina física*). Para ello configuraremos una nueva conexión en guacamole de tipo VNC. Esto significa, que aunque en este ejercicio se hace sobre un GNU/Linux, podría instalarse VNC en un Windows y acceder a él en lugar de vía RDP, vía VNC.

Solución Ej. Pr. 7.6.2.I.- Conexiones VNC desde Guacamole: Escritorio Linux

Al igual que en el ejercicio práctico anterior, para ello configuraremos por separado el equipo GNU/Linux y por otro el Ubuntu Server bajo el cual se ha instalado **guacamole**:

(1) Configuramos nuestro equipo GNU/Linux para que acepte conexiones al servicio de Escritorio Remoto. Para ello, por ejemplo si estamos en Ubuntu Gnome (*p.e. 13.04 o 13.10*) disponemos de la herramienta de configuración "**vino-preferences**" (*preferencias de compartición de Escritorio*) que nos permitirá configurarlo:

(2) Configuraremos guacamole en el equipo Ubuntu Server editando el archivo "user-mapping.xml" tal como se ha hecho en el ejercicio práctico anterior:

```
[root@servidor]# nano /etc/guacamole/user-mapping.xml
<user-mapping>
        <authorize username="arturo" password="1">
        <connection name="w7">
                <protocol>rdp</protocol>
                <param name="hostname">ip_windows</param>
                <param name="port">3389</param>
        </connection>
        <connection name="Ubuntu Remote Desktop">
                <protocol>vnc</protocol>
                <param name="hostname">ip_linux</param>
                <param name="port">5900</param>
                <param name="password">passvnc</param>
        </connection>
        </authorize>
</user-mapping>
```

(3) Reiniciaremos el servicio guacamole "**guacd**" para que surtan efectos los cambios realizados en el archivo de configuración user-mapping.xml:

```
[root@servidor]# /etc/init.d/guacd restart
```

(4) Por último, abriremos el navegador y accederemos "guacamole" y a la gestión de las conexiones configuradas: **http://***ip_server***:8080/guacamole**.

Ej. Práctico 7.6.3: Conexión Remota a Ubuntu Server mediante Guacamole

Configura un equipo GNU/Linux y guacamole para poder acceder a su terminal SSH remotamente desde el navegador Web de un equipo cliente (*máquina física*).

Solución Ej. Pr. 7.6.3.I.- Conexiones SSH desde Guacamole

Al igual que en el ejercicio práctico anterior, para ello configuraremos por separado el equipo GNU/Linux y por otro el Ubuntu Server bajo el cual se ha instalado **guacamole**:

(1) Para que nuestro equipo GNU/Linux acepte conexiones SSH deberemos asegurarnos de que tiene instalado el **openssh-server**:

```
[root@servidor]# apt-get install openssh-server
```

(2) Configuraremos guacamole en el equipo Ubuntu Server editando el archivo "user-mapping.xml" tal como se ha hecho en el ejercicio práctico anterior:

```
[root@servidor]# nano /etc/guacamole/user-mapping.xml
<user-mapping>
        <authorize username="arturo" password="1">
        <connection name="w7">
                <protocol>rdp</protocol>
                <param name="hostname">ip_windows</param>
                <param name="port">3389</param>
        </connection>
        <connection name="Ubuntu Remote Desktop">
                <protocol>vnc</protocol>
                <param name="hostname">ip_linux</param>
                <param name="port">5900</param>
                <param name="password">passvnc</param>
        </connection>
        <connection name="ssh-server">
                <protocol>ssh</protocol>
                <param name="hostname">ip_linux_ssh</param>
                <param name="port">22</param>
        </connection>
        </authorize>
</user-mapping>
```

(3) Reiniciaremos el servicio guacamole "**guacd**" para que surtan efectos los cambios realizados en el archivo de configuración user-mapping.xml:

```
[root@servidor]# /etc/init.d/guacd restart
```

(4) Por último, abriremos el navegador y accederemos "guacamole" y a la gestión de las conexiones configuradas: **http://***ip_server***:8080/guacamole**.

Práctica Nº8.- Copias de Seguridad Cifradas almacenadas en Remoto: Duplicity

Un aspecto fundamental que debe ser abordado en la implementación de todo servidor es la realización de copias de seguridad de manera automatizada, con la finalidad de poder salvaguardar los datos ante cualquier desastre imprevisto.

Hace años la realización de estas copias de seguridad se solía realizar mediante la utilización de algún tipo de unidad de almacenamiento externa que se conectaba directamente al equipo servidor (*p.e. cintas magnéticas, CDs, DVDs, etc.*). Aunque este tipo de estrategias de realización de backups se sigue realizando actualmente, se desaconseja su realización ya que ante un desastre (*incendio, robo, etc.*) podrían perderse datos confidenciales muy valiosos. En su lugar se aconseja que la copia de seguridad se almacene en un lugar remoto donde los datos estén bien protegidos, haciendo uso de algún tipo de estrategia de cifrado para el envío de los datos que viajan de origen a destino para que no puedan ser interceptados por un *sniffer*, tal como veremos en la presente práctica.

8.1.- Introducción a Duplicity

Tal como se va a tratar de mostrar en la siguiente práctica, el software "**duplicity**" es una excelente y versátil herramienta de software libre que puede ser usada para la realización de copias de seguridad. En concreto, cabría destacar las siguientes características de **duplicity**:

- Nos permite almacenar la copia de seguridad tanto de manera cifrada (*su comportamiento por defecto*), como sin cifrar. Para ello, tal como veremos hará uso de una pareja de claves asimétricas.
- Nos permite almacenar la copia de seguridad tanto de manera local, como de manera remota. Para transferir la copia de seguridad al almacén de copias de seguridad remoto tendremos la opción de hacerlo vía FTP o SSH.
- Nos permite hacer tanto copias de seguridad completas (*full*), como incrementales. Por defecto, la primera vez que duplicity haga la copia de seguridad de los directorios locales que se le indiquen lo hará en modo "**full**", y el resto de veces se limitará únicamente a hacer un backup de aquellos archivos nuevos o modificados, es decir, realizará un backup "**incremental**". En el caso de que queramos llevar a cabo un nuevo backup completo será necesario indicárselo explícitamente mediante el parámetro "full".

Ahora, si queremos empezar a probar **duplicity** será necesaria su instalación. Aunque esta disponible en los repositorios, es aconsejable hacer uso de la última versión estable de **duplicity**, la cual la podremos obtener directamente desde los repositorios PPA oficiales (*duplicity-team*) para Ubuntu:

```
[root@linux]# apt-get install software-properties-common python-software-properties
[root@linux]# add-apt-repository ppa:duplicity-team/ppa
[root@linux]# apt-get update
```

```
[root@linux]# apt-get install duplicity
```

8.2.- Copias de Seguridad No Cifradas almacenadas en un Directorio Local

Bueno, como la mejor forma de aprender algo es practicando, a continuación se mostrarán diferentes ejemplos prácticos de utilización de **duplicity**.

Comenzaremos por un ejemplo de uso muy sencillo consistente en realizar un **backup no cifrado** de un directorio de nuestro equipo que contiene información supuestamente importante (*p.e. "/home/usulinux1/datos"*), y que almacenaremos en nuestro propio equipo local, en una unidad externa que ha sido montada en "/media/backups" (*p.e. un pendrive USB*). El porque de realizar la copia sin cifrar puede resultar útil cuando queremos que la "disponibilidad" de los datos sea lo más ágil posible, ya que hay que comprender que las operaciones de cifrado y descifrado llevan un coste computacional que hace que el acceso a los datos sea más costoso. Para ello se hará uso del parámetro "**--no-encryption**":

```
[usulinux1@maqlocal]$ duplicity --no-encryption /home/usulinux1/datos file:///media/backups
```

En el caso de que queramos excluir a algún subdirectorio, por ejemplo, "/home/usulinux1/datos/fichas" será tan sencillo como indicarlo a través del parámetro "**--exclude**":

```
[usulinux1@maqlocal]$ duplicity --no-encryption \
                        --exclude /home/usulinux1/datos/fichas \
                        /home/usulinux1/datos file:///media/backups
```

Si por el contrario solo hubiéramos querido realizar el backup de dos subdirectorios y del resto no, por ejemplo, "/home/usulinux1/datos/facturas" y "/home/usulinux1/datos/informes", haríamos uso del parámetro "**--include**":

```
[usulinux1@maqlocal]$ duplicity --no-encryption \
                                --include /home/usulinux1/datos/facturas \
                                --include /home/usulinux1/datos/informes \
                                --exclude '**' \
                                /home/usulinux1/datos file:///media/backups
```

Ej. Práctico 8.2.1: Backup Sin Cifrar con destino Local

Suponiendo que existe una cuenta de usuario en el sistema que se llama **usulinux1**, cuyo HOME (~) se localiza en **/home/usulinux1**, ejecuta un backup mediante **duplicity** de tal forma que cumpla los siguientes requisitos:

Directorios a Respaldar	Tipo Backup		Destino Backup
~/Documentos	**full**	**Sin Cifrar**	**/mnt/backups/usulinux1 (/dev/sdb1)**
~/Informes			
~/Confidencial			

Solución Ej. Pr. 8.2.1.I.- Cómo Hacer un Backup Sin Cifrar con destino Local

Para realizar el backup sugerido tan sólo será necesario ejecutar el siguiente comando:

```
[usulinux1@maqlocal]$ duplicity full --no-encryption \
                                     --include /home/usulinux1/Documentos \
                                     --include /home/usulinux1/Informes \
                                     --include /home/usulinux1/Confidencial \
                                     --exclude '**' \
                                     /home/usulinux1 file:///mnt/backups/usulinux1
```

¡¡Observación!! Observa en los ejemplos anteriores que para especificar la ruta de destino del backup se hace uso de "file:" seguido de 3 barras "///": "**file:**///mnt/backups/usulinux1". La razón de porque son tres barras es porque dos de ellas se usan en la especificación del protocolo "**file://**" y la tercera es la correspondiente a la especificación de la ruta absoluta del directorio de destino "/mnt/backups/usulinux1".

¡¡Advertencia!! Advierte que para indicar a **duplicity** que deseamos hacer una copia o backup de tipo **full** es necesario indicarlo explícitamente como parámetro (*duplicity full …*), ya que por defecto **duplicity** lo hace de forma **incremental** (*respalda los archivos nuevos o modificados que son detectados en relación al último backup realizado*). En el caso de que el backup se haga por primera vez se podrá obviar el parámetro **full** y hacerlo de forma **incremental**, ya que el primer backup **incremental** siempre es equivalente a uno **full** al detectarse que todos los archivos son nuevos y que por tanto tienen que ser respaldados.

Para verificar el estado del backup (*nos permite saber si algún archivo ha sido modificado o se ha creado alguno nuevo*) o para listar los archivos que contiene el backup realizado ejecutaremos los siguientes comandos:

```
[usulinux1@maqlocal]$ duplicity verify --no-encryption \
                                     --include /home/usulinux1/Documentos \
                                     --include /home/usulinux1/Informes \
                                     --include /home/usulinux1/Confidencial \
                                     --exclude '**' \
                                     file:///mnt/backups/usulinux1 /home/usulinux1
[usulinux1@maqlocal]$ duplicity list-current-files file:///mnt/backups/usulinux1
```

Si realizáramos varios backups podríamos obtener una mínima información sobre todos ellos ejecutando el siguiente comando (*podríamos conocer la fecha de su creación con la finalidad de poder especificar una de ellas en una restauración*):

```
[usulinux1@maqlocal]$ duplicity --no-encryption \
                      collection-status file:///mnt/backups/usulinux1
```

Si quisiéramos probar a restaurar la copia de seguridad realizada, podríamos eliminar alguno de los directorios respaldados y recuperarlos mediante duplicity:

```
[usulinux1@maqlocal]$ rm -Rf /home/usulinux1/Documentos /home/usulinux1/Informes ...
[usulinux1@maqlocal]$ duplicity --no-encryption --force restore \
                      file:///mnt/backups/usulinux1 /home/usulinux1
```

En el caso de querer restaurar un archivo o directorio en concreto deberemos indicar en primer lugar que archivo o directorio restaurar, después la ubicación del backup y por último donde restaurarlo:

```
[usulinux1@maqlocal]$ rm -f /home/usulinux1/Documentos/doc1.pdf
[usulinux1@maqlocal]$ duplicity --no-encryption \
                      [restore] --file-to-restore Documentos/doc1.pdf \
                      file:///mnt/backups/usulinux1 /home/usulinux1/Documentos/doc1.pdf

[usulinux1@maqlocal]$ rm -Rf /home/usulinux1/Documentos
[usulinux1@maqlocal]$ duplicity --no-encryption \
                      [restore] --file-to-restore Documentos \
                      file:///mnt/backups/usulinux1 /home/usulinux1/Documentos
```

8.3.- Copias de Seguridad Cifradas almacenadas en un Directorio Local

Tal como se explicó durante el capítulo o práctica relativa a la criptografía existen dos estrategias para establecer una comunicación cifrada: usar claves simétricas o asimétricas. En concreto, aquí veremos como **duplicity** hace uso de claves asimétricas con la finalidad de cifrar el resultado del backup y así garantizar su confidencialidad.

Si en el apartado anterior vimos como realizar backups sin cifrar en modo local, en este segundo caso explicaré como realizar exactamente lo mismo que antes pero con la diferencia de que el backup se almacenará de manera cifrada (*su comportamiento por defecto*). Para ello será necesario que el usuario que haga uso de **duplicity** se cree un par de claves asimétricas mediante "**gpg**" previamente, tal como vimos en la práctica relativa a al criptografía:

```
[usulinux1@maqlocal]$ gpg --gen-key
```

El comando "**gpg --gen-key**" nos permitirá generar unas claves asimétricas que nos podrán servir tanto para cifrar información, como para firmar digitalmente documentos, o únicamente útiles para firmar documentos. En nuestro caso, las necesitaremos del primer tipo, por eso escogeremos la opción (1) entre las que se muestran a elegir:

```
Por favor seleccione tipo de clave deseado:
  (1) RSA y RSA (predeterminado)
  (2) DSA y Elgamal
  (3) DSA (sólo firmar)
  (4) RSA (sólo firmar)
```

En cuanto a la longitud de las claves, por la cual se nos preguntará a continuación, para hacer pruebas, será mejor elegir "1024 bits" ya que su generación y utilización son menos costosas

desde el punto de vista computacional:

```
las claves RSA pueden tener entre 1024 y 4096 bits de longitud.
¿De qué tamaño quiere la clave? (2048) 1024
```

Después se decidirá el periodo de validez de las claves asimétricas, y el nombre y apellidos entre otros datos del usuario que hará uso de ellas:

```
¿Validez de la clave (0)?
La clave nunca caduca
¿Es correcto? (s/n) s
```

```
Nombre y apellidos: UsuLinux1
Dirección de correo electrónico: usulinux1@gmail.com
Comentario: Usuario Linux 1
Está usando el juego de caracteres `utf-8'.
Ha seleccionado este ID de usuario:
«UsuLinux1 (Usuario Linux 1) <usulinux1@gmail.com>»
¿Cambia (N)ombre, (C)omentario, (D)irección o (V)ale/(S)alir?
```

Por último, antes de generar las claves, "gpg" nos preguntará por una frase de acceso o **passphrase** que se nos solicitará cada vez que queramos hacer uso de la parte secreta de las claves asimétricas que se están generando:

```
Introduzca frase contraseña:
```

En el caso de que no queramos que se nos solicite ese **passphrase** cada vez que hagamos uso en **duplicity** de sus claves asociadas, tenemos la opción de crear una variable de entorno cuyo valor coincida con la dicha frase, la cual será leída al ejecutar "duplicity":

```
[usulinux1@maqlocal]$ export PASSPHRASE="frase indicada en la creación de la clave"
```

¡¡Observación!! El uso de variables de entorno va a ser muy habitual en esta práctica, ya que tal como veremos más adelante, la realización de copias de seguridad se suele hacer de manera automatizada (*p.e. Cron*), y en tal situación no habrá nadie para que pueda teclear la contraseña cuando esta sea solicitada. En su lugar, **duplicity** consultará las variables de entorno que tiene predeterminadas.

Una vez generadas las claves se puede listar las claves que posee el usuario local mediante el uso de cualquiera de los siguientes comandos (*el primero lista tanto las claves públicas propias como las claves públicas importadas de otros usuarios, y el segundo las secretas*):

```
[usulinux1@maqlocal]$ gpg --list-key
[usulinux1@maqlocal]$ gpg --list-secret-keys
```

```
pub   1024R/2DE13969 2012-01-23
uid   UsuLinux1 (Usuario Linux 1) <usulinux1@gmail.com>
```

```
sub   1024R/23C15BA4 2012-01-23
```

De la clave generada, nos interesará su identificador, "2DE13969", el cual será utilizado en el comando "**duplicity**" para hacer referencia a estas claves generadas.

Una vez generadas las claves, ya podremos ejecutar el comando "**duplicity**" tal como se muestra en el siguiente ejemplo, en el cual se muestra como hacer una copia de seguridad de los subdirectorios deseados, dejando el backup correspondiente de manera cifrada en el "Dropbox" del usuario, con la finalidad de tener una copia de seguridad en la "nube" de "Dropbox":

```
[usulinux1@maqlocal]$ duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                      --include /home/usulinux1/datos/facturas \
                      --include /home/usulinux1/datos/informes \
                      --exclude '**' \
                      /home/usulinux1/datos file:///home/usulinux1/Dropbox/backups
```

Una vez realizada la copia de seguridad convendría eliminar la variable de entorno que almacena la frase de paso:

```
[usulinux1@maqlocal]$ unset  PASSPHRASE
```

Por tanto, un "script" que realizará el backup anterior podría ser de la siguiente forma:

```
[usulinux1@maqlocal]$ nano script-backup.sh
#!/bin/bash
export PASSPHRASE="frase indicada en la creación de la clave"
duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                      --include /home/usulinux1/datos/facturas \
                      --include /home/usulinux1/datos/informes \
                      --exclude '**' \
                      /home/usulinux1/datos file:///home/usulinux1/Dropbox/backups
unset  PASSPHRASE
```

Ej. Práctico 8.3.1: Backup Cifrado con destino Local

Crea un script que implemente el backup solicitado en el ejercicio práctico anterior pero ahora de manera cifrada, haciendo uso de las claves asimétricas que la cuenta de usuario del sistema **usulinux1** genero previamente (*gpg --gen-key*).

Directorios a Respaldar	Tipo Backup		Destino Backup
~/Documentos	**Incremetal**	**Cifrado**	**/mnt/backups/usulinux1 (/dev/sdb1)**
~/Informes			
~/Confidencial			

Solución Ej. Pr. 8.3.1.I.- *Cómo Hacer un Backup Cifrado con destino Local*

Para realizar el backup sugerido tan sólo será necesario crear el siguiente script y luego ejecutarlo:

```
[usulinux1@maqlocal]$ nano script-backup.sh
#!/bin/bash
export PASSPHRASE="frase indicada en la creación de la clave"
duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                                    --include /home/usulinux1/Documentos \
                                    --include /home/usulinux1/Informes \
                                    --include /home/usulinux1/Confidencial \
                                    --exclude '**' \
                                    /home/usulinux1 file:///mnt/backups/usulinux1
unset  PASSPHRASE
```

```
[usulinux1@maqlocal]$ bash script-backup.sh
```

Para verificar el estado del backup (*nos permite saber si algún archivo ha sido modificado o se ha creado alguno nuevo*) o para listar los archivos que contiene el backup realizado ejecutaremos los siguientes comandos:

```
[usulinux1@maqlocal]$ export PASSPHRASE="frase indicada en la creación de la clave"
[usulinux1@maqlocal]$ duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                            verify \
                            --include /home/usulinux1/Documentos \
                            --include /home/usulinux1/Informes \
                            --include /home/usulinux1/Confidencial \
                            --exclude '**' \
                            file:///mnt/backups/usulinux1 /home/usulinux1
[usulinux1@maqlocal]$ duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                            list-current-files file:///mnt/backups/usulinux1
[usulinux1@maqlocal]$ unset PASSPHRASE
```

Si quisiéramos probar a restaurar la copia de seguridad realizada, podríamos eliminar alguno de los directorios respaldados y recuperarlos mediante duplicity:

```
[usulinux1@maqlocal]$ rm -Rf /home/usulinux1/Documentos /home/usulinux1/Informes ...
[usulinux1@maqlocal]$ export PASSPHRASE="frase indicada en la creación de la clave"
[usulinux1@maqlocal]$ duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                            --force restore file:///mnt/backups/usulinux1 /home/usulinux1
[usulinux1@maqlocal]$ unset PASSPHRASE
```

En el caso de querer restaurar un archivo o directorio en concreto:

```
[usulinux1@maqlocal]$ rm -Rf /home/usulinux1/Documentos
[usulinux1@maqlocal]$ export PASSPHRASE="frase indicada en la creación de la clave"
```

```
[usulinux1@maqlocal]$ duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                      restore --file-to-restore Documentos \
                      file:///mnt/backups/usulinux1 /home/usulinux1/Documentos
[usulinux1@maqlocal]$ unset PASSPHRASE
```

8.4.- Copias de Seguridad Cifradas almacenadas Remotamente vía FTP

Como ya se ha explicado al comienzo de la práctica, para minimizar el efecto de posibles desastres que se pudieran dar (*robo, incendio, etc.*) convendría alojar el resultado del backup en un servidor remoto. A modo de ejemplo, en este apartado de la práctica veremos como duplicity puede enviárnoslo vía FTP.

Para su comprobación deberemos disponer de un servicio FTP en alguna máquina remota (*p.e. otra máquina virtual*). Una de las opciones más sencillas sería instalar proftpd en alguna máquina "**apt-get install proftpd**" (*no requiere de ninguna post-configuración para empezar a trabajar con él*), y crear una cuenta de usuario en cuyo HOME se almacenará el resultado del backup (*toda cuenta de usuario del sistema por defecto es una cuenta FTP válida para proftpd*).

Una vez instalado un servidor FTP, por defecto, ya se podrá acceder al servicio a través de cualquiera de las cuentas de usuario que tenga creadas en el sistema. En la máquina local desde la cual vamos a realizar el backup, para evitar que el comando "duplicity" nos solicite la "password" del usuario con el que nos queremos conectar vía FTP a la máquina remota, crearemos una variable de entorno, cuyo valor será el de dicha password:

```
[usulinux1@maqlocal]$ export FTP_PASSWORD="password del usuario FTP"
```

¡¡Observación!! Como ya se indicó en el apartado práctico anterior, el uso de variables de entorno va a ser muy habitual durante esta práctica, ya que tal como veremos más adelante, la realización de copias de seguridad se suele hacer de manera automatizada (*p.e. Cron*), y en tal situación no habrá nadie para que pueda teclear la contraseña cuando esta sea solicitada. En su lugar, **duplicity** consultará las variables de entorno que tiene predeterminadas: **PASSPHRASE** y **FTP_PASSWORD**.

Además será necesario tener instalado el cliente FTP "**ncftp**" que por defecto utilizará "duplicity" para transferir vía FTP la copia de seguridad desde el equipo local al equipo remoto. En caso de no disponer de él, será necesario instalarlo:

```
[usulinux1@maqlocal]$ sudo apt-get install ncftp
```

Después ya podremos ejecutar el comando "duplicity" con una sintaxis similar a la que se ha visto anteriormente (*donde usuftp será la cuenta de usuario creada en el sistema donde esta instalado el proftpd*):

```
[usulinux1@maqlocal]$ duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                      --include /home/usulinux1/datos/facturas \
                      --include /home/usulinux1/datos/informes \
```

```
                    --exclude '**' \
                    /home/usulinux1/datos ftp://usuftp@ip_maq_remota/dir_backups
```

En el comando anterior, se ha dado por hecho que el usuario de la máquina remota, "**usuftp**" con el que nos conectamos vía FTP, tiene en su HOME un subdirectorio llamado "backups" donde se almacenará la copia de seguridad.

Una vez realizada la copia de seguridad convendría eliminar la variable de entorno que almacena la password FTP:

```
[usulinux1@maqlocal]$ unset  FTP_PASSWORD
```

Por tanto, según todo lo visto hasta ahora, un "script" que realizará el backup anterior podría ser de la siguiente forma:

```
[usulinux1@maqlocal]$ nano script-backup-ftp.sh
#!/bin/bash
export PASSPHRASE="frase indicada en la creación de la clave"
export FTP_PASSWORD="password de acceso al servicio FTP no anónimo"
duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                    --include /home/usulinux1/datos/facturas \
                    --include /home/usulinux1/datos/informes \
                    --exclude '**' \
                    /home/usulinux1/datos ftp://usuftp@ip_maq_remota/dir_backups
unset  PASSPHRASE
unset  FTP_PASSWORD
```

Ej. Práctico 8.4.1: Backup Cifrado almacenado en un Servidor FTP

Crea un script que implemente el backup especificado en la siguiente tabla:

Directorios a Respaldar	Tipo Backup		Destino Backup
~/Documentos	**Incremental**	**Cifrado**	**ftp://usuftp@192.168.1.100/bks**
~/Informes			
~/Confidencial			

Se supondrá que se ha configurado un servicio FTP en un equipo servidor (*p.e. 192.168.1.100*), p.e. **proftpd**:

```
[root@servidorftp]# apt-get install proftpd
[root@servidorftp]# useradd -m -d /home/usuftp -s /bin/bash usuftp
[root@servidorftp]# passwd usuftp
```

Solución Ej. Pr. 8.4.1.I.- *Cómo Hacer un Backup Cifrado con destino Remoto FTP*

Para realizar el backup sugerido tan sólo será necesario crear el siguiente script y luego ejecutarlo:

```
[usulinux1@maqlocal]$ nano script-backup.sh
#!/bin/bash
export PASSPHRASE="frase indicada en la creación de la clave"
export FTP_PASSWORD="password de acceso al servicio FTP no anónimo de usuftp"
duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                                    --include /home/usulinux1/Documentos \
                                    --include /home/usulinux1/Informes \
                                    --include /home/usulinux1/Confidencial \
                                    --exclude '**' \
                                    /home/usulinux1 ftp://usuftp@192.168.1.100/bks
unset  PASSPHRASE
unset  FTP_PASSWORD
```

```
[usulinux1@maqlocal]$ bash script-backup.sh
```

Para verificar el estado del backup (*nos permite saber si algún archivo ha sido modificado o se ha creado alguno nuevo*) o para listar los archivos que contiene el backup realizado ejecutaremos los siguientes comandos (*podría hacerse con otro script de una manera más comoda*):

```
[usulinux1@maqlocal]$ export FTP_PASSWORD="password del usuario usuftp"
[usulinux1@maqlocal]$ export PASSPHRASE="frase indicada en la creación de la clave"
[usulinux1@maqlocal]$ duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                        verify \
                        --include /home/usulinux1/Documentos \
                        --include /home/usulinux1/Informes \
                        --include /home/usulinux1/Confidencial \
                        --exclude '**' \
                        ftp://usuftp@192.168.1.100/bks /home/usulinux1
[usulinux1@maqlocal]$ duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                        list-current-files ftp://usuftp@192.168.1.100/bks
[usulinux1@maqlocal]$ unset PASSPHRASE
[usulinux1@maqlocal]$ unset FTP_PASSWORD
```

Si quisiéramos probar a restaurar la copia de seguridad realizada, podríamos eliminar alguno de los directorios respaldados y recuperarlos mediante duplicity:

```
[usulinux1@maqlocal]$ rm -Rf /home/usulinux1/Documentos /home/usulinux1/Informes ...
[usulinux1@maqlocal]$ export FTP_PASSWORD="password del usuario usuftp"
[usulinux1@maqlocal]$ export PASSPHRASE="frase indicada en la creación de la clave"
[usulinux1@maqlocal]$ duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                        --force restore \
                        ftp://usuftp@192.168.1.100/bks /home/usulinux1
```

```
[usulinux1@maqlocal]$ unset PASSPHRASE
[usulinux1@maqlocal]$ unset FTP_PASSWORD
```

En el caso de querer restaurar un archivo o directorio en concreto:

```
[usulinux1@maqlocal]$ export FTP_PASSWORD="password del usuario ftp"
[usulinux1@maqlocal]$ export PASSPHRASE="frase indicada en la creación de la clave"
[usulinux1@maqlocal]$ rm -Rf /home/usulinux1/Documentos
[usulinux1@maqlocal]$ duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                        restore --file-to-restore Documentos \
                        ftp://usuftp@192.168.1.100/bks /home/usulinux1/Documentos
[usulinux1@maqlocal]$ unset PASSPHRASE
[usulinux1@maqlocal]$ unset FTP_PASSWORD
```

¡¡Observación!! En el caso de que el servidor FTP no permita hacer conexiones FTP de manera pasiva, sería conveniente definir un rango de puertos para ello, ya que puede ser necesario para que el servidor nos acepte la solicitud. En el caso de usar proftpd tan sólo sería neceario indicar la siguiente directiva en /etc/proftpd/proftpd.conf:

```
[root@servidorftp]$ /etc/proftpd/proftpd.conf
PassivePorts 1024 65534
…
```

Obviamente, para que todo sea coherente, será necesario redireccionar el rango de puertos indicados en la directiva **PassivePorts** en el router ISP hacía dicho servidor FTP.

8.5.- Copias de Seguridad Cifradas almacenadas Remotamente vía SSH

Por último, se mostrará otra posible alternativa a la hora de guardar la copia de seguridad. En concreto, vamos a ver como guardar el backup en un servidor remoto haciendo uso del protocolo SCP/SSH, con las ventajas que ello conlleva: canal de comunicaciones seguro mediante la utilización de un sistema híbrido de claves simétricas/asimétricas (*para ver otras posibles estrategias se recomienda ojear la ayuda "man duplicity"*).

¡¡Importante!! Para que la copia de seguridad se pueda realizar de manera desatendida es conveniente que el usuario que va a realizarla sea un usuario confiable para el servidor SSH/SCP. Para ello, tal como se mostró en la práctica de SSH, debe generar un par de claves asimétricas (*ssh-keygen*) y transferir la clave pública a la cuenta de usuario SSH del servidor donde se va almacenar la copia (*ssh-copy-id*).

```
[usulinux1@maqlocal]$ ssh-keygen (no es necesario ni introducir frase de paso, ni modificar la ubicación de las claves; puede pulsarse al INTRO sin más)
[usulinux1@maqlocal]$ ssh-copy-id -i ~/.ssh/id_rsa.pub  usuario_ssh@ip_servidor_SSH
```

De esta forma, el la cuenta del usuario remoto del servidor SSH tendrá almacenada en su fichero "**~/.ssh/authorized_keys**" la clave pública del usuario que va a hacer la copia de seguridad, pasando a ser un usuario confiable al cual no se le solicitará login/password al ejecutar el comando

SCP. Es decir, mediante dicha clave pública el servidor SSH corroborará la autenticidad del usuario que trata de establecer la conexión SSH/SCP (*comprobará la firma que se envía con la solicitud de conexión*), dejándole acceder en caso de que la comprobación sea correcta.

En concreto, la sintaxis del comando **duplicity** que nos permitirá realizar el backup será la siguiente:

```
[usulinux1@maqlocal]$ duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                      [--ssh-askpass] \
                      --include /home/usulinux1/datos/facturas \
                      --include /home/usulinux1/datos/informes \
                      --exclude '**' \
                      /home/usulinux1/datos scp://usuario_ssh@ip_servidor_SSH/directorio
```

Donde el parámetro "**--ssh-askpass**" será necesario ponerlo cuando necesitemos que nos pregunte por la password del usuario del equipo remoto. En nuestro caso no es necesario, ya que el usuario que va a hacer la copia de seguridad es un usuario confiable, al encontrarse su clave pública en el "**~/.ssh/authorized_keys**" del usuario remoto.

Ej. Práctico 8.5.1: Backup Cifrado almacenado en un Servidor SSH/SCP

Crea un script que implemente el backup especificado en la siguiente tabla:

Directorios a Respaldar	Tipo Backup		Destino Backup
~/Documentos	**Incremental**	**Cifrado**	**scp://ususshh@192.168.1.100/bks**
~/Informes			
~/Confidencial			

Solución Ej. Pr. 8.5.1.I.- Cómo Hacer un Backup Cifrado con destino Remoto SCP

Para realizar el backup sugerido tan sólo será necesario crear el siguiente script y luego ejecutarlo:

```
[usulinux1@maqlocal]$ ssh-keygen
[usulinux1@maqlocal]$ ssh-copy-id -i ~/.ssh/id_rsa.pub  ususshh@192.168.1.100
[usulinux1@maqlocal]$ nano script-backup.sh
#!/bin/bash
export PASSPHRASE="frase indicada en la creación de la clave"
duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                                   --include /home/usulinux1/Documentos \
                                   --include /home/usulinux1/Informes \
                                   --include /home/usulinux1/Confidencial \
                                   --exclude '**' \
                                   /home/usulinux1 scp://ususshh@192.168.1.100/bks
unset  PASSPHRASE
```

```
[usulinux1@maqlocal]$ bash script-backup.sh
```

Para verificar el estado del backup (*nos permite saber si algún archivo ha sido modificado o se ha creado alguno nuevo*) o para listar los archivos que contiene el backup realizado ejecutaremos los siguientes comandos:

```
[usulinux1@maqlocal]$ export PASSPHRASE="frase indicada en la creación de la clave"
[usulinux1@maqlocal]$ duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                        verify \
                        --include /home/usulinux1/Documentos \
                        --include /home/usulinux1/Informes \
                        --include /home/usulinux1/Confidencial \
                        --exclude '**' \
                        scp://ususshh@192.168.1.100/bks /home/usulinux1
[usulinux1@maqlocal]$ duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                        list-current-files scp://ususshh@192.168.1.100/bks
[usulinux1@maqlocal]$ unset PASSPHRASE
```

Si quisiéramos probar a restaurar la copia de seguridad realizada, podríamos eliminar alguno de los directorios respaldados y recuperarlos mediante duplicity:

```
[usulinux1@maqlocal]$ rm -Rf /home/usulinux1/Documentos /home/usulinux1/Informes ...
[usulinux1@maqlocal]$ export PASSPHRASE="frase indicada en la creación de la clave"
[usulinux1@maqlocal]$ duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                        --force restore scp://ususshh@192.168.1.100/bks /home/usulinux1
[usulinux1@maqlocal]$ unset PASSPHRASE
```

En el caso de querer restaurar un archivo o directorio en concreto:

```
[usulinux1@maqlocal]$ export PASSPHRASE="frase indicada en la creación de la clave"
[usulinux1@maqlocal]$ rm -Rf /home/usulinux1/Documentos
[usulinux1@maqlocal]$ duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                        restore --file-to-restore Documentos \
                        scp://ususshh@192.168.1.100/bks /home/usulinux1/Documentos
[usulinux1@maqlocal]$ unset PASSPHRASE
```

¡¡Observación!! Tras los ejercicios prácticos propuestos se habrá podido comprender la potencia que presenta el comando **duplicity**. Además de las opciones del comando ya usadas, cabría destacar las siguientes:

Opción Duplicity	Ejemplo y Descripción
full/incremental	**duplicity full --encrypt-key "2DE13969" --sign-key "2DE13969" \ /home/usulinux1 scp://ususshh@192.168.1.100/bks**
	**duplicity [incremental] \ --encrypt-key "2DE13969" --sign-key "2DE13969" **

	/home/usulinux1 scp://ususshh@192.168.1.100/bks
	Nos permite indicar el tipo de copia a realizar. Por defecto es incremental, lo que implica que la primera vez que se ejecute será full al no haberse echo antes, y posteriormente tan sólo se hará un backup de los archivos nuevos y aquellos que hayan sido modificados. Para hacer un backup full a posteriori, tiene que indicarse expresamente.
remove-older-than *time*	**duplicity remove-older-than 6M scp://ususshh@192.168.1.100/bks**
	duplicity remove-older-than 1Y ftp://usuftp@192.168.1.100/bks
	Elimina las copias de seguridad de hace más del "time" indicado como parámetro (*un mes*, ***1M***, *2 meses*, ***2M***, *…, un año*, ***1Y***, *etc.*)

Ej. Práctico 8.5.2: Configuración de Backups Automáticos mediante Cron

Suplanta al usuario que hayas estado utilizando durante todos los ejercicios prácticos (*p.e. usulinux1*) y configura el backup que se especifica a continuación teniendo en cuenta las siguientes premisas:

(I) Se realizará un backup **incremental** todos los días a la 1:30h de la madrugada.
(II) Todos los viernes a las 23:45h se realizará un backup de tipo **full**.
(III) Una vez al año, el 31 de diciembre a las 12:00, se eliminarán los backups de hace más de 1 año.
(IV) Tras realizar el backup se enviará un email (*p.e. mediante PHPMailer*) al usuario para advertir de su resultado (*p.e. usulinux1@gmail.com*).

Directorios a Respaldar	Tipo Backup		Destino Backup
~/Documentos	**Incremental / Full**	**Cifrado**	**scp://ususshh@192.168.1.100/bks**
~/Informes			
~/Confidencial			

Solución Ej. Pr. 8.5.2.I.- Cómo Realizar Backups de Manera Automatizada

Para dar solución al ejercicio propuesto, primero crearemos el script encargado de hacer la copia de respaldo, después configuraremos el servicio **crontab** y por último explicaremos brevemente como configurar PHPMailer para el envío de los emails de aviso.

(1) En relación al script a implementar para realizar las copias de seguridad, se basará en el mismo que se ha utilizado en el ejercicio práctico anterior, pero pasándole como parámetro al comando **duplicity** ($1) el tipo de backup a realizar (*full o incremental*), además de comprobar mediante un **if** su ejecución para mandar un correo u otro (*éxito o fracaso*):

```
[usulinux1@maqlocal]$ ssh-keygen
[usulinux1@maqlocal]$ ssh-copy-id -i ~/.ssh/id_rsa.pub  ususshh@192.168.1.100
```

```
[usulinux1@maqlocal]$ nano script-backup.sh
#!/bin/bash
export PASSPHRASE="frase indicada en la creación de la clave"
if duplicity $1 --encrypt-key "2DE13969" --sign-key "2DE13969" \
                                        --include /home/usulinux1/Documentos \
                                        --include /home/usulinux1/Informes \
                                        --include /home/usulinux1/Confidencial \
                                        --exclude '**' \
                                        /home/usulinux1 scp://ususSH@192.168.1.100/bks
then
        php /home/usulinux1/mandar-email-exito.php
else
        php /home/usulinux1/mandar-email-error.php
fi
unset  PASSPHRASE
```

(2) El usuario usulinux1 configurar su servicio crontab para que se automaticen los requisitos establecidos en el enunciado:

```
[usulinux1@maqlocal]$ crontab -e
#minuto hora dia mes dia-semana ruta_programa/script_ejecutar
30 1 * * * /home/usulinux1/script-backup.sh incremental >> /home/usulinux1/auditoria1.log
45 23 * * 5 /home/usulinux1/script-backup.sh full >> /home/usulinux1/auditoria2.log
0 12 31 12 * /usr/bin/duplicity --encrypt-key "2DE13969" --sign-key "2DE13969"
remove-older-than 1Y scp://ususSH@192.168.1.100/bks >> /home/usulinux1/auditoria3.log
```

(3) Respecto a la configuración de PHPMailer para el envío de correos, tan sólo habrá que descargarse la clase **phpmailer** y descomprimirla en el mismo directorio donde se ubican los scripts PHP que enviarán los emails, crear un objeto **phpmailer** y definir un valor para las propiedades del objeto creado (*destinatario*, *remitente*, *asunto*, *cuerpo mensaje*, *etc.*). A modo de ejemplo a continuación se muestra como podría ser el código PHP del script **mandar-email-exito.php**:

```
[usulinux1@maqlocal]$ nano /home/usulinux1/mandar-email-exito.php
<?
require_once 'phpmailer/class.phpmailer.php';
$mail = new phpmailer();
$mail->Mailer = "smtp";
$mail->Host = "ssl://smtp.gmail.com";
$mail->Port="465";
$mail->SMTPAuth = true;
$mail->Timeout=30;
// Indicamos la cuenta de gmail desde la que se envía el email
$usuario_gmail = "alumnoiestm@gmail.com";
$alias_remitente_email = "Avisador Backup";
$pass_usuario_gmail = "alumno1234";
$mail->From = $usuario_gmail;
$mail->FromName = $alias_remitente_email;
```

```
$mail->Username = $usuario_gmail;
$mail->Password = $pass_usuario_gmail;
// Indicamos cuantos y a quienes lo enviamos. Por ejemplo:
$numEmails = 1;
$email = array ("amartinromero@gmail.com");
for ($i=0; $i<$numEmails; $i++){
        $mail->AddAddress($email[$i]);
}
// Indicamos el asunto y el cuerpo del mensaje (se puede usar lenguaje HTML)
$asunto = "Aviso: Backup Realizado Correctamente";
$comando = "date +'%R-%D'";
exec ($comando,$fecha);
$cuerpoMail = "<fieldset style='background-color: orange; color: blue;'>
<h3 style='text-align: center;'>Backup Realizado con Exito <br>".$fecha[0]."</h3>
El backup se realizo sin problemas. No hay que preocuparse!!
</fieldset>";
$mail->Subject = $asunto;
$mail->Body = $cuerpoMail;
$mail->AltBody = $cuerpoMail;
// Si queremos adjuntar algún archivo podemos indicarlo, o dejarlo vacío
/*
$ruta_archivo = "/home/usulinux1/auditoria1.log";
$nombre_archivo = "auditoria1.log";
$mail->AddAttachment($ruta_archivo, $nombre_archivo);
*/
// Finalmente enviamos el email
$resultado = $mail->Send();
?>
```

¡¡Observación!! Como habrá podido observarse a lo largo de la práctica, **duplicity** va almacenando las copias de seguridad que va realizando identificándolas con la fecha de creación. Esto nos permite poder seleccionar la fecha de aquello que queremos restaurar, y de aquello que queramos eliminar. Por ejemplo, si quisiéramos recuperar un archivo en el estado que se encontraba hace 12 días podríamos ejecutar el siguiente comando:

```
[usulinux1@maqlocal]$ duplicity --encrypt-key "2DE13969" --sign-key "2DE13969" \
                      restore --file-to-restore -t 12D Documentos/midoc1.doc \
                      scp://ususshh@192.168.1.100/bks /home/usulinux1/Documentos/midoc1.doc
```

Práctica Nº9.- Uso de la Criptografía en los Protocolos de Comunicaciones: HTTPS, FTPS o SFTP

En la presente práctica advertiremos como prácticamente todos los protocolos actuales tienen una versión en modo seguro (*HTTP/HTTPS, FTP/FTPS, SMTP/SMTPS, etc.*) con la finalidad de garantizar alguna de las dos siguientes características:

(1) Autenticidad:

Hoy en día Internet nos permite hacer prácticamente de todo a través de la red: realizar transferencias bancarias, hacer compras online, reservar hoteles o vuelos, etc. En muchas de estas ocasiones es muy importante poder corroborar que el servicio al que me conecto realmente lo ofrece quien dice ser que es, y que no se trata de un atacante que lo esta suplantando. Es decir, cuando introducimos nuestros datos personales, o los datos de nuestra tarjeta de crédito en un formulario HTML de un sitio Web, nos gustaría tener la certeza de que esos datos realmente los va recoger y usar la entidad destinataria esperada y no otra que lo esta suplantando.

Tal como vimos en el tema de criptografía la solución es bien sencilla: el servicio al que me conecto me entregará un documento firmado por ella con su clave secreta, y nosotros con su clave pública corroboraremos la autenticidad de quien lo ofrece. No obstante, el asunto se complica, ya que como nos aseguramos de que la clave pública que usamos es de quien dice ser que es y no de otro. Para subsanar este escoyo surge las entidades o autoridades de certificación, CA, ya mencionadas en la práctica o capítulo de criptografía. En concreto, la finalidad de la CA será la de firmar con su clave secreta un certificado de autenticidad del emisor, y nosotros haciendo uso de la clave pública o certificado público de la CA (*los certificados públicos de la mayoría de las CAs de confianza ya se encuentran cargadas en los navegadores Web más comunes*) comprobaremos la firma de la CA que nos asegura que ese que dice ser quien es, realmente lo es.

Un ejemplo de implementación de todo esto es el servicio HTTPS, de hay que todos los sitios Web seguros a los cuales nos conectamos lo implementan. En concreto, HTTPS hace uso de la criptografía tanto para garantizar la autenticidad, como la confidencialidad, ya que una vez establecida la comunicación HTTPS entre un equipo servidor y cliente, la comunicación entre ambos ya viaja cifrada siendo prácticamente imposible descifrar que datos están viajando entre ambos. Su implementación se mostrará a continuación como ejercicio práctico.

Además de todo lo anterior, no debemos olvidar la amenaza siempre existente, ya tratada en el capítulo de criptografía, correspondiente a un ataque Man in the Middle (*p.e. Spoofing mediante el software Cain*).

(2) Confidencialidad:

En otras ocasiones nos puede interesar no tanto la autenticidad del emisor, sino garantizar que la comunicación viaje cifrada con la finalidad de asegurar una confidencialidad en los datos que se transfieren entre emisor y receptor. Para ello, la mayor parte de los protocolos en modo seguro actuales hacen uso de técnicas criptográficas híbridas. Es decir, durante el establecimiento de la comunicación en modo seguro haciendo uso de claves asimétricas se pasan una clave simétrica que será utilizada durante el resto de la comunicación para cifrar lo que viaja entre emisor y receptor asegurando una comunicación segura. Esto se hace así ya que desde el punto de vista

computacional, sería mucho más costoso mantener una comunicación haciendo de claves asimétricas que con claves simétricas.

9.1.- Implementación de una Autoridad Certificadora (CA)

Para crear una CA tan sólo necesitaremos crear una pareja de claves asimétricas (*clave pública y privada*), y el certificado auto firmado por la propia entidad CA con todos sus datos. Antes deberemos instalar el software "**openssl**":

```
[root@ca]# apt-get install openssl
```

Ej. Práctico 9.1.1: Creación de una Entidad o Autoridad Certificadora (CA)

En este ejercicio práctico nos crearemos nuestra propia **Autoridad Certificadora**, la cual nos permitirá firmar los certificados necesarios para poder ofrecer un servicio en modo seguro (*HTTPS, FTPS, VPN, etc.*). Es decir, la CA mediante su firma avalará a la empresa o entidad que ofrece el servicio a través de Internet (*banco, tienda online, etc.*), de tal forma que los clientes al poder comprobar dicha firma mediante la clave pública de la CA, confiarán en el servicio y no dudarán a la hora de introducir datos personales y confidenciales (*passwords, números de cuenta bancaria, número de tarjeta, etc.*).

Para la creación de la CA tan sólo necesitaremos crear **una pareja de claves asimétricas** que permitirán tanto firmar como comprobar la firma, y **un certificado auto firmado** por la propia CA que contendrá la parte pública de la clave y que les servirá a los clientes para comprobar la validez de los certificados firmados por dicha CA de las empresas que ofrecen los servicios.

Solución Ej. Pr. 9.1.1.I.- Cómo Crear una Autoridad Certificadora (CA)

Para poder almacenar los archivos resultantes de la creación de la CA, claves y el certificado auto firmado, crearemos un directorio dentro del servidor p.e. **/etc/ca**, y posteriormente generaremos los archivos necesarios.

Además, deberemos tener en cuenta al generar el certificado de la CA su formato, ya que dependiendo de la plataforma y navegador Web usados, será necesario un formato u otro. Es decir, mientras que los formatos más habitual de certificado son ***.pem** o ***.crt**, por ejemplo en Windows se hace uso de ***.p12** o ***.pfx**. Por este motivo, en la resolución del ejercicio crearemos un certificado auto firmado ***.pem**, y luego posteriormente lo convertiremos también en ***.p12** o ***.pfx**, teniendo disponibles ambos.

¡¡Advertencia!! El certificado de la CA lo va a firmar la propia entidad certificadora haciendo uso de su clave secreta. Como es obvio, al no certificarnos nadie que la CA es quien dice ser que es, debemos confiar en ella. Por ello, todas las entidades certificadoras que existen en la actualidad se ganan la confianza en base a su prestigio (*p.e. verisign, http://www.verisign.es*), firmando certificados a entidades sin margen de equivocación, garantizando a los usuarios clientes que el servicio al que se conectan lo esta ofreciendo ofreciendo realmente quien dice ser que es.

```
[root@ca]# mkdir /etc/ca
```

```
[root@ca]# cd /etc/ca
[root@ca:/etc/ca]# openssl genrsa -out ca.pem 2048
[root@ca:/etc/ca]# openssl req -new -x509 -key ca.pem -out cacert.pem -days 1000
...
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Zaragoza
Locality Name (eg, city) []:Zaragoza
Organization Name (eg, company) [Internet Widgits Pty Ltd]:CA IESTM
Organizational Unit Name (eg, section) []:Depart. Informatica
Common Name (e.g. server FQDN or YOUR name) []:www.ca-iestm.es
Email Address []:ca-iestm@gmail.com
```

```
[root@ca:/etc/ca]# openssl pkcs12 -export -out cacert.pfx \
                        -inkey ca.pem -in cacert.pem -certfile  cacert.pem
[root@ca:/etc/ca]# openssl pkcs12 -export -out cacert.p12 \
                        -inkey ca.pem -in cacert.pem -certfile  cacert.pem
```

¡¡Observación!! Para obtener los certificados para la plataforma Windows a partir del certificado ***.pem** o ***.crt** podemos hacerlo vía Web, como por ejemplo, a través de la página Web "https://www.sslshopper.com/ssl-converter.html":

En relación a los parámetros usados en **openssl** destacar los siguientes:

Parámetros del Comando openssl	
genrsa	Indica a **openssl** que las claves asimétricas a generar serán de tipo RSA. El número **2048** es la longitud en bits de la clave. Por defecto es 512 pero actualmente para claves RSA se recomienda un tamaño de 2048 o mayor.
req	Se utiliza para crear y procesar solicitudes de certificado en formato PKCS#10. En este caso la estamos utilizando para autofirmar un certificado que será usado como CA raíz.
-new	Crea una nueva solicitud de certificación preguntando al usuario la información relevante de la entidad certificadora que sea necesaria.
-key	Indica el fichero con las claves asimétricas (*pública y privada*), ya que el certificado

	contendrá la clave pública que permitirá comprobar la firma; en el caso de no haberse utilizado esta opción **openssl** hubiera generado automáticamente una clave RSA.
-x509	Se usa para proporcionar como salida del comando un certificado firmado, en lugar de una solicitud de certificación. Por tanto, esta opción sólo tiene sentido utilizarla si deseamos crear una autoridad de certificación (CA) raíz autofirmada.
-out	Indica el nombre del fichero que contiene el certificado autofirmado de la CA.
-days	Indica el número de días de validez que tendrá el certificado creado. Por defecto el número de días es 30.
pkcs12 -export	Mientras otras plataformas suelen hacer uso de de certificados en formato ***.pem** o ***.crt**, Microsoft Windows hace uso de certificados ***.p12** o ***.pfx**. Estás opciones de openssl nos permiten generar un certificado ***.p12** o ***.pfx** a partir de un certificado ***.pem** o ***.crt**.

¡¡Importante!! Tanto al crear un certificado como al crear una solicitud de certificado mediante **openssl**, se nos van a pedir unos datos. Estos datos deberían corresponderse con los datos reales de la entidad o empresa a la que se certifica. En nuestro caso, al tratarse de una "supuesta" CA, podemos inventarnos todos ellos, teniendo en cuenta que el **Common Name** no debe repetirse en la creación de ningún otro certificado firmado por ella, ya que éste es el parámetro que identifica realmente de manera univoca a la entidad.

¡¡Observación!! En relación al servicio HTTPS que se configurará a continuación, será necesario hacer llegar el certificado creado de la CA (**.pem, *.p12, etc.*) al equipo cliente desde el cual se vayan a realizar las pruebas de conexión con dicho servicio, para que lo importe desde su navegador Web. De esta forma, tal como podrá comprobarse al finalizar la práctica, el cliente Web podrá comprobar la firma del certificado del servicio HTTPS y confiar en él.

9.2.- Creación de las Claves y Certificados para el Servicio HTTPS

Tal como se ha explicado durante la introducción a la presente práctica, con la finalidad de que los usuarios finales que acceden desde su navegador Web al servicio HTTPS confíen en el servicio, y se atrevan a introducir datos confidenciales (*"login" y "password", códigos, etc.*) necesitaremos crear un par de claves asimétricas, y una solicitud de certificado con nuestros datos para que sea firmada por la CA creada anteriormente.

Ej. Práctico 9.2.1: Creación de las Claves y Certificados para el Servicio HTTPS

Una vez creada la autoridad certificadora, CA, en el ejercicio práctico anterior, en esté crearemos los certificados necesarios para implementar un servicio HTTPS. Este certificado firmado por la CA le permitirá a la empresa o entidad que ofrece el servicio HTTPS generar confianza en el cliente (*el cliente Web comprobará la firma, y se dirá a si mismo: "... si la CA lo ha firmado, será porque quien dice ser que es, lo es en verdad ...", pasando a ser un sitio Web de confianza*).

***Solución Ej. Pr. 9.2.1.I.-** Cómo Crear las Claves y Certificados para HTTPS*

En concreto deberemos seguir los siguientes pasos:

(1) Crearemos un directorio se almacenarán todos los archivos necesarios (*p.e.* ***/etc/server***).
(2) La entidad o empresa que quiere ofrecer el servicio HTTPS creará una pareja de claves asimétricas (*p.e.* ***server.pem***). Estas le permitirán ofrecer autenticidad y confidencialidad. Para garantizar una mayor robustez en las claves, serán de 2048 bits de longitud.
(3) La entidad o empresa que quiere ofrecer el servicio HTTPS generará una solicitud de certificado, ***.p10**, donde incluirá los datos de la empresa (*p.e.* ***servercert.p10***).
(4) La solicitud de certificado anterior se enviará a la CA para que compruebe la veracidad de los datos incluidos, y en ese caso, firmar el certificado para avalar a la empresa (*p.e.* ***servercert.pem***).

```
[root@server]# mkdir /etc/server
[root@server]# cd /etc/server
[root@server]# openssl genrsa -out server.pem 2048
[root@server]# openssl req -new -key server.pem -out servercert.p10
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Zaragoza
Locality Name (eg, city) []:Zaragoza
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Tienda Online Profesor
Organizational Unit Name (eg, section) []:Depart. Ventas
Common Name (e.g. server FQDN or YOUR name) []:www.tiendaonlineprofesor.es
Email Address []:tiendaonlineprofesor@gmail.com
```

¡¡Importante!! Entre todos los datos introducidos al crear la solicitud de certificado del servicio Web, el más importante es el "**Common Name**", el cual debe corresponderse con el nombre de dominio cualificado (**FQDN**) asociado al servicio web que se quiere dar (*p.e. "www.miservicioweb.es"*). En una situación real la CA sería la encargada de corroborar que los datos suministrados por la empresa a través de la solicitud de certificado son correctos, y en ese caso, avalarlos con su firma.

Por último, la CA firmará la solicitud anterior generando el correspondiente certificado que le servirá a la empresa que ofrece el servicio HTTPS para asegurar al cliente que acceda a su sitio Web que son un sitio de confianza, y que quien lo esta ofreciendo es quien dice ser que es:

```
[root@ca]# openssl x509 -req -in servercert.p10 -out servercert.pem \
                        -CA /etc/ca/cacert.pem -CAkey /etc/ca/ca.pem -CAcreateserial -days 365
Signature ok
subject=/C=ES/ST=Zaragoza/L=Zaragoza/O=Tienda Online Profesor/OU=Depart. Ventas/CN=www.tiendaonlineprofesor.es/emailAddress=tiendaonlineprofesor@gmail.com
```

9.3.- Configuración Básica de un Servicio en Apache

Antes de empezar a configurar Apache, lo instalaremos (*añadiremos la librería PHP para dar servicio a sitios Web que hacen uso de este lenguaje*) y repasaremos sus archivos y directorios

de configuración más importantes (*/etc/apache2*), además de los comandos del sistema más relevantes asociados a Apache:

[root@apache]# **apt-get install apache2 libapache2-mod-php5**

Archivos y Directorios de Configuración	Descripción
apache2.conf	Archivo de configuración global de Apache. Aunque puede configurarse el servicio Apache editando directamente este archivo, no suele hacerse, ya que se editan otros archivos que están incluidos en él mediante directivas **Include**: [root@servidor]# **more /etc/apache2/apache2.conf | grep Include**
sites-enabled/	Subdirectorio encargado de albergar los archivos de configuración asociados a los distintos sitios Web habilitados (*sites-enabled*) a los cuales da servicio Apache. Para facilitar la gestión y mantenimiento de los sitios Web a servir, estos archivos deberían haberse creado previamente dentro de **sites-available/** y posteriormente habilitados mediante el comando **a2ensite**. Por el contrario, para deshabilitar un sitio Web servido por Apache haremos uso del comando **a2dissite**. [root@servidor]# **nano /etc/apache2/sites-available/misitio.conf** [root@servidor]# **a2ensite misitio.conf** [root@servidor]# **a2dissite misitio.conf**
sites-available/	Subdirectorio encargado de albergar los archivos de configuración de los sitios Web disponibles (*sites-available*) que tenemos pensados que puedan ser servidos por Apache en un momento u otro. Los sitios Web declarados en estos archivos no estarán habilitados hasta que no sean enlazados al subdirectorio **sites-enabled/** mediante el uso del comando **a2ensite**.
ports.conf	Archivo de configuración de Apache donde indicaremos las direcciones IP y puertos del equipo servidor a través de los cuales dará servicio Apache.
conf.d/	Subdirectorio encargado de albergar archivos de configuración del servicio Apache que deseemos que le afecten con carácter general.
mods-enabled/	Una de las características más importantes de Apache es su modularidad. Es decir, Apache dispone de multitud de módulos que permiten personalizar su comportamiento. En este subdirectorio se localizan los módulos disponibles que están habilitados (*mods-enabled*), y que por tanto pueden ser configurados. Es aconsejable únicamente habilitar aquellos módulos cuyas directivas de configuración nos resulten útiles, con la finalidad de cargar lo mínimo posible el servicio. Para habilitar un módulo que se encuentre disponible en **mods-available/** haremos uso del comando **a2enmod**. Por el contrario, para deshabilitarlo utilizaremos **a2dismod**. [root@servidor]# **ls mods-available | grep auth** [root@servidor]# **a2enmod auth_digest authn_dbm**

mods-available/	Este subdirectorio contiene la lista de todos los módulos disponibles en Apache. En el caso de que la funcionalidad que deseemos no nos la pueda proporcionar ninguno de los módulos disponibles será necesario instalar la librería de Apache correspondiente.

En cuanto a las directivas de configuración de Apache, cabría resaltar las siguientes por ser las más comunes:

Directiva de Configuración	Descripción y Ejemplos de Uso
Listen	**Listen** [*direccion-IP:*]*puerto* [*protocolo*]. Establece bajo que dirección IP y puerto de comunicaciones Apache atenderá a las peticiones HTTP que lleguen al equipo servidor. En el caso de no indicar ninguna dirección IP, se entenderá que debe escuchar por el puerto indicado a través de cualquiera de las IP que tenga asignadas nuestro equipo. El protocolo soló se indicará de manera opcional, cuando el puerto indicado no sea un puerto well-know (*0:1023*): **Listen 80** **Listen 192.168.1.1:8080 http** **Listen 192.168.1.1:8000 https**
ServerName	**ServerName** [protocolo://]*nombre-dominio*[:*puerto*]. Asigna un nombre de dominio cualificado (*FQDN, Fully Qualified Domain Name*) al servicio Web al que da servicio Apache. En el caso de que Apache se configure para dar servicio a múltiples sitios Web bajo una misma dirección IP, el criterio que utiliza para su diferenciación es hacer uso de un nombre de dominio diferente (*Hosts Virtuales basados en nombre*). **ServerName http://www.midominio.es:8080** También es posible definir alias del nombre de dominio especificado mediante la directiva "**ServerAlias** *nombre-equipo* [*otros-nombres-equipo*]". Como puede observarse en la sintaxis de uso de la directiva **ServerName**, opcionalmente es posible indicar el tipo de protocolo utilizado y el puerto de escucha. Para ser congruentes, el puerto indicado deberá haberse especificado previamente en la directiva **Listen**.
DocumentRoot	**DocumentRoot** "*directorio-documentos"*. Indica la ruta del directorio raíz que contiene todos los documentos que componen el sitio Web que deseamos servir. Aunque puede ubicarse en cualquier lugar, por cuestiones de organización se recomienda ubicar estos directorios en **/var/www**. **DocumentRoot /var/www/sitioweb1**
DirectoryIndex	**DirectoryIndex** *lista-archivos-inicio*. Indica cual será el documento que será servido por defecto ante una solicitud HTTP/HTTPS entre todos los que componen el sitio Web. En caso de no asignársele valor a esta directiva, el documento que será servidor por defecto será "*index.html*", "*index.pl*", "*index.cgi*", "*index.php*", etc. **DirectoryIndex indice.html inicio.php**
LogFormat	**LogFormat** *formato nombre-formato*. **CustomLog** *ruta-fichero-log formato*.

<table>
<tr><td>CustomLog
TransferLog
ErrorLog</td><td>TransferLog ruta-fichero-log. ErrorLog ruta-fichero-log. Un aspecto muy importante que hay que tener en cuenta cuando se ofrece un servicio es la auditoría. Esta nos permite conocer quien, cuando, desde donde y que hizo al solicitar el servicio. Esta información de auditoría se almacena en los llamados ficheros log. La directiva LogFormat nos permite definir el formato de la información que será almacenada en los ficheros de auditoría, CustomLog especifica la ruta del fichero donde se guarda dicha información haciendo uso de alguno de los formatos creados con LogFormat, y TransferLog es similar a CustomLog, con la diferencia de que no se puede especificar cualquier formato de auditoría, sino que se hará uso de un formato por defecto definido en la configuración de Apache.
LogFormat "%h %l %u %t \"%r\" %>s %b" miformato
CustomLog /var/log/apache/micontrol_acceso.log miformato
TransferLog /var/log/apache/accesos.log
ErrorLog /var/log/apache/errores.log
Para definir el formato puede hacerse uso de las variables siguientes:
<table>
<tr><th>Variable</th><th>Significado</th></tr>
<tr><td>%a %h</td><td>Dirección IP del equipo cliente que solicita la conexión HTTP.</td></tr>
<tr><td>%l %u</td><td>Nombre de usuario con que se accede al servidor.</td></tr>
<tr><td>%t</td><td>Hora en que fue recibida la solicitud de conexión.</td></tr>
<tr><td>%r</td><td>Información sobre la primera línea de la solicitud recibida.</td></tr>
<tr><td>%s</td><td>Información sobre el estado de la solicitud.</td></tr>
</table></td></tr>
<tr><td>VirtualHost</td><td><VirtualHost <i>direccion-IP</i>[:<i>puerto</i>] [<i>direccion-IP</i>[:<i>puerto</i>]] ...>
<i>directivas-configuración</i>
</VirtualHost>
Directiva de configuración utilizada cuando queremos dar servicio a más de un sitio Web. Es necesario indicar una dirección IP y opcionalmente el puerto de comunicaciones del equipo servidor bajo la cual se realizarán escuchas de solicitudes de conexión HTTP al sitio Web que da servicio el VirtualHost.
Listen 80
Listen 8088
NameVirtualHost 192.168.1.1:80
NameVirtualHost 127.0.0.1:8088
<VirtualHost 192.168.1.1:80 127.0.0.1:8088>
ServerName www.sitioweb1.es
DocumentRoot /var/www/sitioweb1
DirectoryIndex inicial.html
</VirtualHost>
<VirtualHost 192.168.1.1:80 127.0.0.1:8088>
ServerName www.sitioweb2.es
DirectoryIndex paginicio.php
</VirtualHost></td></tr>
<tr><td>NameVirtualHost</td><td>En el caso de que se desee dar servicio a más de un sitio Web bajo la misma IP y por el mismo puerto de comunicaciones, será necesario diferenciarlos a</td></tr>
</table>

	través del nombre de dominio asignado a la directiva **ServerName**, informando de ello a Apache mediante la directiva "**NameVirtualHost** *direccion-IP[:puerto]*". Ha esta configuración se denomina *Hosts Virtuales basados en nombre.* **NameVirtualHost 192.168.1.1:80**
Alias	**Alias** */nombre_alias /ruta_directorio*. La directiva **Alias** nos permite acceder desde nuestro sitio Web a otro directorio del sistema que puede encontrarse fuera del **DocumentRoot** del sitio Web. Para hacer uso de un Alias que haya sido definido tan sólo tendremos que añadirlo al final de la URL de nuestro sitio Web, p.e. "http://www.misitioweb.es/*nombre_alias*". **Alias /descargas /mnt/discob/descargas**
UserDir	**Userdir** *ruta-directorio-publicacion* **Userdir enabled** *lista-nombres-cuentas-usuario-permitidas* **Userdir disabled** *lista-nombres-cuentas-usuario-no-permitidas* En ocasiones nos puede interesar que determinadas cuentas de usuario dadas de alta en el sistema puedan publicar contenidos vía Web a través de Apache desde su propio *Home*. **UserDir** es la directiva encargada de especificar la ruta del directorio raíz o *DocumentRoot* dentro del *Home* de los usuarios del sistema donde dejarán los documentos Web que serán publicados por Apache. También nos permite decidir si esta publicación de contenidos es extensible a todas las cuentas de usuario del sistema, o esta restringido a unos determinados usuarios (*enabled*|*disabled*). Si no se indica lo contrario, por defecto, todo usuario podrá aprovecharse de esta opción. **UserDir /home/*/miweb** **UserDir enabled ususis1 ususis2** **UserDir disabled** **<Directory /home/*/miweb>** **Allow from all** **</Directory>** Para hacer uso de esta directiva será necesario cargar el módulo **userdir**: [root@servidor]# **a2enmod userdir** Para acceder al sitio Web servido desde el Home del usuario se especificará al final de la URL "*~nombre_usuario*", p.e. *http://www.misitio.es/~***ususis1**.
<Directory> <Files></Files> </Directory>	Directivas usadas en la configuración de un sitio Web con la finalidad de particularizar el comportamiento de determinadas zonas del sitio Web (archivos o directorios). Por ejemplo, resultan útiles cuando queremos delimitar el acceso a determinados contenidos.

Además, a parte de las directivas anteriores, también cabría hacer referencia a las directivas pertenecientes a los módulos auth_basic, auth_digest, authn_dbm, entre otros, las cuales nos permiten definir zonas de acceso restringido dentro de un sitio Web, accesibles únicamente mediante una autenticación correcta (*login + password*):

Directiva Configuración	Descripción y Ejemplo de Uso
AuthType	**AuthType** Basic|Digest. Indica el tipo de autenticación que se va a

	realizar sobre las áreas restringidas a usuarios. El tipo **basic** requiere del módulo *mod_auth_basic* y **digest** de *mod_auth_digest*. Su diferencia esencial, es que en el primer caso, las contraseñas viajan entre cliente y servidor sin cifrar, y en el segundo cifradas. No todos los clientes Web soportan la autenticación **digest**. **AuthType Basic** **AuthType Digest**
AuthName	**AuthName** "*Dominio Autenticacion – Realm*". Establece un dominio de autenticación o realm. Permite diferenciar entre diferentes areas de acceso restringido dentro de un mismo sitio Web. La mayoría de los clientes Web lo muestran en la ventana de autenticación con la finalidad de que el usuario reconozca dentro de que área de acceso restringido trata de entrar. Una vez validado el nombre y contraseña, el usuario ya no tendrá que volver a introducirlos al acceder a cualquier recurso que pertenezca al mismo dominio de autenticación. **AuthName** "Zona Restrigida N1"
AuthBasicProvider **AuthDigestProvider**	**AuthBasicProvider** file|dbm|dbd|ldap. **AuthDigestProvider** file|dbm|dbd. Indican a Apache de donde se van a obtener los datos para la comprobación de la autenticación (*login y password introducidos por el usuario al tratar de acceder a una área restringida*). Puede hacerse uso de un fichero plano **file**, de un gestor de bases de datos **dbm** (*mod_authn_dbm*) o **dbd** (*mod_authn_dbd*), o un servicio **ldap** (*mod_authnz_ldap*). Dependiendo del valor asignado a la directiva **AuthType**, Basic o Digest, se utilizará **AuthBasicProvider** o **AuthDigestProvider**. **AuthName "Area Restringida 1"** **AuthType Basic** **AuthBasicProvider dbm**
AuthDBMType	**AuthDBMType** DB|SDBM|GDBM|NDBM. En el caso de hacer uso de un gestor de bases de datos (*DBM, DataBase Management*) para la gestión de usuarios y contraseñas, *p.e.* **AuthBasicProvider dbm**, esta directiva le indicará a Apache el tipo de gestor dentro de los cuatro posibles que se pueden usar. **AuthName "Area Restringida 1"** **AuthType Basic** **AuthBasicProvider dbm** **AuthDBMType SDBM**
AuthUserFile **AuthGroupFile** **AuthDBMUserFile** **AuthDBMGroupFile**	**AuthUserFile** *ruta-Fichero-usuarios.* **AuthGroupFile** *ruta-Fichero-grupos.* ***AuthDBMUserFile*** *ruta-DB-usuarios.* **AuthDBMGroupFile** *ruta-DB-grupos.* Indican la ruta donde se localiza el fichero plano o la base de datos donde se almacenan los nombres de las cuentas de usuario Web y contraseñas que serán utilizados para validar el *login* y *password*

	introducidos por el usuario durante la autenticación. Se hará uso de uno u otro dependiendo de si el valor de la directiva **AuthBasicProvider** o **AuthDigestProvider** es *file* o *dbm*. **AuthName "Area Restringida 1"** **AuthType Basic** **AuthBasicProvider dbm** **AuthDBMType SDBM** **AuthDBMUserFile /etc/apache2/acceso/permitidos.db**
Require user **Require group**	**Require user** *lista-usuarios*. ***Require*** **group** *lista-grupos-usuarios*. Establece que cuentas de usuario de las registradas en **AuthUserFile** o **AuthDBMUserFile** serán válidas en el acceso. En el caso de que se desee indicar que todas las cuentas registradas son válidas, se usara el argumento **valid-user**. **AuthName "Area Restringida 1"** **AuthType Basic** **AuthBasicProvider dbm** **AuthDBMType SDBM** **AuthDBMUserFile /etc/apache2/acceso/permitidos.db** **Require valid-user**
Order **Allow** **Deny**	Restringen el acceso al servicio Web a determinados equipos mediante la especificación de sus direcciones IP o nombres de dominio. Las directivas **allow** y **deny** especifican respectivamente cuales son los equipos permitidos, y cuales no. **Order** indica el orden en que serán consultadas las directivas anteriores. **AuthName "Area Restringida 1"** **AuthType Basic** **AuthBasicProvider dbm** **AuthDBMType SDBM** **AuthDBMUserFile /etc/apache2/acceso/permitidos.db** **Require valid-user** **Order deny,allow** **deny from 192.168.2.0/24** **allow from all**

Para crear los ficheros planos o bases de datos que almacenarán las cuentas de usuario usadas en las areas de acceso restringido definidas en Apache, se hará uso de alguno de los siguientes comandos:

- **htpasswd**: Genera e inserta en un fichero plano los usuarios Web y sus contraseñas. Es utilizado cuando se realiza autenticación de tipo **Basic**, basado en fichero, **AuthBasicProvider file**. Su sintaxis más básica es la siguiente:

```
[root@servidor]# htpasswd [-c] ruta-fichero nombre-usuario
[root@servidor]# htpasswd -c /etc/apache2/control-acceso/lista_usuarios.dat usuario1
[root@servidor]# htpasswd /etc/apache2/control-acceso/lista_usuarios.dat usuario2
[root@servidor]# more /etc/apache2/control-acceso/lista_usuarios.dat
```

La opción "**-c**", *create*, solo debe utilizarse para crear el fichero en el momento de insertar el primer usuario Web. Para insertar el resto de usuarios debe quitarse. Tras ejecutar el comando, se solicitará la contraseña para el usuario.

- **htdigest**: Genera e inserta en un fichero plano los usuarios Web y sus contraseñas. Es utilizado cuando se realiza autenticación de tipo **Digest**, basado en fichero, **AuthDigestProvider file**. Su sintaxis más básica es la siguiente (*suponemos que la directiva AuthName tiene asignado el valor "Area1"*):

```
[root@servidor]# htdigest [-c] ruta-fichero "realm/AuthName" nombre-usuario
[root@servidor]# htdigest -c /etc/apache2/control-acceso/lista_usuarios.dg "Area1" usuario1
[root@servidor]# htdigest /etc/apache2/control-acceso/lista_usuarios.dg "Area1" usuario2
[root@servidor]# more /etc/apache2/control-acceso/lista_usuarios. dg
```

La opción "**-c**", *create*, solo debe utilizarse para crear el fichero en el momento de insertar el primer usuario Web. Para insertar el resto de usuarios debe quitarse. Importante advertir que esta utilidad requiere que se le especifique el dominio de autenticación donde será valida la cuenta a crear. Este valor se corresponde con el valor asignado a la directiva *AuthName*. Tras ejecutar el comando, se solicitará la contraseña para el usuario.

Para consultar la lista de usuarios que están registrados en un fichero, podemos hacer uso del comando more, o de un simple editor de textos. La eliminación de un usuario de la lista, implica el simple borrado de la línea correspondiente.

- **htdbm**: Genera e inserta en una base de datos (*BD*) los usuarios Web y sus contraseñas, lo que agiliza la búsqueda y validación de usuarios en la autenticación Web en relación a la utilización de un fichero plano. Puede utilizarse tanto en autenticaciones de tipo *Basic*, como *Digest*, basadas en una base de datos, *AuthBasicProvider dbm* o *AuthDigestProvider dbm*. Su sintaxis más básica es la siguiente:

```
[root@servidor]# htdbm [-c] -Ttipo BD ruta-BD nombre-usuario
[root@servidor]# htdbm -c -TSDBM /etc/apache2/control-acceso/lista_usuarios.db usuario1
[root@servidor]# htdbm -TSDBM /etc/apache2/control-acceso/lista_usuarios.db usuario2
[root@servidor]# htdbm -l -TSDBM /etc/apache2/control-acceso/lista_usuarios.db
```

La opción "**-c**", *create*, solo debe utilizarse para crear el fichero en el momento de insertar el primer usuario Web. Para insertar el resto de usuarios debe quitarse.

La opción "**-T**", *type*, es utilizada para indicar cual de los cuatro tipos de gestor de bases de datos va a utilizarse: DB|SDBM|GDBM|NDBM.

Para consultar la lista de usuarios que están registrados en una base de datos, haremos uso de la opción "**-l**", *list*, pudiendo eliminar alguno de ellos mediante la opción "**-x**".

```
[root@servidor]# htdbm -l -Ttipo BD ruta-BD
[root@servidor]# htdbm -x -Ttipo BD ruta-BD nombre-usuario
```

9.4.- Configuración de un Servicio HTTPS en Apache

Si en el apartado anterior se han repasado las directivas y comandos de configuración más comunes en Apache, en este nos centraremos en las que usaremos para configurar el servicio en modo seguro. Antes, deberemos asegurarnos de que el módulo **ssl** esta habilitado:

```
[root@servidor]# a2enmod ssl
```

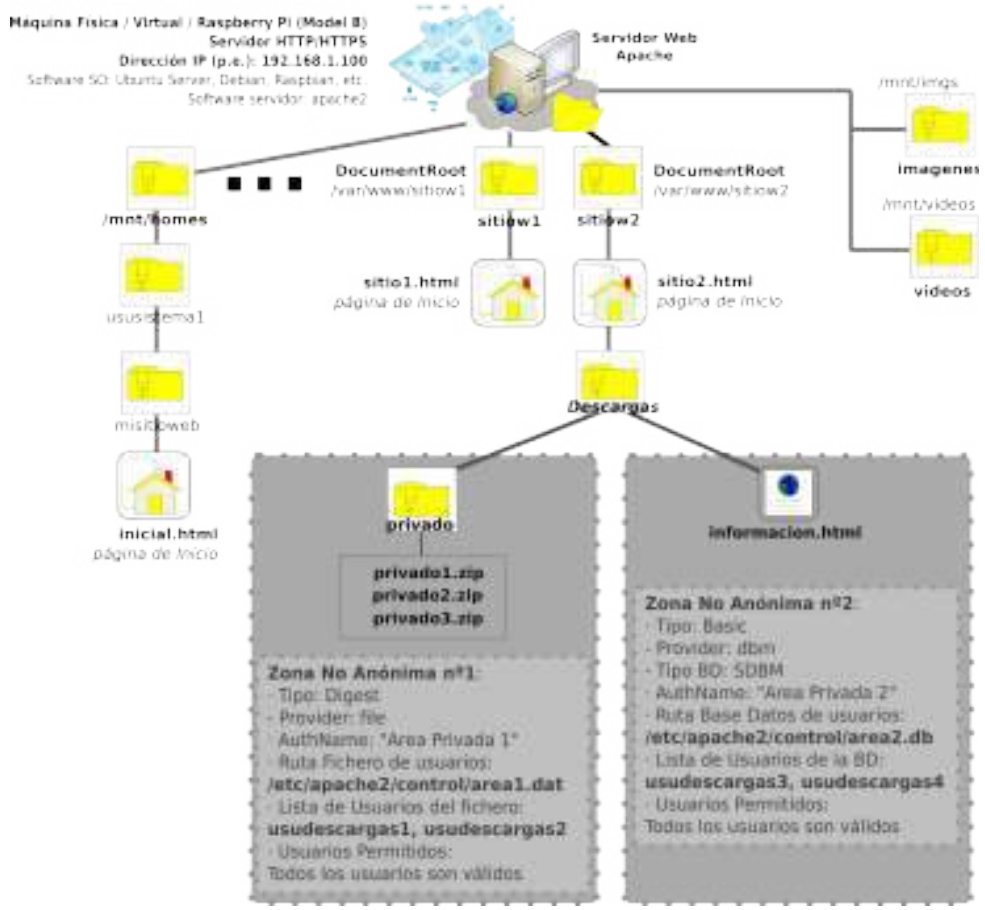
Directiva Configuración	Descripción y Ejemplos de Uso
SSLEngine	Directiva utilizada para habilitar la capa de seguridad SSL/TLS. Es necesario activarla a **on**, ya que su valor por defecto es . **SSLEngine on**
SSLCipherSuite	Con esta directiva se establecen los métodos de cifrado aceptados para dar el servicio web seguro. **SSLCipherSuite RSA:+HIGH:+MEDIUM**
SSLProtocol	Indica los tipos (*versiones*) de protocolos SSL que se permiten usar en el servicio. Estos protocolos pueden ser SSLv2, SSLv3, TLSv1, etc. Si utilizas **all**, se permitirán todos ellos. **SSLProtocol all**
SSLCertificateFile	Directiva utilizada para indicar el certificado del servidor. **SSLCertificateFile /etc/apache2/server/servercert.pem**
SSLCertificateKeyFile	Indica el fichero que contiene la clave privada del servidor. **SSLCertificateKeyFile /etc/apache2/server/serverprivkey.pem**
SSLCertificateChainFile	Directiva utilizada para indicar el archivo donde se encuentran los certificados de las autoridades de certificación (CA). Estos certificados deben estar con extensión ***.pem**.
SSLCACertificateFile	Archivo donde se encuentra el certificado de la autoridad de certificación CA que nos certifica nuestra autenticidad, con la finalidad de garantizar un correcto servicio en modo seguro. **SSLCACertificateFile /etc/apache2/CA/cacert.pem**

Ej. Práctico 9.4.1: Implementación de un Servicio HTTPS en Apache

Instala en una máquina virtual con Ubuntu Server el software Apache y ofrece un servicio HTTPS cumpliendo los requisitos que se especifican en la siguiente tabla. Respecto a la longitud de las claves a generar será de 2048 bits por cuestiones de robustez:

VirtualHost Direcciones IP Puertos	URL del Servicio Directorio Raíz del Servicio Página de Inicio	Clave CA / Cerficado CA / Días Validez Clave Sitio Web / Certificado / Días Validez Directorio CA | Directorio Certificados Sitio
192.168.1.100 443	web1.sitioseguro.es /var/www/**sitiow1** sitio1.html	CA-clave.pem / CA-cerficicado.pem|.p12 / 500 Sitio1-clave.pem / Sitio1-certificado.pem / 110 /etc/apache2/entidadca | /etc/apache2/certsitio1
192.168.1.100	web2.sitioseguro.es	CA-clave.pem / CA-cerficicado.pem|.p12 / 500

443	/var/www/**sitiow2** **sitio2.html**	**Sitio2-clave.pem / Sitio2-certificado.pem / 170** **/etc/apache2/entidadca | /etc/apache2/certsitio2**



Solución Ej. Pr. 9.4.1.I.- Cómo Implementar un Servicio HTTPS

Para dar solución al ejercicio práctico planteado seguiremos los siguientes pasos:

(1) Comenzaremos configurando el equipo servidor como autoridad de certificación, **CA**, y **creando los certificados de cada uno de los dos sitios Web a servir**. Aunque en el mundo real, la CA y el servidor HTTPS serían máquinas independientes, para la realización de la práctica haremos que sean la misma máquina para aminorar la complejidad. La finalidad de la CA será la de certificar al servidor para asegurar a los clientes Web que el servidor HTTPS al que se conectan es un servidor de confianza, es decir, para ofrecer una garantía a los clientes de que quien ofrece el servicio es realmente quien dice ser que es. Para ello seguiremos los pasos de configuración ya mostrados en los apartados anteriores relativos a la implementación de la CA y la generación de los certificados del servidor. Por comodidad, se expondrán los comandos necesarios para ello en formato script:

```
[root@servidor]# nano script-configuracion-ca-claves-certificados.sh
#!/bin/bash
```

```
clear
echo "Comprobamos si esta instalado openssl ..."
if test $(dpkg --get-selections | grep "^openssl" | wc -l) -eq 0
then
        apt-get install openssl
else
        echo "Ya esta instalado openssl. Pulsa una tecla para continuar"
        read
fi
echo "Creamos la entidad certificadora. Claves y certificado ..."
mkdir /etc/apache2/entidadca
cd /etc/apache2/entidadca
openssl genrsa -out CA-clave.pem 2048
openssl req -new -x509 -key CA-clave.pem -out CA-certificado.pem -days 500
echo "Generamos el certificado *.p12 de la CA para plataformas Windows"
openssl pkcs12 -export -out CA-certificado.p12 \
        -inkey CA-clave.pem -in CA-certificado.pem -certfile CA-certificado.pem
echo "Pasamos a generar las claves y certificados de los 2 sitios web. Pulsa Una tecla para continuar ..."
read
mkdir /etc/apache2/certsitio1
cd /etc/apache2/certsitio1
openssl genrsa -out Sitio1-clave.pem 2048
echo "Creamos la solicitud del certificado del sitio1..."
openssl req -new -key Sitio1-clave.pem -out Sitio1-certificado.p10
echo "Ahora la CA firma la solicitud del sitio1..."
openssl x509 -req -in Sitio1-certificado.p10 \
        -out Sitio1-certificado.pem \
        -CA ../entidadca/CA-certificado.pem -CAkey ../entidadca/CA-clave.pem \
        -CAcreateserial -days 110
mkdir /etc/apache2/certsitio2
cd /etc/apache2/certsitio2
openssl genrsa -out Sitio2-clave.pem 2048
echo "Creamos la solicitud del certificado del sitio2 …"
openssl req -new -key Sitio2-clave.pem -out Sitio2-certificado.p10
echo "Ahora la CA firma la solicitud del sitio2 ..."
openssl x509 -req -in Sitio2-certificado.p10 \
        -out Sitio2-certificado.pem \
        -CA ../entidadca/CA-certificado.pem -CAkey ../entidadca/CA-clave.pem \
        -CAcreateserial -days 170
```

¡¡Importante!! Resaltar de nuevo que entre los datos introducidos al crear las solicitudes de certificado (**.p10*) del servidor Web para los dos sitios web, el dato más importante es el "**Common Name**", el cual debe corresponderse con el nombre de dominio cualificado (**FQDN**) asociado a cada uno de los sitios web: **web1.sitioseguro.es** y **web2.sitioseguro.es**. Por ejemplo, para la solicitud de certificado del primer sitio Web:

```
[root@servidor]# openssl req -new -key Sitio1-clave.pem -out Sitio1-certificado.p10
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Zaragoza
Locality Name (eg, city) []:Zaragoza
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Web1 Segura Profesor
Organizational Unit Name (eg, section) []:Depart. Ventas e Informatica
Common Name (e.g. server FQDN or YOUR name) []:web1.sitioseguro.es
Email Address []:web1sitioseguro@gmail.com
```

Al crear la solicitud también se nos solicitarán los datos "**A challenge password []:** " y "**An optional company name []:**", los cuales pueden dejarse vacíos.

```
[root@servidor]# chmod +x script-configuracion-ca-claves-certificados.sh
[root@servidor]# ./script-configuracion-ca-claves-certificados.sh
```

(2) Tras crear los certificados necesarios para ofrecer el sitio Web de manera segura, pasaremos a crear la estructura del sitio Web que se muestra en la figura anterior:

```
[root@servidor]# mkdir -p /mnt/homes/ususistema1/misitioweb /mnt/videos \
/mnt/imgs /var/www/sitiow1 /var/www/sitiow2/Descargas/privado
[root@servidor]# touch /mnt/videos/video1.avi /mnt/videos/video2.avi …
[root@servidor]# touch /mnt/imgs/imagen1.png /mnt/imgs/imagen2.png …
[root@servidor]# touch /var/www/sitiow2/Descargas/privado/privado1.zip ...
[root@servidor]# useradd -d /mnt/homes/ususistema1 -s /bin/bash ususistema1
[root@servidor]# passwd ususistema1
[root@servidor]# chown -R ususistema1 /mnt/homes/ususistema1
[root@servidor]# su ususistema1
[ususistema1@servidor]$ cd
[ususistema1@servidor]$ nano misitioweb/inicial.html
<br><hr><h3 style='text-align: center; color: blue'>Sitio Web Ususistema1</h3><hr><br>
<p>Este ...</p>
```

```
[ususistema1@servidor]$ exit
[root@servidor]# nano /var/www/sitiow1/sitio1.html
<br><hr><h3 style='text-align: center; color: blue'>Sitio Web 1</h3><hr><br>
<p>Este es el primer sitio Web en modo seguro...</p>
```

```
[root@servidor]# nano /var/www/sitiow2/sitio2.html
<br><hr><h3 style='text-align: center; color: blue'>Sitio Web 2</h3><hr><br>
Para acceder a datos confidenciales: <a href='Descargas/informacion.html'>Confidencial</a>
<br><br>
Para descargarte documentos privados:
<br> 1) <a href='Descargas/privado/privado1.zip'>Doc Privado 1</a>
<br> 2) <a href='Descargas/privado/privado2.zip'>Doc Privado 2</a>
<br> ...
```

```
[root@servidor]# nano /var/www/sitiow2/Descargas/informacion.html
```

```
<br><hr><h3 style='text-align: center; color: blue'>Datos Confidenciales</h3><hr><br>
<p>Esto son datos confidenciales del segundo sitio Web...</p>
<hr><a href='../sitio2.html'>Volver al Inicio</a>
```

(3) A continuación configuraremos Apache. Para ello (a) crearemos un archivo de configuración (*p.e. sitios-seguros.conf*) para los dos nuevos sitios Web a servir, (b) habilitaremos los módulos necesarios (*a2enmod*) relacionados con las directivas que se han utilizado en la configuración y que no vienen cargados por defecto, (c) crearemos los archivos necesarios para la autenticación de las áreas de acceso restringido (*htdigest*, *htdbm*), (d) habilitaremos la configuración realizada (*a2ensite*) y (e) por último reiniciaremos el servicio HTTP/HTTPS ofrecido por Apache:

```
[root@servidor]# cd /etc/apache2
[root@servidor:/etc/apache2]# nano sites-available/sitios-seguros.conf
<VirtualHost 192.168.1.100:443>
        ServerName "web1.sitioseguro.es"
        DocumentRoot /var/www/sitiow1
        DirectoryIndex sitio1.html inicial.html
        # Definimos Alias para el acceso a directorios fuera del sitio Web
        Alias /imagenes /mnt/imgs
        Alias /videos /mnt/videos
        # Permitimos la publicación de contenidos Web desde el Home de los usuarios
        UserDir /mnt/homes/*/misitioweb
        UserDir enabled
        # Asignamos valor a las directivas relacionadas con la capa de seguridad SSL
        SSLEngine on
        SSLCipherSuite RSA:+HIGH:+MEDIUM
        SSLProtocol all
        SSLCertificateFile /etc/apache2/certsitio1/Sitio1-certificado.pem
        SSLCertificateKeyFile /etc/apache2/certsitio1/Sitio1-clave.pem
        SSLCACertificateFile /etc/apache2/entidadca/CA-certificado.pem
        <Directory /var/www/sitiow1>
                Allow from all
        </Directory>
</VirtualHost>

<VirtualHost 192.168.1.100:443>
        ServerName "web2.sitioseguro.es"
        DocumentRoot /var/www/sitiow2
        DirectoryIndex sitio2.html inicial.html
        # Definimos Alias para el acceso a directorios fuera del sitio Web
        Alias /imagenes /mnt/imgs
        Alias /videos /mnt/videos
        # Permitimos la publicación de contenidos Web desde el Home de los usuarios
        UserDir /mnt/homes/*/misitioweb
        UserDir enabled
        # Asignamos valor a las directivas relacionadas con la capa de seguridad SSL
        SSLEngine on
```

```
        SSLCipherSuite RSA:+HIGH:+MEDIUM
        SSLProtocol all
        SSLCertificateFile /etc/apache2/certsitio2/Sitio2-certificado.pem
        SSLCertificateKeyFile /etc/apache2/certsitio2/Sitio2-clave.pem
        SSLCACertificateFile /etc/apache2/entidadca/CA-certificado.pem
        <Directory /var/www/sitiow2>
                Allow from all
        </Directory>
        # Definimos las áreas de acceso restringido del sitio Web
        <Directory /var/www/sitiow2/Descargas/privado>
                AuthType Digest
                AuthName "Area Privada 1"
                AuthDigestProvider file
                AuthUserFile /etc/apache2/control/area1.dat
                Require valid-user
        </Directory>
        <Directory /var/www/sitiow2/Descargas>
                <Files informacion.html>
                        AuthType Basic
                        AuthName "Area Privada 2"
                        AuthBasicProvider dbm
                        AuthDBMType SDBM
                        AuthDBMUserFile /etc/apache2/control/area2.db
                        Require valid-user
                </Files>
        </Directory>
</VirtualHost>
```

Ahora habilitaremos los módulos **ssl** (*nos permite añadir la capa de seguridad al protocolo HTTP – HTTPS*), **userdir** (*nos permite que los usuarios puedan publicar contenidos desde sus HOMES*), **auth_digest** (*nos permite autenticar usuarios Web de manera Digest*), **authn_dbm** (*nos permite autenticar usuarios Web de manera Basic mediante el uso de una base de datos de Apache*):

```
[root@servidor:/etc/apache2]# a2enmod ssl userdir auth_digest authn_dbm
```

Crearemos los archivos relacionados con la autenticación:

```
[root@servidor:/etc/apache2]# mkdir control
[root@servidor:/etc/apache2]# htdigest -c control/area1.dat "Area Privada 1" usudescargas1
[root@servidor:/etc/apache2]# htdigest control/area1.dat "Area Privada 1" usudescargas2
[root@servidor:/etc/apache2]# htdbm -c -TSDBM control/area2.db usudescargas3
[root@servidor:/etc/apache2]# htdbm -TSDBM control/area2.db usudescargas4
```

Además, antes de reiniciar el servicio, deberemos editar el archivo **ports.conf** y añadir la directiva "**NameVirtualHost 192.168.1.100:443**" (*suponiendo que la IP del servidor Web es 192.168.1.100*), la cual informará a Apache de que hay más de un sitio Web a servir

(*web1.sitioseguro.es y web2.sitioseguro.es*) que hacen uso de la misma IP (*192.168.1.100*) y el mismo puerto (*443*), y que por tanto tiene que fijarse en la directiva **ServerName** para diferenciarlos ante una solicitud de conexión HTTPS:

```
[root@servidor:/etc/apache2]# nano ports.conf
NameVirtualHost 192.168.1.100:443
...
```

Y por último habilitaremos el archivo anterior de configuración y reiniciaremos Apache:

```
[root@servidor:/etc/apache2]# a2ensite sitios-seguros.conf
[root@servidor:/etc/apache2]# /etc/init.d/apache2 restart
```

(4) Ahora tan sólo falta comprobar el correcto funcionamiento del servicio HTTPS ofrecido por Apache desde un cliente Web. Para ello, en primer lugar, será necesario que el cliente pueda resolver los nombres de dominio FQDN asignados a los sitios Web: **web1.sitioseguro.es** y **web2.sitioseguro.es**. Una opción sería configurar un servidor DNS (p.e. bind9), dar de alta la zona **sitioseguro.es** y registrar en su archivo de zona asociada que los nombres web1 y web2 se corresponden con la IP del equipo servidor configurado (*registro A*, *Address*). Otra opción mucho más sencilla sería editar el archivo hosts, disponible tanto en GNU/Linux como en Windows, y añadir la resolución de los dos nombres de dominio utilizados:

```
[root@clienteweb]# nano /etc/hosts
# Suponiendo que la dirección IP del servidor HTTPS es 192.168.1.100
192.168.1.100        web1.sitioseguro.es
192.168.1.100        web2.sitioseguro.es
...
```

¡¡Importante!! En una situación real (*acceso desde la Internet*) la resolución de los nombres de dominio anterior debería corresponderse con la dirección IP pública del router suministrado por el proveedor de servicios de Internet que se tiene contratado (*p.e. Telefónica, ONO, Orange, etc.*) y que nos da acceso a la Intranet donde se encuentra el servidor Web HTTPS, dando por hecho que previamente se ha configurado en el router el redireccionamiento del puerto 443 hacía dicho equipo servidor.

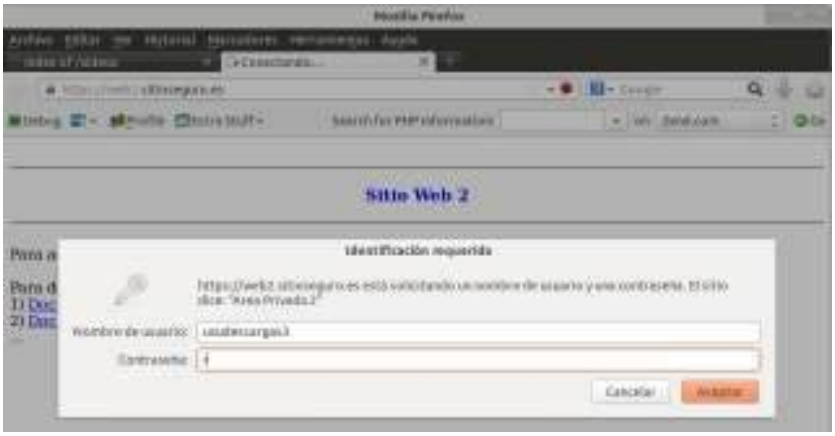
Para comprobar la resolución y conectividad con el servidor podríamos ejecutar un simple ping:

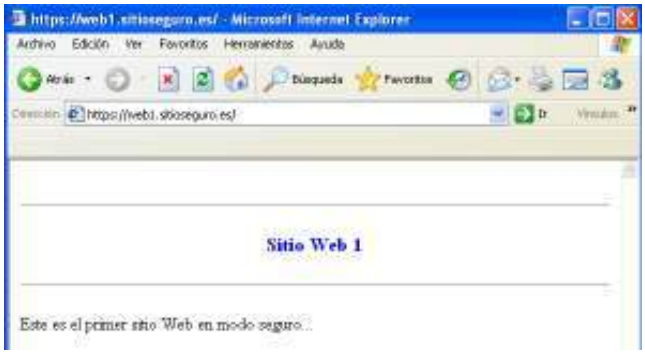
```
[root@clienteweb]# ping web1.sitioseguro.es
```

(5) Para comprobar el acceso escribiríamos en la barra de direcciones de nuestro navegador preferido la URL: https://web1.sitioseguro.es o https://web2.sisitioseguro.es. Al acceder deberemos observar que nuestro navegador nos advierte que el sitio Web al que nos conectamos es no confiable. Esto es porque no hay ninguna CA configurada en el navegador que nos pueda certificar que el sitio Web al que nos conectamos lo esta sirviendo quien dice ser que es.

(6) Para comprobar la funcionalidad de la CA creada anteriormente, añadiremos el certificado de la CA **CA-certificado.pem** o **CA-certificado.p12** (*Windows*) a nuestro navegador. Para ello transferiremos el certificado del servidor al cliente, y desde la preferencias del navegador lo añadiremos.

(7) Ahora se reiniciamos nuestro navegador y volvemos a conectarnos vía https podremos comprobar que el sitio ya es de confianza (*lo avala nuestra CA*) y que por tanto el navegador no nos muestra ningún mensaje de advertencia. De igual forma deberíamos comprobar las zonas de acceso restringido.





(8) Otras comprobaciones que podrían hacerse:

- Acceso a los Alias: https://web1.sitioseguro.es/videos https://web2.sitioseguro.es/imagenes
- Acceso a los sitios Web servidos desde el HOME de los usuarios del sistema: https://web1.sitioseguro.es/~ususistema1
- Acceso restringido a las áreas "Area Privada 1" y "Area Privada 2". Comprobar que únicamente puden acceder y descargarse los archivos los usuarios indicados en el archivos de autenticación.

9.5.- Transferencia de Archivos en Modo Seguro: FTPS y SFTP

Al igual que HTTP, muchos de los protocolos existentes hoy en día disponen de una versión en modo seguro (*FTPS, SFTP, SMTPS, IMAPS, etc.*) con la finalidad de asegurar una comunicación confidencial, poder comprobar la autenticidad de quien ofrece el servicio y garantizar la integridad de los datos que se transfieren. En concreto, estos protocolos suelen hacer uso de una capa de seguridad SSL/TLS tal como se ha configurado en Apache para ofrecer el servicio HTTPS, o hacen uso de un túnel SSH en modo seguro (*p.e. SFTP*).

Por ejemplo, si instalamos el software servidor ProFTPd en un equipo (*apt-get install proftpd*), este se convertirá sin hacer ningún tipo de configuración añadida en un servidor FTP y SFTP (*SSH File Transfer Protocol*). Podríamos comprobar su correcto funcionamiento desde un cliente SFTP como es FileZilla.

Práctica Nº10.- Redes Privadas Virtuales: VPNs

Gracias a los grandes avances tecnológicos que existen actualmente cada vez son más las empresas que ofrecen a algunos de sus trabajadores la posibilidad trabajar remotamente desde su casa, desde un hotel, o desde cualquier otro lugar donde exista una conexión a Internet. Es lo que se conoce como teletrabajo. Para poder ofrecer esta posibilidad las empresas deben implementar una red privada virtual o también conocida por sus siglas como VPN. Esta VPN, además de permitir el teletrabajo dentro de una empresa, va a conllevar un conjunto de ventajas incuestionables que podrán apreciarse una vez implementada. A modo de ejemplo, la implementación de una VPN nos permitiría como clientes de la VPN, entre otras muchas cosas, las siguientes:

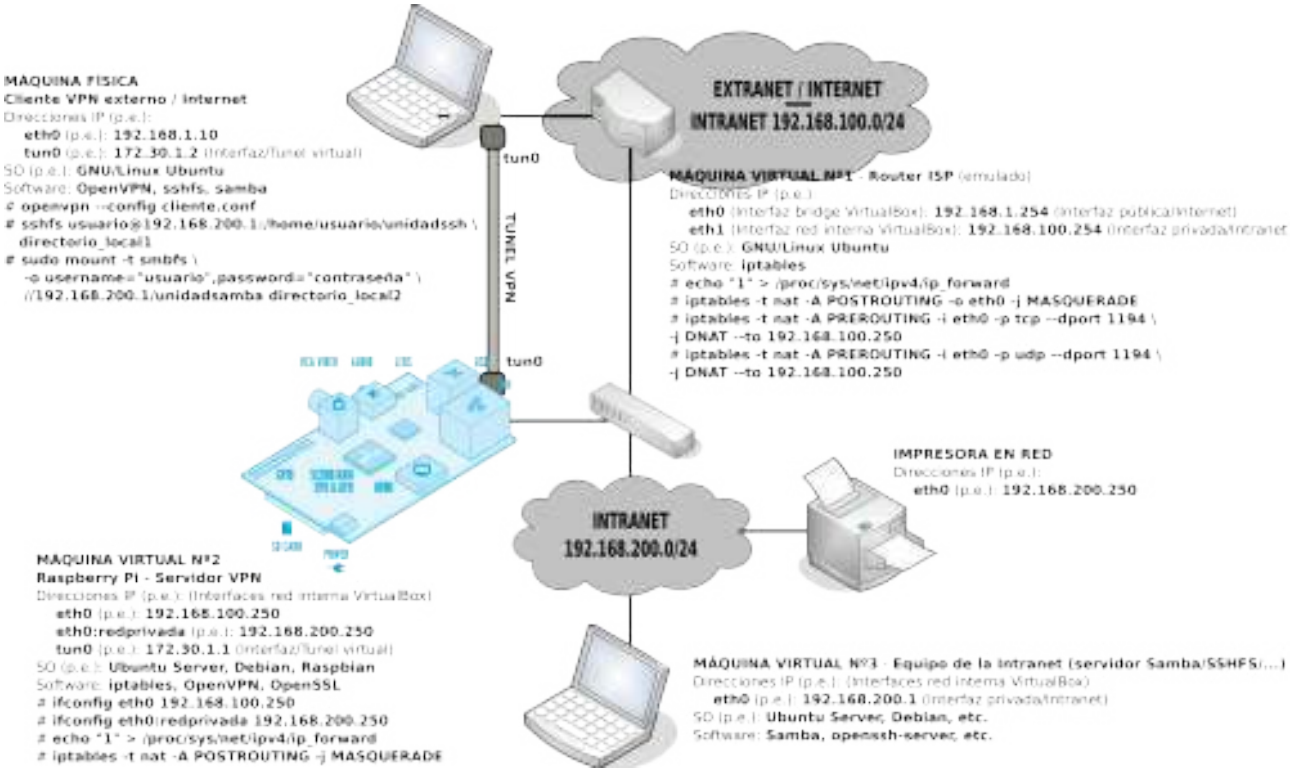
- Acceder remotamente a los servicios que se ofrecen dentro de una Intranet privada (*p.e. acceso a directorios compartidos*) mediante el uso de las direcciones IP privadas, dándonos la impresión de que nos encontramos dentro de la propia Intranet.
- Mandar remotamente trabajos de impresión a cualquier impresora en red o compartida que haya en una Intranet accesible a través de la VPN.
- Configurar desde cualquier lugar de Internet el Router ADSL de nuestro proveedor de servicios de Internet (ISP) que se localiza dentro de la Intranet a la que nos da acceso la VPN haciendo referencia a la dirección IP privada que tenga asignada éste dentro de la Intranet donde se encuentra (*p.e. 192.168.1.1*).
- … En definitiva, la VPN nos hace creer que nos encontramos en la Intranet remota siendo accesible para el cliente VPN cualquier equipo o servicio que se localice en ella.

Además, en empresas compuestas por varias sucursales, la implementación de una VPN puede ser una gran solución a muchos problemas, ya que garantiza que las Intranet de las distintas sucursales se mantengan conectadas como si estuvieran todas ellas dentro de la misma red de área local (LAN).

10.1.- Introducción a la Implementación Práctica de una VPN

Tal como se ha indicado antes, mediante la implementación de una VPN se pretende hacer creer que al usuario que tiene un acceso directo a la red lógica de la Intranet remota con la que normalmente trabaja. Para ello, tras el establecimiento de un túnel en modo seguro, el servidor VPN hará de gateway entre el equipo cliente y la Intranet a la que desea conectarse.

En la presente práctica se mostrará como configurar una conexión VPN (*Red Privada Virtual*) entre un servidor VPN de una Intranet (*p.e. una Raspberry, una máquina virtual, etc.*) y un cliente VPN de la Internet. Esto permitirá al cliente externo acceder de manera segura a cualquier recurso de la Intranet, siempre y cuando no haya algún firewall que lo evite.



La implementación de la VPN evitará tener que configurar el redireccionamiento másivo de puertos (NAT PREROUTING, *redireccionamiento de puertos*) en el router suministrado por nuestro proveedor de servicios de Internet (*ISP*) con la finalidad de alcanzar servicios ofrecidos en la Intranet.

La red privada virtual, se denomina así, ya que hará creer tanto al servidor VPN, como a los clientes VPN, que se encuentran en una nueva Intranet con su propio rango de direcciones IP privadas (*p.e. 172.30.1.0/24*), como si estuvieran conectados directamente entre sí a través de un dispositivo de interconexión (*hub, switch, punto de acceso, etc.*), cuando en realidad, la conexión entre todos ellos se realiza a través de la Internet con multitud de dispositivos de enrutamiento entre ellos.

Aunque existen otras alternativas, para la implementación de la VPN haremos uso del software **OpenVPN**, necesaria su instalación tanto en los clientes como en el servidor. Este software nos creará un nuevo dispositivo de red virtual (*tun/tap*) en el equipo el cuál recibirá una dirección IP dentro de la red privada reservada para ello, y que nos permitirá que se puedan interconectar entre sí los equipos que la forman. En concreto, dependiendo del ámbito de la VPN, la interfaz de red virtual deberá ser "**tun**" (*interfaz network tunel*) en el caso de necesitar manejar paquetes con cabeceras del nivel 3 o de la capa red del modelo TCP/IP, o "**tap**" (*interfaz network tap*) si tan sólo necesitamos manejar paquetes de nivel 2 o de capa de enlace. Según lo anterior, si la VPN va a establecerse a través de Internet o una Intranet con direccionamiento IP haremos uso de un dispositivo "**tun**", en cambio, si la VPN va a establecerse sin necesidad de direccionamiento IP, en la configuración del servicio escogeremos "**tap**".

En concreto, para la práctica que aquí se plantea, se ha escogido la dirección de red/subred privada "**172.30.1.0/24**", y dispositivos de red de tipo "**tun**". A la hora de escoger dicha dirección de red en la práctica, debemos tener cuidado con no entrar en conflicto con ninguna otra dirección de red privada o Intranet correspondientes a las redes privadas asociadas a los propios equipos que forman parte de la VPN. Por ejemplo, si observamos el esquema de la figura que se presenta al principio, las direcciones de red 192.168.1.0/24, 192.168.100.0/24 y 192.168.200.0/24 están siendo utilizadas para direccionar los equipos que forman las distintas redes privadas reales del esquema. Por ello, si queremos evitar conflictos con la VPN, la red privada virtual que deseamos crear no debe volver a usar ninguno de los rangos anteriores.

Respecto a la dirección de red o subred a utilizar para la VPN advertir que la IANA (*Internet Assigned Numbers Authriy*) tiene reservados varios rangos de direcciones de red para asignarlos a redes privadas (*el resto de redes de clase A, B o C son de asignación pública, con validez en Internet*). En concreto son las siguientes:

Clase y Máscara de red	Rango de direcciones de Red disponibles para asignación privada
A (255.0.0.0)	10.0.0.0 – 10.255.255.255
B (255.255.0.0)	172.16.0.0 – 172.31.255.255
C (255.255.255.0)	192.168.0.0 – 192.168.255.255

¡¡Muy Importante!! El aspecto más importante a tener en cuenta para la exitosa implementación de una VPN, es asegurarnos que las direcciones de red lógicas asociadas a las redes internas o Intranet que se interconectan mediante la VPN no se repitan para evitar conflictos. Es decir, una situación de conflicto podría ser la siguiente: un equipo cliente que se encuentra en su casa con una dirección IP asignada 192.168.1.22 (*dirección de red lógica de la Intranet de casa 192.168.1.0/24*) quiere establecer una conexión VPN con un servidor para acceder a la Intranet de la empresa cuya dirección de red lógica es también la 192.168.1.0/24. Una vez establecida la VPN el equipo cliente no sabría como enrutar sus paquetes, ya que en su tabla de enrutamiento le aparecerían dos reglas diferentes para alcanzar la misma red de destino (192.168.1.0/24).
Para evitar el conflicto anterior, muy común a consecuencia de que la mayoría de los router ISP vienen preconfigurados como servidores DHCP dentro de la red 192.168.1.0/24, el administrador de la red de la empresa debería asignar una dirección de red de uso no muy habitual (*p.e. 192.168.101.0/24, 172.30.101.0/24, etc.*).

10.2.- Configuración del Router ISP

Antes de empezar con la práctica deberemos configurar el router ISP (*Proveedor de Servicios de Internet*) o equipo que haga de puerta de acceso hacia el exterior. Es decir, desde el exterior, Internet, el único equipo accesible de nuestra red, es el router ISP, ya que es el único que tiene una dirección IP pública univoca y reconocida desde Internet. Por ello, cuando queramos hacer una solicitud de conexión al servicio VPN, se la tendremos que hacer a él, y este programarlo para que la redireccione (*NAT Prerouting*) hacía el equipo servidor VPN de la Intranet que ofrece ese servicio. Es decir, debemos decirle al router ISP que cuando reciba una solicitud de conexión vía protocolo **TCP/1194** o **UDP/1194** (*el 1194 es el puerto por defecto del servicio VPN*) se redireccione dicha solicitud a la dirección IP servidor VPN interno.

Ej. Práctico 10.2.1: Configuración del Router ISP para una conexión VPN

Tal como se puede observar en la figura anterior, para la implementación práctica de la VPN haremos uso de 4 equipos:

(1) La máquina física hará las veces de **cliente VPN**. Para esta máquina (*p.e. 192.168.1.10*) tan sólo será accesible la máquina virtual Nº1 cuya dirección IP pertenece a la misma red lógica (*p.e. 192.168.1.254*).
(2) Una primera máquina virtual hará las veces de **router ISP**. Esta máquina se encargará de hacer de intermediaría (*gateway*) entre la red lógica 192.168.1.0/24 (*emula la red pública de Internet*) y las Intranet privadas (*192.168.100.0/24 y 192.168.200.0/24*). En concreto hará de gateway hacia el exterior para los equipos de la Intranet y permitirá el acceso desde el exterior a los servicios que se ofrecen en la Intranet (*servicio VPN*) mediante el correspondiente redireccionamiento de puertos (*NAT Prerouting tcp/upd 1194*).
(3) Una segunda máquina virtual hará las veces de **servidor VPN** y de **gateway interno**, al encargarse de separar la Intranet privada en dos redes lógicas separadas (*192.168.100.0/24 y 192.168.200.0/24*). Por tanto, en relación a la VPN, esta máquina hará de intermediaria o gateway entre los clientes VPN y los equipos de la Intranet a los que se les dará acceso. Para evitar una mayor complejidad, está máquina dispondrá de una única interfaz de red con dos direcciones IP asignadas, una de cada Intranet, **192.168.100.250** y **192.168.200.250**, por lo que la separación física entre las redes privadas no existirá, y por tanto, tan sólo será una separación lógica.
(4) Una tercera máquina virtual hará de **servidor de la Intranet**. Esta máquina se encargará de ofrecer servicios de red (*p.e. servicios de transferencia de archivos Samba, SSHFS, etc.*) a los equipos de la Intranet, permitiéndonos comprobar desde el cliente VPN que podemos acceder a ellos una vez establecida la VPN sin ningún problema, como si tratase de un equipo más de la Intranet.

En este primer ejercicio práctico nos limitaremos a configurar la primera máquina virtual encargada de emular al router ISP.

Solución Ej. Pr. 10.2.1.I.- Cómo Configurar el Redireccionamiento en el Router

En un supuesto caso real lo que haríamos sería acceder desde mediante un navegador Web

de un equipo cliente de la Intranet al router ISP para configurar el redireccionamiento de los puertos necesarios hacia el servidor VPN (*p.e. NAT Prerouting tcp/upd/1194*).

En nuestro caso, al emular el router ISP mediante una máquina virtual (*p.e. Ubuntu Server*), tan sólo tendremos que ejecutar los comandos **iptables** necesarios (*para saber más, echar un ojo a la práctica de firewalls*):

```
[root@routerisp]# echo "1" > /proc/sys/net/ipv4/ip_forward
[root@routerisp]# iptables -t nat -A POSTROUTING -j MASQUERADE
[root@routerisp]# iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 1194 -j DNAT \
                        --to 192.168.100.250
[root@routerisp]# iptables -t nat -A PREROUTING -i eth0 -p udp --dport 1194 -j DNAT \
                        --to 192.168.100.250
```

¡¡Observación!! La emulación del router ISP que hacemos en la práctica se aleja muy poco de la realidad, ya que prácticamente la totalidad de todos los router comerciales que existen tienen preinstalado un sistema operativo GNU/Linux adaptado a las necesidades especificas de un router.

Si tenemos en cuenta que la configuración anterior se pierde al reiniciar el equipo, para no tener que ejecutar los anteriores comandos cada vez que iniciamos sesión, lo más idóneo sería crear un script que se ejecute al iniciarse el equipo, o incluirlas al final del script "**/etc/init.d/rc.local**":

```
[root@routerisp]# nano /etc/init.d/rc.local
## Líneas añadidas al final de rc.local:
…
# Si quisiéramos configurar la red por comandos al iniciarse la máquina virtual:
/etc/init.d/networking stop
ifconfig eth0 down
ifconfig eth1 down
ifconfig eth0 192.168.1.254
ifconfig eth1 192.168.100.254
route add default gw 192.168.1.X
echo "nameserver 8.8.8.8" > /etc/resolv.conf
# Configuramos el equipo como gateway y redireccionamos el puerto tcp/udp/1194
echo "1" > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -j MASQUERADE
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 1194 -j DNAT --to 192.168.100.250
iptables -t nat -A PREROUTING -i eth0 -p udp --dport 1194 -j DNAT --to 192.168.100.250
```

10.3.- Instalación del Software OpenVPN

OpenVPN será el software que tendremos que instalar **tanto en el servidor como en los equipos clientes** para poder establecer la red privada virtual:

```
[root@servidorvpn|clientesvpn]# apt-get install openvpn
```

10.4.- Instalación de OpenSSL. Creación de la Autoridad de Certificación (CA), y las Claves y Certificados del Servidor y de los Clientes.

Al igual que vimos en la práctica del servicio HTTPS *(repasar lo visto en dicha práctica)*, para garantizar una comunicación segura será necesario disponer de una autoridad certificadora (CA) para que suministre los correspondientes certificados a los equipos servidor y clientes. Para todo ello necesitaremos tener instalado "**openssl**":

```
[root@servidorvpn]# apt-get install openssl
```

Ej. Práctico 10.4.1: Creación de Certificados VPN para Servidor y Clientes

Antes de configurar el servicio VPN será necesario crear las claves y certificados necesarios que permitan garantizar la **autenticidad** y **confidencialidad** en la comunicación entre los clientes y el servidor (*repasar la práctica asociada a sistemas criptográficos*).

En concreto, en este ejercicio práctico crearemos mediante el software **openssl** los siguientes archivos:

(1) Claves asimétricas y certificado autofirmado de la CA. Estos son los únicos archivos necesarios para crear una entidad certificadora.
Por cuestiones de sencillez la CA no se corresponderá con un equipo que haga expresamente de entidad certificadora, sino que la implementaremos en el equipo servidor VPN.
(2) Claves y certificados del servidor VPN. Permitirán garantizar al cliente que el servidor VPN al que se conectan es realmente quien dice ser que es, además de asegurar una comunicación confidencial. El certificado se encargará de firmarlo la CA creada.
(3) Claves y certificados del cliente VPN. Permitirán garantizar al servidor que el cliente VPN que hace la solicitud de conexión es realmente quien dice ser que es, además de asegurar una comunicación confidencial. El certificado se encargará de firmarlo la CA creada.

Solución Ej. Pr. 10.4.1.I.- Cómo crear las Claves y Certificados del Servicio VPN

Para automatizar la creación de la CA, claves y certificados, podemos copiar las líneas siguientes en un archivo y ejecutarlo como si se tratase de un script:

```
[root@servidorvpn]# nano script-vpn.sh
#!/bin/bash
# Comenzaremos creando la CA (claves y certificado)
echo "Creamos la CA. En primer lugar su clave, y luego el certificado con extensión crt"
read -p "Pulsa Intro para continuar ..."
mkdir /etc/openvpn/ca
cd /etc/openvpn/ca
openssl genrsa -out ca.key 2048
openssl req -new -x509 -key ca.key -out cacert.crt -days 1000

# Generamos las claves y certificados para el servidor VPN con extensión crt, en lugar de pem
```

```
echo "Creamos las claves y certificado para el servidor"
read -p "Pulsa Intro para continuar ..."
mkdir /etc/openvpn/server
cd /etc/openvpn/server
openssl genrsa -out server.key 2048
openssl req -new -key server.key -out servercert.p10
openssl x509 -req -in servercert.p10 -out servercert.crt -CA ../ca/cacert.crt \
        -CAkey ../ca/ca.key -CAcreateserial -days 365

# Generamos la Diffie-Hellman key exchange
echo "Generamos la Diffie-Hellman key..."
read -p "Pulsa Intro para continuar ..."
cd /etc/openvpn/ca
openssl dhparam -check -text -5 1024 -out dh1024.pem

# Generamos las claves y  certificados para el cliente1 con extensión crt
echo "Creamos las claves y certificado para el cliente1"
read -p "Pulsa Intro para continuar ..."
mkdir /etc/openvpn/cliente1
cd /etc/openvpn/cliente1
openssl genrsa -out cliente1.key 2048
openssl req -new -key cliente1.key -out cliente1cert.p10
openssl x509 -req -in cliente1cert.p10 -out cliente1cert.crt -CA ../ca/cacert.crt \
        -CAkey ../ca/ca.key -CAcreateserial -days 365
```

Ahora tan sólo tendremos que dar permiso de ejecución al script anterior, y ejecutarlo:

```
[root@servidorvpn]# chmod +x script-vpn.sh
[root@servidorvpn]# ./script-vpn.sh
```

10.5.- Configuración del Servidor VPN

En este paso configuraremos el equipo servidor encargado de atender las solicitudes de conexión realizas por los clientes remotos de la VPN. Para ello editaremos el archivo de configuración "**/etc/openvpn/server.conf**". En concreto, estableceremos la siguiente configuración, congruente con el esquema de red mostrado al comienzo:

- Se asignará a la VPN la dirección de subred "**172.30.1.0/24**". El servidor VPN se asignará a si mismo de manera automática la dirección IP más baja, "**172.30.1.1**", y el resto de las direcciones IP de la subred las irá asignando a los clientes de la VPN a medida que vaya recibiendo solicitudes de conexión.

```
server 172.30.1.0 255.255.255.0
```

- Habilitaremos un dispositivo de red virtual de tipo "**tun**" (*tunel*), "**tun0**", al cual asignaremos la dirección IP privada anterior, "**172.30.1.1**". Mediante esta interfaz el servidor se comunicará con la interfaz "**tun**" de los clientes.

```
dev tun
persist-tun
```

- Indicaremos al servidor VPN que tras aceptar la solicitud de conexión de un cliente VPN, altere la tabla de enrutamiento del equipo cliente, para que sepa que redes internas o Intranet pueden ser alcanzadas desde los clientes externos VPN a través del servidor VPN haciendo este de gateway o puerta de enlace. Según el esquema planteado al comienzo de la práctica, el servidor VPN hará de gateway para alcanzar cualquiera de los equipos que forman parte de las red privadas a las que tiene acceso "**192.168.100.0/24**" y "**192.168.200.0/24**".

```
push "route 192.168.100.0 255.255.255.0"
push "route 192.168.200.0 255.255.255.0"
```

- El servicio VPN puede ofrecerse a través de los protocolos de la capa de transporte "**tcp**" o "**udp**". Para este caso práctico se ha seleccionado "**udp**" (*no orientado a conexión*), aunque puede elegirse "**tcp**", y advertir cual podría dar un mejor servicio (*dependerá del tráfico de red, de la cantidad de servicios que ya se estén dando en el servidor, etc.*). Además se ha respetado el puerto por defecto del servicio VPN "**1194**", aunque puede igualmente modificarse en el caso en que queramos dar servicio a múltiples VPNs simultáneamente.

```
port 1194
proto udp
topology subnet
```

Ej. Práctico 10.5.1: Configuración del Servidor VPN

Siguiendo con la práctica que se plantea en el presente capítulo, tras haber configurado el router ISP, haber creado la CA y haber creado las claves y certificados, en este ejercicio práctico configuraremos el servidor VPN con las características ya comentadas.

Solución Ej. Pr. 10.5.1.I.- *Cómo Configurar un Servidor VPN*

A continuación se muestra una configuración ya probada previamente de manera exitosa en un servidor VPN, con la explicación de las directivas utilizadas:

```
[root@servidorvpn]# apt-get install openvpn
[root@servidorvpn]# nano /etc/openvpn/server.conf
# Contenido de "/etc/openvpn/server.conf"
## Comenzaremos indicando el puerto (por defecto, 1194), el protocolo (tcp/udp) y la topología
port 1194
proto udp
topology subnet

## Indicamos la ubicación de los certificados y claves creados anteriormente:
## Clave asimétrica del servidor, certificados de la CA y del servidor, y la clave Diffie-Hellman
```

ca /etc/openvpn/ca/**cacert.crt**
cert /etc/openvpn/server/**servercert.crt**
key /etc/openvpn/server/**server.key**
dh /etc/openvpn/ca/**dh1024.pem**
En el caso de desear que todos los clientes puedan usar el mismo certificado,
descomentaremos la siguente línea:
;duplicate-cn

Definimos las características del tunel seguro que se va a generar:
Tipo de dispositivo que creará el tunel (tun/tap), y características de la conexión
dev tun
persist-tun
persist-key
keepalive 10 120
comp-lzo

Especificamos el rango de direcciones que se asignarán a los equipos de la VPN
Debe tratarse de una dirección de red o subred no utilizada por ninguna de las Intranet
en la que se encuentran los equipos servidor o clientes.
En caso contrario, habrá ambigüedad y confusión en las tablas de enrutamiento de los clientes.
*## Por ejemplo, usaremos la dirección de subred **172.30.1.0/24**.*
El servidor recibirá automáticamente la dirección IP más baja del rango: "172.30.1.1"
server 172.30.1.0 255.255.255.0
Para registrar la asignación de direcciones IP realizada por el servidor:
ifconfig-pool-persist ipp.txt
Indicamos las redes internas que serán accesibles desde el exterior a través del servidor VPN.
Si miramos el esquema del comienzo, las Intranet privadas son 192.168.1.0/24 y 192.168.2.0/24
Esto modificará la tabla de enrutamiento de los clientes VPN en consecuencia
¡¡Importante!! Esas redes no deben aparecer previamente en la tabla de enrutamiento del
cliente ya que sino se producirá confusión en el cliente a la hora de saber como alcanzarlas
push "route 192.168.100.0 255.255.255.0"
push "route 192.168.200.0 255.255.255.0"

Si queremos modificar la tabla de enrutamiento de los clientes VPN para que todo el
tráfico dirigido hacía Internet sea redirigido a través del servidor VPN,
descomentaremos la siguiente línea. En ese caso, el servidor VPN hará de gateway a Internet
para los clientes VPN
;push "redirect-gateway"
Si deseamos configurar el DNS y WINS de los clientes:
;push "dhcp-option DNS 192.168.100.251"
;push "dhcp-option WINS 192.168.100.251"

Para permitir que los clientes VPN se vean entre sí:
client-to-client

Usuario suplantado por el servicio para restringir los privilegios
user nobody

```
group nogroup
## Para limitar el número de conexiones:
;max-clients 100

## Para auditar el estado de la VPN, y el nivel de auditoría o verbosidad:
status openvpn-status.log
log-append /var/log/openvpn.log
verb 4
```

Una vez creado el fichero de configuración del servidor VPN, iniciaremos el servicio. Para ello deberemos ejecutar el siguiente comando:

```
[root@servidorvpn]# openvpn --config /etc/openvpn/server.conf
```

¡¡Observación!! El comando **openvpn --config /etc/openvpn/server.conf** dejará al servidor a la escucha de solicitudes de conexión VPN procedentes de clientes, dejando la terminal inoperativa. Para poder seguir trabajando con el servidor será necesario abrir una nueva terminal o conexión SSH.
Una vez que se reinicie el servidor, el servicio VPN debería cargar automáticamente la configuración anterior indicada en **server.conf**, ya no siendo necesaria la ejecución expresa del comando anterior. Es decir, **OpenVPN** es un servicio que se inicia automáticamente al arrancar el equipo, busca en el directorio **/etc/openvpn** algún archivo de configuración, y si lo encuentra lo carga.
Para comprobar que el servicio VPN se inicia tras reiniciar el servidor tan sólo será necesario mirar su tabla de enrutamiento, **route -n**, y comprobar que en la columna de destinos aparece la red lógica indicada en la configuración de la VPN, **172.30.1.0/255.255.255.0**. Otra opción es comprobar que se ha auto creado la interfaz virtual que se usará para crear el túnel VPN, **ifconfig tun0**.

Y por último, para que el servidor VPN pueda hacer de puerta de enlace o gateway para los clientes VPN externos hacia los equipos internos de la Intranet o Intranets a las que da acceso, será necesario activar el "**ip_forwarding**" (reenvío *de paquetes entre sus diferentes interfaces de red*) y configurar la **NAT Postrouting**:

```
[root@servidorvpn]# echo "1" > /proc/sys/net/ipv4/ip_forward (equivalente a descomentar la línea "net.ipv4.ip_forward=1" en el archivo de configuración /etc/sysctl.conf)
[root@servidorvpn]# iptables -t nat -A POSTROUTING -j MASQUERADE
```

Estos últimos comandos sería conveniente añadirlos en el script "**/etc/init.d/rc.local**" para que se ejecuten al iniciar el sistema, o crear nuestro propio script de configuración de arranque dentro de "**/etc/init.d**" y posteriormente crear el correspondiente enlace simbólico a dicho script desde el directorio "**/etc/rc*X*.d**" en función del nivel de arranque de nuestro servidor, el cual se puede conocer ejecutando "**runlevel**" (*"X", se corresponderá con el nivel de arranque*).

10.6.- Configuración de los clientes VPN

Tras configurar el servidor, tan sólo tendremos que configurar el cliente VPN. Para ello, en primer lugar habrá que hacer llegar a los clientes, por ejemplo, mediante "**scp**" (*suponemos que el*

*cliente o el servidor tiene instalado "**openssh-server**"*), las claves generadas en el paso anterior, el certificado de cliente firmado por la CA, y el certificado de la CA. En el caso de que sea el equipo cliente de la VPN quien tenga instalado "**openssh-server**", desde el servidor se ejecutarían los siguientes comandos (*el usuario "usucliente" debe tener configuradas las ACL para tener permiso de escritura sobre el directorio /etc/openvpn*):

```
[root@servidorvpn]# scp /etc/openvpn/ca/cacert.crt usucliente@Ip_Cliente:/etc/openvpn
[root@servidorvpn]# scp /etc/openvpn/cliente1/cliente1.key usucliente@Ip_Cliente:/etc/openvpn
[root@servidorvpn]# scp /etc/openvpn/cliente1/cliente1cert.crt \
                      usucliente@Ip_Cliente:/etc/openvpn
```

En el caso de que sea al revés, que sea el propio servidor VPN el que tenga instalado el "**openssh-server**", será desde el propio cliente quien ejecutará el "**scp**" (*el usuario "ususervidor" debe tener permiso de lectura sobre el contenido de "/etc/openvpn"*):

```
[root@cliente1vpn]# scp ususervidor@IP_servidor:/etc/openvpn/ca/cacert.crt /etc/openvpn
[root@cliente1vpn]# scp ususervidor@IP_servidor:/etc/openvpn/cliente1/cliente1.key \
                      /etc/openvpn
[root@cliente1vpn]# scp ususervidor@IP_servidor:/etc/openvpn/cliente1/cliente1cert.crt  \
                      /etc/openvpn
```

Por último, crearemos el archivo de configuración del equipo cliente VPN "**/etc/openvpn/cliente1.conf**". En dicha configuración deberemos tener en cuenta lo siguiente:

- El servidor da el servicio vía protocolo "**udp**", por el puerto **1194**, tal como tiene configurado en su archivo de configuración.
- El tipo de dispositivo virtual necesario para comunicarse con el servidor será "**tun**", tal como justificamos en la configuración del servidor VPN.

```
dev tun
proto udp
```

- La solicitud de conexión desde el cliente no se realiza directamente sobre el servidor VPN, ya que este se encuentra en el interior de una Intranet que es inaccesible de manera directa. Es decir, la solicitud se realizará sobre la dirección IP pública que tenga asignada nuestro router ISP (*proporcionado por nuestro Proveedor de Servicios de Internet*), que en el ejemplo propuesto en este caso práctico es "**192.168.1.254**". Por tanto será necesario haber configurado una NAT prerouting en el router ISP para que redireccione las solicitudes de conexión recibidas por **udp/1194** hacía el equipo de la Intranet que hace de servidor VPN (*en el esquema 192.168.100.250*). Dicha configuración se ha realizado al comienzo de la práctica.

Ej. Práctico 10.6.1: Configuración de un Cliente VPN bajo GNU/Linux

Suponiendo que ya han sido enviados al cliente GNU/Linux los archivos necesarios para establecer la conexión VPN (*claves y certificados generados anteriormente*), en este ejercicio práctico se instalará el software OpenVPN y se creará el archivo de configuración del **cliente VPN**.

Solución Ej. Pr. 10.6.1.I.- *Cómo Configurar el Cliente VPN bajo GNU/Linux*

Para configurar el cliente tan sólo tendremos que crear el siguiente archivo de configuración, donde habrá que revisar las rutas asociadas a la ubicación de los archivos de claves y certificados:

```
[root@cliente1vpn]# apt-get install openvpn
[root@cliente1vpn]# nano /etc/openvpn/cliente1.conf
## Contenido del archivo de configuración "/etc/openvpn/cliente1.conf"
client
dev tun
proto udp
topology subnet
persist-key
persist-tun
resolv-retry infinite
nobind

## Indicamos la dirección IP pública del router ISP al que se hará la solicitud de conexión.
## Se podría poner más de uno para balancear carga.
remote 192.168.1.254 1194

## En el caso de querer desviar todo el trafico de Internet generado por el cliente
## a través del servidor VPN descomentaremos la siguiente línea:
;redirect-gateway def1

## Indicamos la ubicación de las claves del cliente y certficados
ca /etc/openvpn/cacert.crt
cert /etc/openvpn/cliente1cert.crt
key /etc/openvpn/cliente1.key

## Usuario suplantado por la VPN con privilegios restringidos
user nobody
group nogroup

## Indicamos que los datos de la comunicación entre cliente y servidor VPN viajarán comprimidos
comp-lzo
## Nivel de verbosidad: cantidad y tipo de mensajes de información que aparecerán por pantalla
verb 4
```

A continuación haremos la solicitud de conexión mediante la ayuda del archivo de configuración creado:

```
[root@cliente1vpn]# openvpn --config /etc/openvpn/cliente1.conf
```

¡¡Advertencia!! La ejecución del comando **openvpn --config /etc/openvpn/cliente1.conf** no será

necesario para establecer la VPN una vez se reinicie la máquina, ya que en GNU/Linux **OpenVPN** funciona como un servicio más dentro del sistema cargando de manera automática todo aquel archivo de configuración que se localice dentro del directorio **/etc/openvpn**. En el caso de que no queramos que la VPN se inicie de manera automática habrá ubicar el archivo de configuración del cliente **cliente1.conf** en otro directorio diferente o indicar al sistema que el servicio **openvpn** no se inicie de manera automática sino manual. Por ejemplo, en Ubuntu Server suponiendo que el sistema se inicia en modo 2 (*puede consultarse ejecutando el comando **runlevel***), ejecutaríamos el siguiente comando para deshabilitar el inicio automático de **openvpn**:

```
[root@cliente1vpn]# update-rc.d openvpn disable 2
```

Para comprobar el correcto funcionamiento de la VPN, deberemos advertir cambios producidos en la tabla de enrutamiento del equipo cliente de la VPN, ya que deben aparecer nuevas reglas de enrutamiento necesarias para que el equipo cliente sepa como alcanzar las Intranet privadas reales a través del servidor VPN. A continuación se muestran las tablas antes y después de establecer la conexión con la VPN:

[root@cliente1vpn]# **route -n** *(antes de iniciar la conexión VPN)*

Destino	Gateway	Máscara	Flag	Métrica	Interfaz
192.168.100.0	0.0.0.0	255.255.255.0	U	0	eth0
0.0.0.0	192.168.100.110	0.0.0.0	UG	0	eth0

[root@cliente1vpn]# **route -n** *(después de iniciar la conexión VPN)*

Destino	Gateway	Máscara	Flag	Métrica	Interfaz
172.30.1.0	**0.0.0.0**	**255.255.255.0**	**U**	**0**	**tun0**
192.168.1.0	**172.30.1.1**	**255.255.255.0**	**UG**	**0**	**tun0**
192.168.2.0	**172.30.1.1**	**255.255.255.0**	**UG**	**0**	**tun0**
192.168.100.0	0.0.0.0	255.255.255.0	U	0	eth0
0.0.0.0	192.168.100.110	0.0.0.0	UG	0	eth0

Si habilitáramos la directiva **push "redirect-gateway"** en el archivo de configuración del servidor **server.conf** y "**redirect-gateway def1**" en el del cliente, además de las reglas anteriores, también aparecerían nuevas reglas que enrutarían todo el trafico de Internet generado por el cliente de la VPN, hacía el servidor de la VPN (*puede tener interés para control los accesos a Internet realizados por el equipo cliente*), pasando este hacer de gateway o puerta de enlace a Internet para los clientes VPN:

[root@cliente1vpn]# **route -n** *(incluyendo "**redirect-gateway def1**")*

Destino	Gateway	Máscara	Flag	Métrica	Interfaz
172.30.1.0	0.0.0.0	255.255.255.0	U	0	tun0
192.168.1.0	172.30.1.1	255.255.255.0	UG	0	tun0
192.168.2.0	172.30.1.1	255.255.255.0	UG	0	tun0
192.168.100.0	0.0.0.0	255.255.255.0	U	0	eth0
0.0.0.0	**172.30.1.1**	**0.0.0.0**	**UG**	**1000**	**tun0**
0.0.0.0	192.168.100.110	0.0.0.0	UG	0	eth0

Por último, deberíamos comprobar que el cliente VPN puede hacer uso de los servicios

ofrecidos en la Intranet haciendo uso de las direcciones IP privadas de la propia Intranet. Por ejemplo, si en la Intranet hubiera un servidor SAMBA con dirección IP 192.168.200.1, deberíamos poder acceder a sus recursos compartidos.

Ej. Práctico 10.6.2: Configuración de un Cliente VPN bajo Windows

Suponiendo que ya han sido enviados al cliente Windows los archivos necesarios para establecer la conexión VPN (*claves y certificados generados anteriormente*), en este ejercicio práctico instalaremos el software **OpenVPN** y crearemos el archivo de configuración del **cliente VPN**.

Solución Ej. Pr. 10.6.2.I.- Cómo Configurar el Cliente VPN bajo Windows

Para instalar **OpenVPN** en Windows lo descargaremos de Internet desde alguno de los muchos sitios Web disponibles (*p.e. https://openvpn.net/index.php/open-source/downloads.html*).

A continuación mediante algún editor de texto (*p.e. bloc de notas*) crearemos el archivo de configuración del cliente VPN **cliente1.conf** dentro de la mismo directorio donde hemos almacenado el resto de archivos del cliente (*claves y certificados previamente generados en otro ejercicio práctico*):

```
## Contenido del archivo de configuración "cliente1.conf"
client
dev tun
proto udp
topology subnet
persist-key
persist-tun
resolv-retry infinite
nobind

## Indicamos la dirección IP pública del router ISP al que se hará la solicitud de conexión.
## Se podría poner más de uno para balancear carga.
remote 192.168.1.254 1194

## En el caso de querer desviar todo el trafico de Internet generado por el cliente
## a través del servidor VPN descomentaremos la siguiente línea:
;redirect-gateway def1

## Indicamos la ubicación de las claves del cliente y certficados
ca /etc/openvpn/cacert.crt
cert /etc/openvpn/cliente1cert.crt
key /etc/openvpn/cliente1.key

## Usuario suplantado por la VPN con privilegios restringidos
user nobody
group nogroup
```

```
## Indicamos que los datos de la comunicación entre cliente y servidor VPN viajarán comprimidos
comp-lzo
## Nivel de verbosidad: cantidad y tipo de mensajes de información que aparecerán por pantalla
verb 4
```

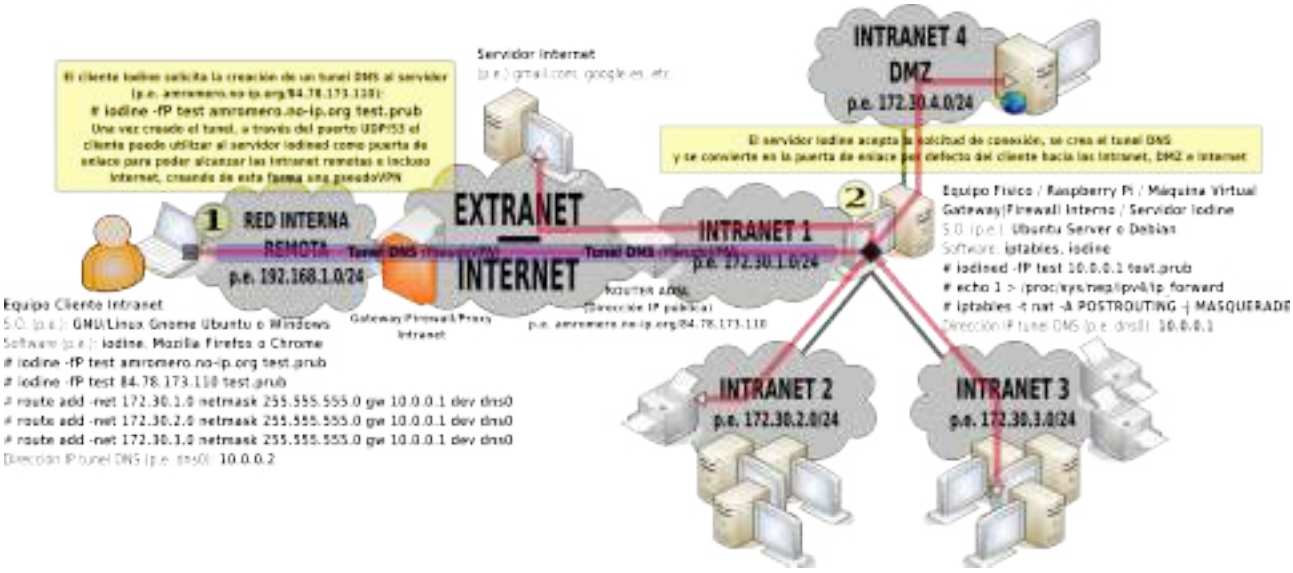
Por último, ejecutaremos el software OpenVPN y cargaremos el archivo de configuración del cliente. Al igual que con el cliente GNU/Linux, en Windows podremos comprobar el correcto funcionamiento consultando su tabla de enrutamiento (*comando* ***route print***) y advirtiendo que deben aparecer nuevas reglas de enrutamiento que le indican como llegar a las Intranet remotas utilizando como puerta de enlace o gateway al servidor VPN 172.30.1.1.

Ej. Práctico 10.6.3: Pseudo-VPN mediante un Túnel DNS

Mediante el siguiente ejercicio aprenderemos a crear un túnel DNS con la finalidad de poder alcanzar los recursos de una red privada remota (*impresoras, servidores de archivos, bases de datos, etc.*), de forma equivalente a una VPN. La implementación de un **Túnel DNS** ya se mostró en el capítulo de prácticas asociadas al Proxy HTTP Caché Squid, con la finalidad saltarnos un portal cautivo, por lo que este ejercicio tan sólo servirá para repasar lo allí visto. Para ello haremos uso del software **iodine**, el cual nos permitirá emular a un pseudo-servidor DNS y crear un túnel DNS bajo el cual podremos encapsular cualquier tipo de tráfico (*HTTP/HTTPS, FTP/FTPS, etc.*).

Solución Ej. Pr. 10.6.3.I.- Cómo Implementar una Pseudo-VPN con un Túnel DNS

A continuación se va a implementar una "especie de VPN" (*pseudo-vpn*) mediante un túnel DNS udp/53, que nos permitirá encapsular todo el trafico que genere el cliente hacia un servidor, para que este a su vez lo redireccione hacia quien corresponda (*p.e. un equipo de la propia Intranet o Internet*).



¡¡Advertencia!! Antes de empezar a implementar la práctica deberíamos tener en cuenta que un túnel DNS puede servirnos para implementar una pseudo-VPN de manera muy sencilla, o también podemos usarlo como vía de escape para poder navegar a través de una red controlada mediante un portal cautivo sin necesidad de autenticarnos, pero teniendo presente las limitaciones de ancho de banda que presenta el túnel a consecuencia de querer tunelizar tráfico HTTP/HTTPS/FTP/FTPS/etc. sobre el protocolo DNS udp/53, el cual esta pensado originalmente para un trasiego de tramas de tamaño reducido y con necesidades de ancho de banda muy reducidas. Los desarrolladores de **iodine** (*software utilizado para la realización de la práctica*) advierten que el ancho de banda resultante que un equipo cliente puede llegar a obtener es de **680 Kbit/s** de subida y **2,3 Mbit/s** de bajada en una Intranet cableada, y de **50 Kbit/s** de subida y **200 Kbit/s** de bajada en una Intranet inalámbrica. Por este motivo, el uso de un túnel DNS puede ser interesante cuando queremos gestionar un servidor remoto mediante ssh (*p.e. un servidor Apache ubicado en una DMZ*), o

queremos consultar un sitio Web con poca carga de contenidos, pero no en casos donde se requiera un importante ancho de banda.

Por esta razón, al final de la práctica se sugiere medir la velocidad de la conexión (*p.e. mediante speedtest*), y así advertir que tipo de conexiones abrir a través del túnel DNS.

En concreto para su implementación seguiremos los siguientes pasos, empezando en primer lugar por la configuración el servidor, y después con la del cliente:

Paso Nº1.- Comenzaremos instalando el software servidor **iodine** en el servidor e iniciando su servicio **iodined** (*demonio o servicio asociado a iodine*). Señalar que las opciones de inicialización del servicio **iodined** son muy variadas (*man iodine*), pero destacaremos las siguientes:

iodined [-f] [-P password] *dirección_IP*[/máscara] *nombre_dominio*

- **-f**, parámetro opcional que provoca que el servicio **iodined** se ponga a la escucha en modo foreground, evitando de esta forma que lo haga en background, lo cual nos permitirá desactivarlo en cualquier momento tecleando un CTRL+C.
- **dirección_IP**[/máscara], parámetro obligatorio que informará al servicio **iodined** de cual será la dirección IP que recibirá el extremo servidor del túnel DNS. En concreto, **iodined** creará una interfaz de red virtual en el equipo servidor, p.e. **dns0**, la cual recibirá la dirección IP indicada, modificando en consecuencia su tabla de enrutamiento, y que usará para comunicarse con el otro extremo del túnel (*equipo cliente*). Por tanto, la elección de la dirección IP establecerá el rango de direcciones IP privadas que se usarán en la comunicación entre cliente y servidor, motivo por el cual es obligatorio que dicho rango de direcciones IP no se este utilizando simultáneamente en las redes privadas o Intranet en las cuales se encuentran cliente y servidor, ya que en ese caso la tabla de enrutamiento ocasionaría una ambigüedad al sistema al tomar la decisión de como enrutar los paquetes hacia el destino. En el caso práctico que aquí se resuelve se usa el rango de
- **-P**, parámetro opcional que nos permitirá indicar una password que será necesaria para autenticarse. En caso de no indicarla se solicitará al ejecutar el comando **iodined**.
- **nombre_dominio**, parámetro obligatorio que establecerá bajo que resolución de nombres de dominio se encapsulará el trafico que genere el cliente hacia el servidor.

```
[root@servidor]# apt-get install iodine
[root@servidor]# iodined -f -P mipass 10.0.0.1/24 test.prub
```

Una vez iniciado el servicio podremos comprobar el aspecto que tiene su nueva tabla de enrutamiento abriendo una nueva terminal (*iodined se inicia en modo foreground dejando inoperativa la terminal*). En concreto, deberemos fijarnos en que se ha creado una nueva interfaz de red virtual, p.e. **dns0**, a través de la cual se enrutará todo el trafico dirigido hacia equipos de la red 10.0.0.0/24 (*tal como se puede observar en la anterior figura, se supone que las direcciones de red de las Intranet donde se encuentra el servidor iodined son las 172.30.1.0/24, 172.30.2.0/24, 172.30.3.0/24, 172.30.4.0/24*).

```
[root@servidor]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric         Iface
```

0.0.0.0	172.30.1.1	0.0.0.0	UG	0	eth0
10.0.0.0	**0.0.0.0**	**255.255.255.0**	**U**	**0**	**dns0**
172.30.1.0	0.0.0.0	255.255.255.0	U	0	eth0
172.30.2.0	0.0.0.0	255.255.255.0	U	0	eth1
172.30.3.0	0.0.0.0	255.255.255.0	U	0	eth2
172.30.4.0	0.0.0.0	255.255.255.0	U	0	eth3

Por último, para terminar de configurar el servidor **iodined**, deberemos activar el ip_forward y el enmascaramiento de los paquetes (*repasar prácticas del firewall*), con la finalidad de que el servidor pueda hacer las veces de puerta de enlace para los equipos clientes que establezcan el túnel DNS.

```
[root@servidor]# echo 1 > /proc/sys/net/ipv4/ip_forward
[root@servidor]# iptables -t nat -A POSTROUTING -j MASQUERADE
```

¡¡Observación Importante!! Para comprobar que el servicio **iodined** está a la escucha, puede hacerse una exploración de puertos mediante los comandos **netstat** o **nmap** (*nmapfe o zenmap son herramientas basadas en nmap con entorno gráfico*).

En el caso de usar **netstat**, es conveniente usar los siguientes parámetros para no obtener demasiada información: (1) **-l**, para filtrar los servicios que están a la escucha (*listening*), (2) **-u**, para indicar que en el resultado del escáner solo muestre información sobre los servicios UDP (*el servicio DNS al que suplanta iodined trabaja por defecto bajo udp/53*), (3) **-p**, para poder identificar el PID del proceso que da dicho servicio, y (4) **-n**, para informar mediante direcciones IP en lugar de nombres de dominio asociados. No obstante, también podemos filtrar al máximo el resultado apoyándonos en el comando **grep**.

```
[root@servidor]# netstat -l -u -p -n
[root@servidor]# netstat -lupn | grep ":53 "
[root@servidor]# netstat -lupn | grep iodined
udp        0      0 0.0.0.0:53              0.0.0.0:*                           32502/iodined
```

En el caso de usar **nmap**, sabiendo que por defecto **iodined** ofrece el servicio vía UDP por el puerto 53 (*suplanta a un servidor DNS*), le diremos que analice ese puerto directamente para evitar tiempo de computo innecesario. Las opciones **-sU** y **-p** nos permiten indicarle a **nmap** que únicamente analice protocolos UDP en el rango de puertos que le especifiquemos:

```
[root@servidor|cliente|equipo_red]# nmap -sU -p 53 dirección_IP_Servidor_iodined
PORT      STATE          SERVICE
53/udp    open|filtered  domain
```

Según lo anterior deberemos tener en cuenta que en el equipo servidor donde iniciemos **iodined** no debe estar iniciado simultáneamente ningún otro software servidor que de servicio por el mismo puerto **udp/53** (*algún servicio DNS*), como podría ser **bind9**. En caso contrario, se produciría un conflicto entre ambos servicios, no pudiendo establecerse el túnel.

Paso Nº2.- Tras configurar el servidor, continuaremos instalando **iodine** en el equipo cliente para

posteriormente establecer el túnel DNS de comunicación. Respecto a los parámetros que podrían usarse en la aplicación cliente **iodine**, podríamos destacar los siguientes:

iodine [-f] [-P password] [dirección_IP_Servidor] *nombre_dominio*

- **-f**, parámetro opcional, que al igual que en el servidor, provoca que **iodine** establezca el túnel de comunicación con el servidor en modo foreground, evitando de esta forma que lo haga en background, lo cual nos permitirá desactivarlo en cualquier momento tecleando un CTRL+C.
- **-P**, parámetro opcional que nos permitirá indicar una password que será necesaria en la autenticación contra el servidor. Obviamente indicaremos la que se asigno en el servidor al iniciar el servicio **iodined**. En caso de no indicarla se solicitará al ejecutar el comando **iodine**.
- **dirección_IP_Servidor**, parámetro opcional que informará a **iodine** de cual será la dirección IP o nombre de dominio del servidor DNS que será usado para salir a Internet y conocer de esta forma la dirección IP asociada al servidor **iodined**. En concreto, para que la solicitud quede totalmente camuflada, debería adquirirse un nombre de dominio de segundo nivel (*2LD, second Level Domain*) con gestión propia, p.e. amromero.es, de tal forma que podamos asociar a su **registro NS** (*NameServer*) la dirección IP del servicio **iodined**. No obstante, si lo único que pretendemos de momento es realizar pruebas de funcionamiento, podremos poner directamente la dirección IP del servidor **iodined**, que es lo que se ha hecho en la resolución de la práctica.
- **nombre_dominio**, relacionado con lo indicado en el punto anterior, este parámetro obligatorio indicará el nombre del dominio que cuya resolución nos llevará a conocer la dirección IP del servidor **iodined**. Éste deberá estar en consonancia con el dominio indicado al iniciar el servidor **iodined**. En el caso práctico que aquí se muestra, al indicar como dirección IP del servidor, la del servidor **iodined**, el nombre de dominio puede ser ficticio (*p.e. test.prub*).

```
[root@cliente]# apt-get install iodine
[root@cliente]# iodine -f -P mipass dirección_IP_Servidor_iodined test.prub
...
Setting IP of dns0 to 10.0.0.2
Setting MTU of dns0 to 1130
Server tunnel IP is 10.0.0.1
Testing raw UDP data to the server (skip with -r)
Server is at dirección_ip_Servidor_iodined, trying raw login: OK
Sending raw traffic directly to dirección_ip_Servidor_iodined
Connection setup complete, transmitting data.
```

Tras lanzar el comando anterior y establecerse exitosamente el túnel DNS, debería haberse creado en el cliente, al igual que ocurrió en el servidor, una interfaz de red virtual **dns0** con una dirección IP del mismo rango reservado por el servidor **iodined 10.0.0.0/24**. En concreto, si es el primer cliente que se conecta recibirá la dirección IP siguiente en el rango a la que se asigno al servidor **10.0.0.2**. Es decir, que el servidor **iodined** actúa como un servidor DHCP asignando a los clientes una dirección IP del rango reservado.

Por lo tanto, si visualizamos la tabla de enrutamiento del equipo cliente deberíamos ver algo

parecido a lo siguiente (*se supone que el equipo cliente se encuentra en una Intranet con dirección de red 192.168.1.0/24, y que su interfaz de red física es inalámbrica, wlan0*):

[root@**cliente**]# **route -n**

Destino	Pasarela	Genmask	Indic	Métric	Interfaz
0.0.0.0	192.168.1.254	0.0.0.0	UG	0	wlan0
10.0.0.0	0.0.0.0	255.255.255.0	U	0	**dns0**
192.168.1.0	0.0.0.0	255.255.255.0	U	9	wlan0

Por último, para aprovecharnos del túnel DNS creado, añadiremos las reglas de enrutamiento estáticas necesarias, con la finalidad de poder alcanzar los equipos pertenecientes a las redes privadas remotas del otro extremos del túnel, usando como puerta del enlace o gateway al servidor **iodined**:

```
[root@cliente]# route add -net 172.30.1.0 netmask 255.255.255.0 gw 10.0.0.1 dev dns0
[root@cliente]# route add -net 172.30.2.0 netmask 255.255.255.0 gw 10.0.0.1 dev dns0
[root@cliente]# route add -net 172.30.3.0 netmask 255.255.255.0 gw 10.0.0.1 dev dns0
[root@cliente]# route add -net 172.30.4.0 netmask 255.255.255.0 gw 10.0.0.1 dev dns0
```

En el caso de que quisiéramos utilizar también al servidor **iodined** como puerta de enlace por defecto, deberíamos eliminar la puerta de enlace actual y añadir la nueva, sin olvidar de que debemos añadir las reglas necesarias que nos permitan seguir manteniendo el túnel DNS creado (*supondremos que el servidor DNS del cliente es el 8.8.8.8*):

```
[root@cliente]# route add -host ip_publiva_Ser_iodined netmask 255.255.255.255 \
                gw 192.168.1.254 del wlan0
[root@cliente]# route add -host 8.8.8.8 netmask 255.255.255.255 \
                gw 192.168.1.254 del wlan0
[root@cliente]# route del default gw 192.168.1.254
[root@cliente]# route add default gw 10.0.0.1
```

De esta forma la tabla de enrutamiento resultante podría quedar así:

[root@**cliente**]# **route -n**

Destino	Pasarela	Genmask	Indic	Métric	Interfaz
192.168.1.0	0.0.0.0	255.255.255.0	U	9	wlan0
ip_pub_Ser_iodined	192.168.1.254	255.255.255.255	**UH**	0	wlan0
8.8.8.8	192.168.1.254	255.255.255.255	**UH**	0	wlan0
10.0.0.0	0.0.0.0	255.255.255.0	U	0	**dns0**
172.30.1.0	**10.0.0.1**	255.255.255.0	**UG**	0	**dns0**
172.30.2.0	**10.0.0.1**	255.255.255.0	**UG**	0	**dns0**
172.30.3.0	**10.0.0.1**	255.255.255.0	**UG**	0	**dns0**
172.30.4.0	**10.0.0.1**	255.255.255.0	**UG**	0	**dns0**
0.0.0.0	**10.0.0.1**	0.0.0.0	**UG**	0	**dns0**

Ahora podemos comprobar el correcto funcionamiento de la pseudo-VPN que hemos creado comprobando que existe conectividad con los equipos de las Intranet privadas remotas (*p.e.*

mediante el uso de ping):

```
[root@cliente]# ping 10.0.0.1
[root@cliente]# ping 172.30.1.1
```

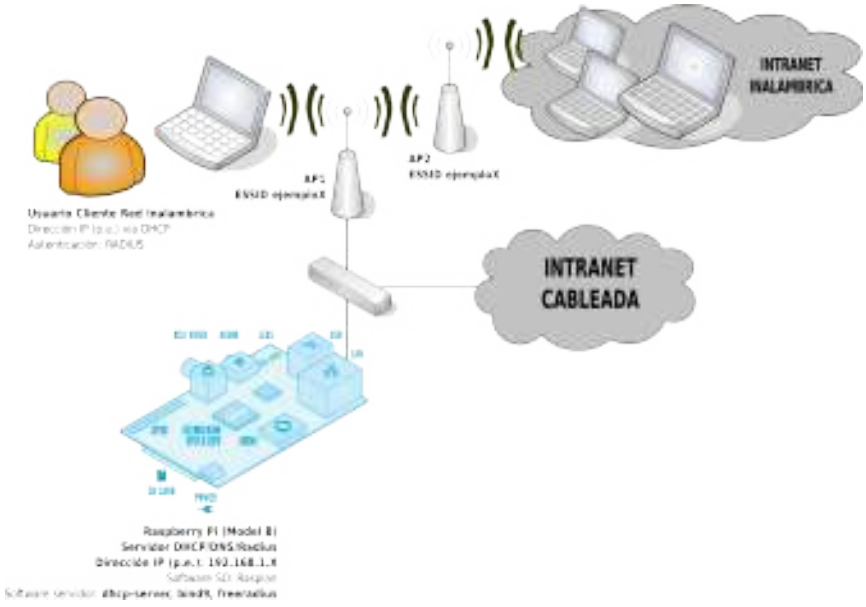
Para terminar con la práctica, sería conveniente hacer un test de velocidad y así comprobar el ancho de banda que nos ofrece un túnel DNS, de tal forma que seamos conscientes de que tipo de aplicaciones podemos tunelizar: conexiones SSH, tráfico HTTP/HTTPS con pequeña carga en sus *contenidos* (*texto, imágenes, etc.*), transferencia de archivos de reducido tamaño, etc. Para la medición podemos descargarnos la aplicación python **speedtest**:

```
[usuario@cliente]$ wget https://raw.github.com/sivel/speedtest-cli/master/speedtest_cli.py
[usuario@cliente]$ chmod +x speedtest_cli.py
[usuario@cliente]$ ./speedtest_cli.py
Retrieving speedtest.net configuration...
Retrieving speedtest.net server list...
Testing from Orange Espana (dirección_IP_pública_asociada_Servidor_iodined)...
Selecting best server based on ping...
Hosted by iA Soft Aragon (Zaragoza) [0.54 km]: 29.643 ms
Testing download speed.....................................
Download: 0.08 Mbits/s
Testing upload speed.....................................................
Upload: 0.10 Mbits/s
```

Práctica Nº11.- Autenticación de Usuarios en Redes Wireless: Servidor RADIUS

Siguiendo con redes inalámbricas, en su implementación nos puede interesar disponer de un servidor de autenticación que controle el acceso de los equipos clientes a la red Wireless, pero no de la manera clásica, sino mediante el uso de un login y una password. Es decir, en la mayoría de las redes Wireless actuales el único requisito de acceso, independientemente de quien sea el usuario que acceda, es conocer una frase de paso o contraseña (WEP/WPA/WPA2), pero puede interesarnos que el acceso sea más personalizado: controlar quien accede mediante un login/password por usuario (*no por máquina)*, a que servicios se le da acceso una vez autentificado, más una auditoría de todo ello.

Una herramienta software que nos va a permitir, entre otras muchas opciones, gestionar la autenticación en una red inalámbrica es hacer uso de un servicio RADIUS. De esta forma, el usuario que quiera hacer uso de la infraestructura Wireless, será necesario que el servidor RADIUS le valide el login y la password introducidos por él. Además, como servidor de autenticación se propone usar una Raspberry Pi, aunque puede implementarse igualmente mediante una máquina virtual funcionando, por ejemplo, bajo Ubuntu Server o Debian.



11.1.- Introducción al servicio RADIUS.

RADIUS (*Remote Authentication Dial-In User Server*) es un protocolo ampliamente empleado para controlar el acceso a servicios en red. Para ello haremos uso de un software servidor de código abierto llamado FreeRADIUS, y lo configuraremos para un uso concreto: controlar el acceso a una red inalámbrica.

Dentro de esta parte práctica, lo primero que vamos a estudiar es la teoría asociada a este protocolo, viendo también conceptos relacionados con el mismo, como AAA, NAS, y mecanismos de protección de redes inalámbricas (como WEP y WPA).

Después instalaremos FreeRADIUS en nuestra Raspberry y la configuraremos como servidor para que dé servicio a los distintos puntos de acceso que pueden encontrarse en la Intranet a la que pertenece. Obviamente también tendremos que configurar el punto de acceso para que funcione como cliente del servicio ofrecido por la Raspberry. En concreto necesitaremos informar a los puntos de acceso de la dirección IP de la Raspberry, el puerto por el que da el servicio RADIUS, y una clave secreta que validará el servidor RADIUS antes de atender una solicitud de autenticación procedente de un punto de acceso. Por último, probaremos nuestra nueva configuración de la red inalámbrica conectando a dicha red una máquina cliente de la Intranet bajo GNU/Linux o Windows.

En definitiva, la estructura que vamos a implementar, es una estructura cliente-servidor de doble nivel. Es decir, el equipo cliente de la Intranet solicita un servicio Wireless al punto de acceso, y este a su vez hace de cliente para el servidor RADIUS, al solicitarle la comprobación de autenticación en relación al login y passwords introducidos por el cliente de la Intranet.

A continuación, se introducirán diversos conceptos cuyo conocimiento es clave para poder entender cuestiones posteriores de la práctica.

11.2.- Definición de RADIUS

RADIUS (*Remote Authentication Dial-In User Server*) es un protocolo que nos permite gestionar la "**autenticación, autorización y registro**" de usuarios remotos sobre un determinado recurso. La tupla "autenticación, autorización y registro" es más conocida como **AAA**, al ser éste su acrónimo de su denominación original inglesa "**Authentication, Authorization, and Accounting**". Veamos a continuación a qué se refiere cada uno de estos términos:

- **Autenticación** (*authentication*) hace referencia al proceso por el cual se determina si un usuario tiene permiso para acceder a un determinado servicio de red del que quiere hacer uso. El proceso de autenticación se realiza mediante la presentación de una identidad y unos credenciales por parte del usuario que demanda acceso.

Un tipo habitual de credencial es el uso de una contraseña (*o password*) que junto al nombre de usuario nos permite acceder a determinados recursos. El nombre de usuario es nuestra identidad, que puede ser públicamente conocida, mientras que la contraseña se mantiene en secreto, y sirve para que nadie suplante nuestra identidad. Otros tipos más avanzados de credenciales son los certificados digitales tal como se ha visto en el servicio SSH, para crear usuarios de confianza.

Existen muchos métodos concretos que implementan el proceso de la autenticación. Algunos de ellos, soportados por RADIUS, son:

- **Autenticación de sistema** (*system authentication*), típica en un sistema Unix, normalmente realizada mediante el uso del fichero "/etc/passwd".
- Los **protocolos PAP** (*Password Authentication Protocol*), y su versión segura CHAP (Challenge Handshake Authentication Protocol), que son métodos de autenticación usados por proveedores de servicios de Internet (ISPs) accesibles vía PPP.
- **LDAP** (*Lightweight Directory Access Protocol*), un protocolo a nivel de aplicación (sobre TCP/IP) que implementa un servicio de directorio ordenado, y muy empleado como base de datos para

contener nombres de usuarios y sus contraseñas.
- **Kerberos**, el famoso método de autenticación diseñado por el MIT.
- **EAP** (*Extensible Authentication Protocol*), que no es un método concreto sino un entorno universal de autenticación empleado frecuentemente en redes inalámbricas y conexiones punto a punto.
- Freeradius también se permite la autenticación basada en ficheros locales de configuración del propio servidor RADIUS "**/etc/freeradius/users**".

- **Autorización** (*authorization*) se refiere a conceder servicios específicos (entre los que se incluye la "negación de servicio") a un determinado usuario, basándose para ellos en su propia autenticación, los servicios que está solicitando, y el estado actual del sistema. Es posible configurar restricciones a la autorización de determinados servicios en función de aspectos como, por ejemplo, la hora del día, la localización del usuario, o incluso la posibilidad o imposibilidad de realizar múltiples "logins" de un mismo usuario.

El proceso de autorización determina la naturaleza del servicio que se concede al usuario, como son: la dirección IP que se le asigna, el tipo de calidad de servicio (QoS) que va a recibir, el uso de encriptación, o la utilización obligatoria de túneles para determinadas conexiones.

Los métodos de autorización soportados habitualmente por un servidor de RADIUS incluyen bases de datos LDAP, bases de datos SQL (como Oracle, MySQL y PostgreSQL), o incluso el uso de ficheros de configuración locales al servidor.

No se debe confundir los términos autenticación con autorización. Mientras que la autenticación es el proceso de verificar un derecho reclamado por un individuo (persona o incluso ordenador), la autorización es el proceso de verificar que una persona ya autenticada tiene la autoridad para efectuar una determinada operación.

- **Registro** (*accounting, a menudo traducido también como contabilidad*) se refiere a realizar un registro del consumo de recursos que realizan los usuarios. El registro suele incluir aspectos como la identidad del usuario, la naturaleza del servicio prestado, y cuándo empezó y terminó el uso de dicho servicio. Es decir, auditar.

Es interesante el uso del protocolo RADIUS cuando tenemos redes de dimensiones considerables sobre las que queremos proporcionar un servicio de acceso centralizado (*aunque posiblemente jerarquizado por medio de diversos servidores RADIUS*). Por este motivo, uno de los principales usos de RADIUS se encuentra en empresas que proporcionan acceso a Internet o grandes redes corporativas, en un entorno con diversas de tecnologías de red (*incluyendo módems, xDSL, VPNs y redes inalámbricas*) no sólo para gestionar el acceso a la propia red, sino también para servicios propios de Internet (*como e-mail, Web o incluso dentro del proceso de señalización SIP en VoIP*).

Un uso de RADIUS que queremos enfatizar mediante esta práctica con Raspberry, es la autenticación en redes inalámbricas (Wi-Fi), sustituyendo métodos más simples de clave compartida (*pre-shared key, PSK*), que son bastante limitados al gestionar una red cuando ésta alcanza un determinado tamaño.

Aunque RADIUS es el protocolo para AAA más extendido en la actualidad, ya existe un

nuevo protocolo que está llamado a sustituir a RADIUS. Su nombre es DIAMETER, y también proporciona manejo de errores y comunicación entre dominios.

11.3.- Definición de NAS

¡¡Aclaración!! No confundir con la definición de NAS vista en otras prácticas que se harán con la Raspberry como "NAS, Network-Attached Storage", que comúnmente se refiere a unidades de almacenamiento compartidas en red.

Un **Network Access Server** (NAS) es un sistema que proporciona acceso a la red. En algunos casos también se conoce como Remote Access Server (RAS) o Terminal Server. En general, NAS es un elemento que controla el acceso a un recurso protegido, que puede ser desde un sencillo teléfono para VoIP o una impresora, hasta el acceso a una red inalámbrica o a Internet (*proporcionado por un ISP*).

Cuando un cliente quiere hacer uso de uno de estos servicios se conecta a NAS, quien a su vez se conecta a un servidor de AAA (*típicamente RADIUS*) preguntando si las credenciales proporcionadas por el cliente son válidas. Basándose en su respuesta, NAS le permitirá acceder o no a este recurso protegido. El sistema NAS no contiene ninguna información sobre los usuarios que se pueden conectar ni sus credenciales, sino que utiliza esta información para enviarla a RADIUS, y que éste le informe sobre los permisos del cliente.

Observa que nos encontramos en un escenario en el que hay dos niveles de la arquitectura cliente-servidor. Desde un punto de vista más global, tenemos la típica arquitectura en la que un usuario quiere acceder a un servicio, siendo el usuario el cliente, y el servidor el sistema que proporciona dicho servicio. Sin embargo, si nos centramos en el proceso de AAA, el cliente sería el sistema que proporciona el acceso a la red (*p.e. NAS*), mientras que el servidor es el sistema que autoriza o no dicho acceso (*p.e. RADIUS*). Como esta práctica se centra en este proceso, nosotros hablaremos de servidores RADIUS cuyos clientes son los elementos a proteger (*p.e. un punto de acceso para la conexión inalámbrica*). Por tanto, desde nuestro punto de vista, los usuarios que quieren acceder al recurso protegido (*p.e. La persona que se desea conectar a la red inalámbrica por medio del punto de acceso*), no son clientes de RADIUS sino que se denominan suplicantes.

Una ventaja del uso de RADIUS es que sus clientes tan sólo tienen que implementar el protocolo de comunicación con RADIUS, y no todas las posibilidades de AAA existentes (PAP, CHAP, LDAP, kerberos, mySQL, etc.). En el ejemplo del punto de acceso, tan sólo necesitamos implementar una solución NAS que realice las consultas a RADIUS.
Otra ventaja del protocolo RADIUS es que, en su comunicación con NAS, nunca transmite las contraseñas directamente por la red, lo que se conoce como en "Cleartext", ni siquiera al usar PAP, sino que usa algoritmos para ocultar las contraseñas como MD5. Sin embargo, al no ser considerado MD5 un sistema de protección de credenciales demasiado seguro, es aconsejable utilizar sistemas adicionales de protección para cifrar el tráfico de RADIUS, como puede ser túneles de IPsec.

11.4.- Definición de Seguridad en Tecnologías de Red Inalámbrica

En redes inalámbricas se utiliza un punto de acceso (*wireless access point, WAP o simplemente AP*) para interconectar todos los dispositivos inalámbricos de la red. Usualmente un AP

se conecta también a una red cableada, transmitiendo datos entre los dispositivos conectados a la red cable y los dispositivos inalámbricos.

La seguridad es un tema importante en las redes inalámbricas porque, al contrario que en una red cableada a la que sólo tienen acceso las personas que físicamente pueden conectarse, cualquier persona de la calle o pisos o edificios vecinos pueden conectarse a una red inalámbrica o ver el contenido de los paquetes que circulan por ella si ésta no está convenientemente protegida. Algunos de los principales protocolos estándar para proporcionar seguridad en redes inalámbricas IEEE 802.11 son:

- **WEP** (*Wired Equivalent Privacy*). Fue introducido en 1997 con objeto de proporcionar un nivel de confidencialidad similar al de las redes cableadas. Usa una clave estática de 64 ó 128 bits con el algoritmo RC4. Su uso se desaconseja completamente, ya que aunque es muy fácil de configurar y está muy extendido al ser el primero que surgió, presenta graves fallos de seguridad.
- **WPA** (*Wi-Fi Protected Access*) fue creado para corregir los múltiples fallos detectados en el protocolo WEP. WPA fue diseñado por el consorcio "Wi-Fi Alliance" basándose en un borrador del estándar 802.11i (*es un subconjunto del mismo*), y utiliza TKIP (Temporal Key Integrity Protocol) como protocolo de cifrado que sustituye a WEP sin necesidad de modificar el hardware existente (podría funcionar actualizando el firmware). En concreto, WPA sigue usando RC4 como algoritmo de cifrado con claves de 128 bits, pero usa TKIP para cambiar dinámicamente estas claves. WPA fue diseñado para ser usado junto a un servidor AAA (*habitualmente RADIUS*), de manera que se le asignan claves distintas a cada uno de los posibles usuarios. Sin embargo, para entornos domésticos o pequeñas oficinas también se puede usar, de forma menos segura, con una única clave compartida (*pre-shared key, PSK*). En este caso hablamos de WPA-PSK.
- WPA2 se basa en el nuevo estándar 802.11i, y el cambio más significativo respecto a WPA es que usa el protocolo de cifrado AES en lugar de RC4. Mientras que WAP puede ejecutarse en el hardware que soporte WEP, WAP2 necesita un hardware más nuevo. Sin embargo, se sabe que WAP también terminará siendo comprometido a medio plazo y por tanto sólo se recomienda como transición a WAP2.

Otro concepto relacionado con la seguridad en redes inalámbricas que merece la pena destacar es **EAP** (*Extensible Authentication Protocol*). EAP es un marco general de autenticación, y no un mecanismo de autenticación concreto. EAP proporciona algunas funciones comunes y un método para negociar el mecanismo de autenticación a usar. Actualmente hay más de 40 métodos distintos. En esta práctica haremos uso del denominado EAP protegido (**PEAP**) para la autenticación de nuestro usuario en la red inalámbrica, ya que los suplicantes, equipos bajo GNU/Linux o Windows, o un móvil bajo sistema Android soportan **PEAP** con **MSCHAPv2**.

11.5.- Instalación y Configuración de FreeRADIUS.

En primer lugar instalaremos el software que convertirá nuestra Raspberry en un servidor RADIUS:

```
[pi@raspberry]# sudo su
[root@raspberry]# apt-get install freeradius
```

Después lo configuraremos en tres pasos: A) configuración de equipos clientes, B) creación de cuentas de usuario (login/password) válidas en la autenticación, y C) protocolos a usar en el proceso de autenticación. Todos los archivos de configuración necesarios los encontraremos en "**/etc/freeradius**". En concreto, el archivo principal de configuración es "radiusd.conf", y en este se encuentran incluidos el resto mediante directivas "$INCLUDE", lo que facilita su gestión y configuración:

```
[root@raspberry]# more radiusd.conf | grep '$INCLUDE'
#  $INCLUDE line.
$INCLUDE proxy.conf
$INCLUDE clients.conf
$INCLUDE ${confdir}/modules/
$INCLUDE eap.conf
#       $INCLUDE sql.conf
#       $INCLUDE sql/mysql/counter.conf
#       $INCLUDE sqlippool.conf
$INCLUDE policy.conf
#       This next $INCLUDE line loads files in the directory that
$INCLUDE sites-enabled/
```

A) Indicaremos al servidor quienes van a ser sus clientes. En concreto, le indicaremos quienes van a ser nuestros puntos de acceso que van a solicitarle comprobación de la autenticación del usuario. Para ello editaremos en primer lugar el archivo "**/etc/freeradius/clients.conf**" y añadiremos lo siguiente:

```
[root@raspberry]# nano /etc/freeradius/clients.conf
#client dirección_IP_AP_Cliente {
#       secret  = Password_AP_acceso_servicio_RADIUS
#       shortname = alias_AP
#       nastype = other
#}
client 192.168.1.201 {
        secret  = passap1
        shortname = ap1
        nastype = other
}
client 192.168.1.202 {
        secret  = passap2
        shortname = ap12
        nastype = other
}
# Lo siguiente nos permitiría englobar a todos los AP bajo la misma clave de acceso al servicio
client 192.168.1.0/24 {
        secret  = passap s
        shortname = aps
        nastype = other
```

```
}
```

B) Daremos de alta cuentas de usuario con las que puedan validarse los usuarios de la red inalámbrica y así poder acceder al servicio. Aunque existen otro tipo de gestión de cuentas más eficiente (*p.e. Mediante una BD MySQL*), aquí usaremos la más simple: editar el fichero "**users**".

```
[root@raspberry]# nano /etc/freeradius/users
# <nombre_cuenta_usuario>       Cleartext-Password := <password_usuario>
#                               Reply-Message = "<mensaje personalizado>"
arturo          Cleartext-Password := "1234"
                Reply-Message = "Hola, %{User-Name}, Bienvenido!!"
usuw1           Cleartext-Password := "usuwireless1"
                Reply-Message = "Hola, %{User-Name}, Bienvenido!!"
```

C) Indicaremos el protocolo de autenticación. Como ya se ha dicho previamente tanto los equipos como dispositivos móviles que funcionan bajo Windows como GNU/Linux (Ubuntu, Debian, Android, etc.) soportan PEAP con **MSCHAPv2**. Para ello comprobaremos la configuración que se encuentra en "**/etc/freeradius/eap.conf**":

```
[root@raspberry]# nano /etc/freeradius/eap.conf
eap {
        ##default_eap_type = md5 | tls | ttls | ...
        default_eap_type = peap
        peap {
                default_eap_type = mschapv2
                ...
        }
        md5 { .. }
        tls { .. }
        ttls { .. }
}
```

Por último, reiniciaremos el servicio para que surtan efecto los cambios realizados en la configuración. También tenemos la opción de arrancar el servicio en modo "**debug**", ejecutando "**freeradius -X**", para comprobar su correcto funcionamiento mediante la herramienta "**radtest**" desde un equipo cliente, haciendo uso de la password "secret" que se indico en el archivo de configuración "**clients.conf**" (*la dirección IP del cliente debe figurar en "clients.conf"*):

```
[root@raspberry]# freeradius -X
[root@cliente]# radtest <nombre-usuario> <password>  <IP_Raspberry> <port> <secret>
[root@cliente]# radtest arturo 1234 dir_IP_Raspberry 1812 pass_secret
```

```
[root@raspberry]# /etc/init.d/freeradius restart
```

11.6.- Configuración del punto de Acceso.

En la configuración del punto de acceso deberemos indicarle los parámetros de configuración del servicio RADIUS que se soliciten: dirección IP de la Raspberry que hace de servidor RADIUS, el puerto por el que da dicho servicio, y el "secret" indicado en el fichero "**clients.conf**" para el punto de acceso en cuestión.

Ahora ya deberíamos poder acceder a la red inalámbrica mediante una autenticación previa.

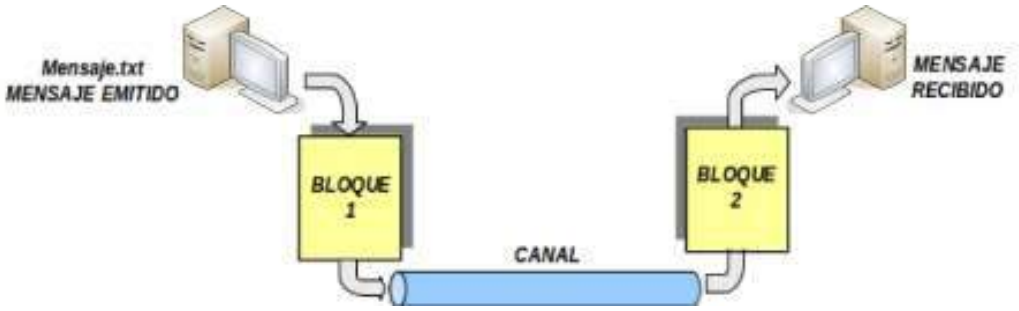
Apéndice A: Repaso al modelo de referencia TCP/IP. Direccionamiento IP.

En concreto, se desarrollarán los siguientes puntos:

1) Estudio del modelo de referencia implantado en comunicaciones informáticas actualmente denominado "**TCP/IP**", centrándonos en sus procesos más importantes.
2) Estructuración lógica de una red "**TCP/IP**". Estudiaremos la manera lógica de organizar los equipos de una red, formando redes y subredes.

A.1.- Introducción al Problema de la Comunicación

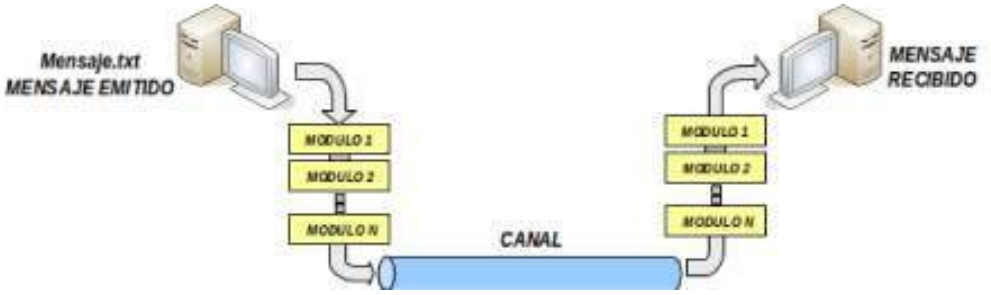
En este tema vamos a abordar/recordar el problema completo de comunicación entre ordenadores, desde el momento en que, a modo de ejemplo, abrimos una aplicación que nos permite escribir un correo electrónico, hasta que este mensaje se transforma en una señal eléctrica/óptica adecuada a las características de canal, y finalmente llega el correo escrito al destinatario, que puede estar, porque no, a miles de kilómetros de distancia del emisor.



Durante los primeros hitos de comunicación entre equipos informáticos remotos, el problema se abordaba como un único problema (*hardware + software*), resultando aplicaciones que daban solución desde la interfaz de usuario, hasta la transformación del mensaje a la forma adecuada del canal. Pronto se percataron del gran problema que esta solución conllevaba: **(1)** en el momento en que se detectaba un fallo cualquiera dentro de la cadena de acciones que se desencadenaban en el establecimiento o posterior puesta en marcha de la comunicación, conllevaba plantearse prácticamente desde un principio una nueva solución, ya que el fallo por pequeño que fuese afecta a prácticamente todo el conjunto (*software y hardware*), y **(2)** cualquier avance tecnológico que se producía en el hardware, provocaba debido a la falta de modularidad en el software, volver a tener que diseñar de nuevo los programas, impidiendo así su aprovechamiento.

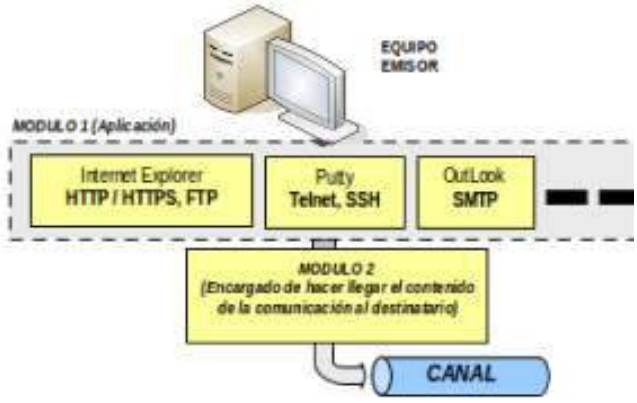
La solución adoptada fue bastante lógica: dividir el problema de comunicación en subproblemas independientes entre sí (*módulos, niveles, capas, etc.*), de tal forma, que la detección de un problema sólo afectase a una de las partes y no a todo el conjunto, garantizando al mismo tiempo la máxima independencia entre el hardware y el software utilizado: "**Divide y Vencerás**"

(*modularidad*). De esta forma, ante la necesidad de cambiar algún componente hardware/software, sólo el módulo afectado tendría que se substituido, sin afectar al resto.



A la hora de hacer la división en subproblemas, se tuvieron en cuenta las siguientes cuestiones:

1) Independientemente de la finalidad a la que se destine la aplicación en red: intercambio de ficheros, envió/recepción de correo electrónico, control remoto, etc., al final lo que va a ser enviado al canal va a ser simplemente un conjunto de bits con un significado que el destinatario se encargará de interpretar.



Esto se traduce, en que durante el proceso de comunicación podemos independizar la aplicación de la que hacemos uso de la estrategia a seguir a la hora de hacer viajar el contenido de la comunicación (los bits) por la red.

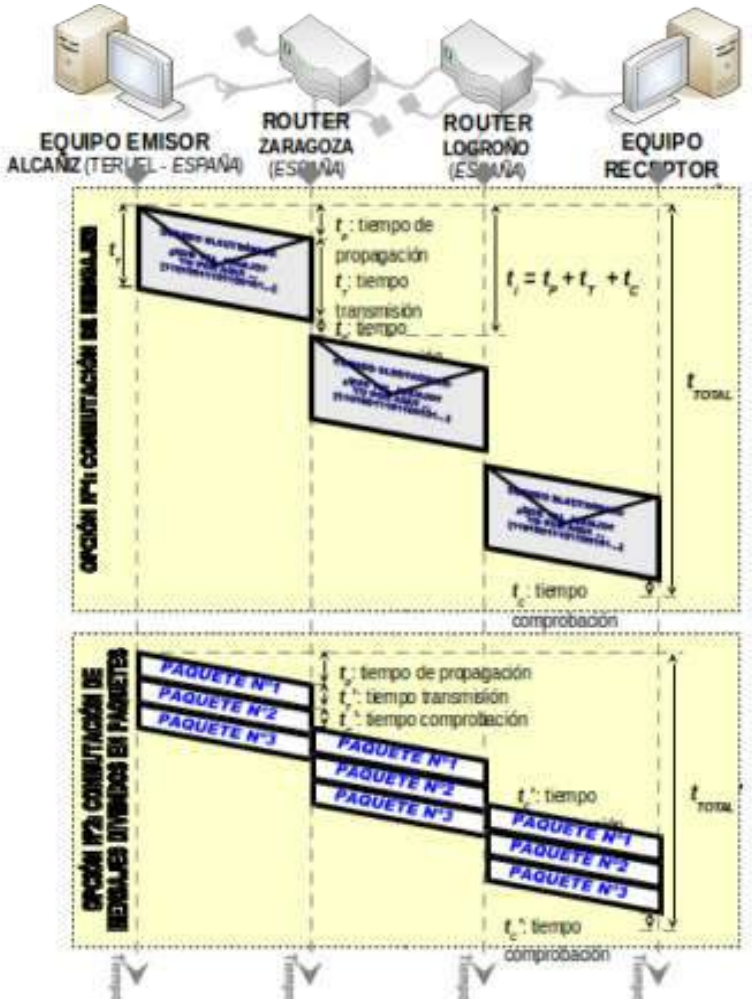
De esta forma, sin llegar a profundizar en el problema de la comunicación entre equipos informáticos, podemos subdividirlo en dos partes, módulos, niveles o capas bien diferenciadas: una relativa a la aplicación concreta de la que hace uso el usuario, y otra encargada de hacer llegar el contenido al destinatario. Esto significa, que cuando estamos haciendo uso de una aplicación que hace uso de la red, como puede ser "Intenet Explorer", "Microsoft Outlook", o "VNC", aunque su finalidad es totalmente diferente, lo único que cambia en ellas es el aspecto o interfaz que se le presenta al usuario, ya que la forma en que viajan por la red una petición de búsqueda en "Google", o un comando sobre una máquina remota mediante "VNC" es exactamente igual.

Teniendo en cuenta todo lo anterior, si lo que va a llegar al destinatario de la comunicación, no es más que una ristra de bits con un determinado significado, para que este sepa,

desde que aplicación fueron mandados, y mediante que aplicación deben ser tratados, se hace uso de un identificador de aplicación. Este identificador es necesario para que cada aplicación trabaje únicamente con los datos que le corresponda, ya que un sitio WEB solicitado desde un explorador WEB, no puede ser abierto por "Outlook", y viceversa, un correo electrónico no puede ser abierto por "Internet Explorer". A este identificador que indicará que aplicación debe tratar los datos que llegan a la máquina se denomina "**puerto**" de comunicación. De esta forma, tal como se explicará posteriormente, cada aplicación tiene asociado un número de puerto, en función de la capacidad de manejo de información que tenga, por ejemplo: "puerto 21", identificador que informa de que el contenido de la comunicación va destinado a una aplicación relacionada con la transferencia de ficheros (FTP) entre equipos remotos; "**puerto 80**", identificador que indica que el contenido de la información es "HTTP", etc.

En definitiva, esta primera división modular garantiza que a la hora de desarrollar una nueva aplicación que haga uso de la red, no nos tendremos que preocupar de los aspectos relacionados con la forma de hacer llegar el contenido de la comunicación al destinatario, sino solamente en desarrollar una atractiva interfaz de usuario que le resulte cómoda a éste.

2) Tal como se puede observar en la siguiente figura, se sabe de antemano que resulta más eficiente mandar el contenido de la comunicación dividido en paquetes independientes, que no mandarlo todo en un único mensaje.



Para advertir esta característica, a modo de ejemplo, se muestra una hipotética comunicación (*chat, correo electrónico, etc.*) entre dos equipos distantes (*Alcañiz-Burgos*), donde los "**ROUTER**" intermedios los podemos ver como dos cruces de caminos que nos encontramos en el camino, donde hay que tomar una decisión de que camino seguir con la finalidad de hacer llegar el mensaje al destinatario según un criterio establecido (coste, rapidez, fiabilidad, etc.). En concreto, se muestran las dos alternativas sugeridas: **(1)** Una primera opción donde los mensajes son mandados sin fragmentar, **(2)** y una segunda opción donde el mensaje, antes de ser enviado, es dividido en paquetes. Para comprender que la segunda opción es la idónea, y la que actualmente se utiliza, es necesario entender antes en primer lugar que representan cada uno de los tiempos involucrados en el proceso de envío de los datos:

A) Aunque no se muestra en el ejemplo anterior, antes de mandar el mensaje o datos de la comunicación, siguiendo un protocolo estipulado ó pasos regulados a seguir, en el 90% de los casos se hace necesario un establecimiento de la comunicación. Es decir, al igual que cuando deseamos realizar una comunicación telefónica, como primer paso, es necesario establecer la conexión entre emisor y receptor, y preparar el enlace para la comunicación, a lo cual se denomina establecimiento de la comunicación, en comunicaciones informáticas ocurre algo similar: descolgar teléfono, marcar números, comprobar que el destinatario esta operativo, advertir a este de que alguien quiere comunicarse con él, y en su caso aceptarla. En el argot informático, estas son las denominadas

comunicaciones "**orientadas a conexión**", ya que existen otro tipo de comunicaciones, en las cuales siguiendo otro protocolo diferente, puede establecerse una comunicación sin conexión previa ("**no orientadas a conexión**").

Esta conexión previa, o establecimiento de la comunicación, se traduce en un retardo inicial que en el caso de comunicaciones breves puede ser apreciable. En nuestro caso, para justificar la división del mensaje en paquetes antes de llevar a cabo la transferencia de los datos, asumiremos que el tiempo perdido en dicho establecimiento no es significativo.

B) "**t_P**" ó "**tiempo de propagación**", es el tiempo que le cuesta a un bit desde que sale desde nuestra tarjeta de red hasta que alcanza un destinatario. Este tiempo, sólo depende de dos factores: la distancia a recorrer, y la velocidad a la que se propaga la señal (eléctrica, electromagnética u óptica) por el medio:

t_P = distancia a recorrer / velocidad de propagación

Este tiempo, como puede advertirse según la expresión anterior, no depende del tamaño del mensaje, sino que únicamente depende del camino escogido (distancia a recorrer) para alcanzar el destino, y del medio o canal del que se hace uso, ya que dependiendo de si éste es un par trenzado, cable coaxial, fibra óptica, o el aire (comunicación inalámbrica), la velocidad de propagación cambia. Esto significa que este tiempo no es decisivo a la hora de decidir si sería más conveniente mandar el mensaje completo, o troceado en paquetes.

C) "**t_T**" ("**t_T'**") ó "**tiempo de transmisión**", es el coste temporal que conlleva el hacer llegar el mensaje o paquete, según la opción escogida (1) ó (2), a un nodo intermedio o destinatario. Este tiempo como es lógico depende del tamaño de unidad de datos (mensaje o paquete), y de la velocidad a la que se transfieren. Esta velocidad de transferencia, no hay que confundirla con la velocidad de propagación por el medio usada en el cálculo de "**t_P**". En este caso, dicha velocidad hace referencia a la rapidez con que la interfaz de red del equipo o del nodo correspondiente "saca" los bits al medio o canal. En el caso en que deseemos conocer el tiempo total de transmisión, será necesario tener en cuenta el número de nodos por los que se pasa.

$t_{T\ total}$ = (tamaño de la unidad de datos / velocidad de transferencia)·(nº nodos + 1)

Como es lógico, este tiempo al depender del tamaño de la unidad de datos, es dependiente de la opción escogida (1) ó (2).

D) "**t_C**" ("**t_C'**") ó "**tiempo de comprobación**". Una vez que el mensaje o paquete llega a un "cruce de caminos", el dispositivo encargado de decidir la vía de salida por donde expulsar el paquete, "**ROUTER**", para que así pueda llegar a su destino, debe comprobar antes si el paquete recibido es correcto o no. Es decir, tal como se vio en el capítulo anterior, existen múltiples estrategias de computo (algoritmos matemáticos) que permiten comprobar si el mensaje recibido no ha sufrido modificaciones, o lo que es lo mismo, si se han producido errores a lo largo de su transferencia (ruido, interferencias, etc.). Este cómputo, va a conllevar un nuevo retardo en el envío de la información que deberá tenerse en cuenta. En caso de que el "**ROUTER**" detecte algún error emitirá un mensaje de aviso al equipo emisor del mensaje para que vuelva a reenviarlo. También podría ocurrir, que los errores en bits se hayan producido sobre la dirección del equipo origen, ante

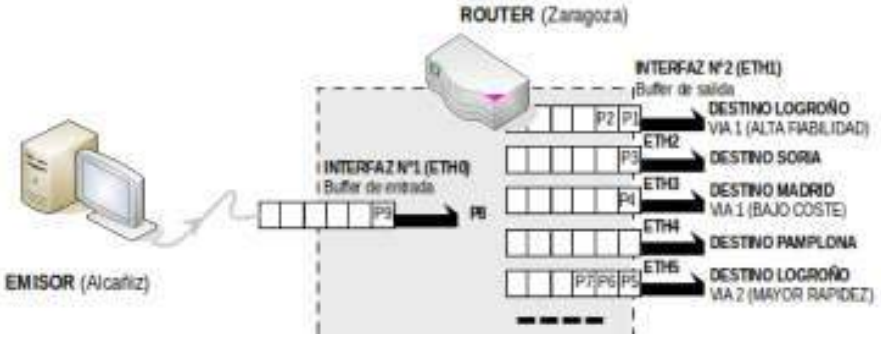
lo cual, el mensaje de aviso nunca le llegará. Ante esta situación, la solución escogida, es que el equipo emisor, una vez emitido el mensaje o paquetes que componen el mensaje, espera una respuesta por parte del destinatario, indicándole que el mensaje o paquetes que componen el mensaje, fueron recibidos correctamente. Este tipo de mensaje de confirmación se denomina "**ACK**" ("ACKnowledgment"). En caso de no recibir dicho "**ACK**", transcurrido un determinado tiempo, reenvía el mensaje o el paquete correspondiente que no haya sido confirmado.

Según lo anterior, puede advertirse que el tamaño de la unidad de datos (mensaje ó paquete) influye sobremanera en el "$\mathbf{t_C}$", ya que contra mayor es el tamaño del paquete, mayor cantidad de tiempo será invertido en el cómputo de comprobación del mensaje o paquete a la hora de detectar errores.

Además, también puede advertirse, que un "**ROUTER**" no puede reenviar el mensaje hacia su destino hasta que no haya sido comprobado que es correcto, lo que provoca tener que esperar "$\mathbf{t_T}$" hasta que ya este completo y dar paso a tal comprobación. Esto se traduce, en que el mensaje o paquete podrá ser reenviado a su destino, y alcanzar éste, en un tiempo que será tanto menor, contra menor sea el tamaño de la unidad de datos, o "$\mathbf{t_T}$". No obstante, cabe reseñar que en la actualidad, esto no es tan significativo, ya que los avances tecnológicos ya garantizan enlaces o líneas de comunicación muy fiables (fibra óptica), donde la tasa de errores es tan reducida, que no merece la pena ni comprobar si la unidad de datos es correcta del todo, comprobando únicamente la cabecera de éste. Esta estrategia es seguida por actuales tecnologías como "**ATM**" ("**Asynchronous Transfer Mode**" ó "Modo de Transferencia Asíncrono"), donde los mensajes a enviar son fragmentados o divididos en reducidos paquetes (llamadas "**celdas**") de 53 bytes de tamaño (5 bytes de cabecera, y 48 bytes de datos), de tal forma que al llegar estos a un nodo intermedio ó "**ROUTER**", este tan sólo comprueba la cabecera, pero no los datos. En caso de que los errores se produzcan en la zona de datos del paquete, no será detectado hasta que éste no llegue a su destino. Al llegar a éste, tratará de corregirlos siguiendo algún algoritmo matemático, y en caso de no resultar exitoso, remitirá al emisor del mensaje un mensaje para que vuelva a reenviar el paquete o "celda" defectuosa.

E) Además de los tiempos anteriores, de manera estricta, también sería necesario tener en cuenta un "**tiempo de espera en cola**". Es decir, una vez que el "**ROUTER**" ha comprobado que todo es correcto, según esté éste programado, pasará a decidir por cual de sus interfaces de salida deberá ser "expulsado" el mensaje o paquete para que pueda llegar a su destino, pero esto no será inmediato. Dependiendo de la velocidad y congestión de los enlaces puede quedar encolado ("cuello de botella") sobre el interfaz de salida en cuestión, y esperar a que tenga vía libre.

De igual forma, este cuello de botella, se puede producir por la interfaz de entrada al "ROUTER", conllevando un tiempo de espera añadido. Al igual que con el tiempo de establecimiento (y de finalización) de la comunicación, estos tiempos pueden ser apreciables, dependiendo de la comunicación concreta que se estudie, pero que asumiremos como no significativos en este estudio.



Según todo lo anterior, el tiempo invertido en una comunicación, en hacer llegar un mensaje a su destino se puede calcular según la siguiente fórmula (*siendo "**n**" el número de nodos intermedios y "n_{PAQ}" el número de paquetes en que es fragmentado el mensaje*):

Opción Nº1 (mensaje sin fragmentar): $T_{TOTAL} = t_P + t_T \cdot (n+1) + n \cdot t_C$
Opción Nº2 (mensaje dividido en "n_{PAQ}" paquetes): $T_{TOTAL} = t_P + n \cdot t_C + n_{PAQ} \cdot t_T + n \cdot t_T$

Concluyendo este apartado, la división de un mensaje en paquetes previamente a su transmisión es fundamental, ya que eso se traduce en una indiscutible eficiencia de la red, tanto desde el punto de vista temporal, como de aprovechamiento de los enlaces de comunicaciones, pudiéndose resumir en los siguientes puntos:

1º.- Contra menor es el tamaño de la unidad de datos que se envía menor es el tiempo de cómputo invertido por los nodos intermedios de la red ó "**ROUTER**'s".

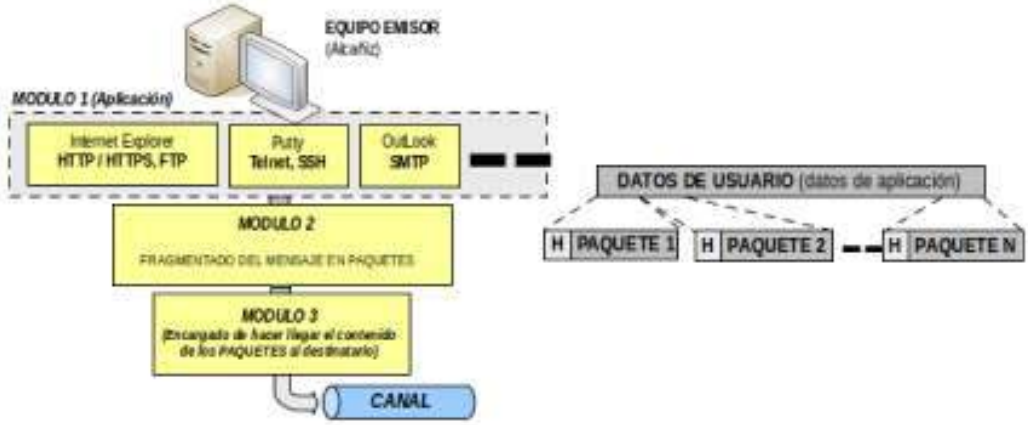
2º.- Para que un nodo intermedio, "**ROUTER**", pueda empezar a comprobar si se ha producido algún error en la transferencia de la unidad de datos (mensaje o paquete) que se envía, debe esperar a que este le llegue completamente. Esto se traduce, en que contra menor sea el tamaño de la unidad de datos, antes podrá comprobar el estado de estos, y en caso afirmativo, reenviarlo al siguiente nodo en dirección al equipo destinatario de la comunicación.

3º.- Contra menor es el tamaño de la unidad de los datos, menor es el coste invertido en el reenvío en el caso de que se detecte un fallo en el mensaje o paquete. Además, un menor tamaño de la unidad de datos se traduce, en el caso de enlaces o vías de comunicación poco fiables, que estas no se saturen tanto, ya que si además de enviar paquetes de gran tamaño, estos hay que reenviarlos con frecuencia, en conjunto se traduce en un gran consumo de ancho de banda.

Todo lo anterior ha causado que en la actualidad, la tecnología utilizada en comunicaciones digitales entre equipos informáticos sea la denominada "**conmutación de paquetes**", nombre que viene a indicar que lo que viaja por la red no son más que fragmentos de mensajes más grandes (paquetes), y que éstos son conmutados (forma de establecer el camino) mediante nodos intermedios. Antiguamente se hacía uso de la denominada "**conmutación de mensajes**", tecnología bajo la cual los mensajes no eran fragmentados para su transmisión, y aunque se siguió utilizando en casos particulares (comunicaciones no orientadas a conexión, o en situaciones donde los mensajes no eran extremadamente grandes), ha quedado en un total desuso.

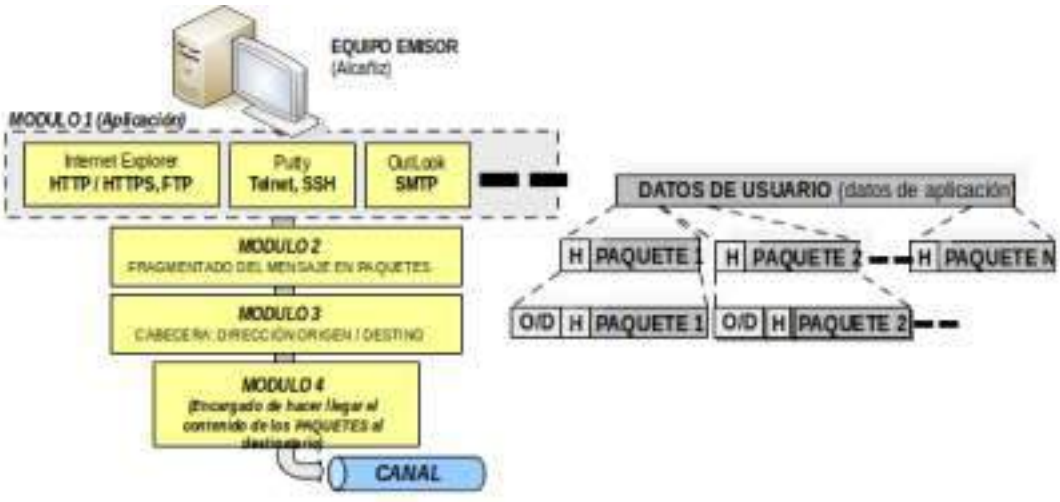
Volviendo al origen de la discusión, según lo visto en este segundo punto, independientemente de la aplicación de la que se haga uso (mensajería electrónica, control remoto de equipos, videoconferencia, etc.), e independientemente del medio y características de este, el

problema de comunicación podría volverse a dividir en un nuevo módulo que se encargará de fragmentar en paquetes los mensajes que forman parte de la comunicación.



Tal como ya se ha expuesto en el punto anterior, con la finalidad de que el origen/destino distinga que aplicación origino el contenido del mensaje, y así poder tratarlo, se hace uso de un identificador o número de puerto. El problema ahora es que sobre una aplicación pueden establecer múltiples sesiones, y por tanto, diferentes comunicaciones, con lo cual podrían mezclarse paquetes de diferentes sesiones establecidas. Para que esto no ocurra, al mismo tiempo que se fragmentan, se añade una información a los paquetes (*cabeceras ó "**headers**", "**H**"*) que permita su reensamblado en el destino: número de secuencia, número de puerto de origen y destino, etc. Todos estos aspectos se verán más en profundidad posteriormente mediante la arquitectura "**TCP/IP**".

3) Una vez que estamos haciendo uso de una aplicación, y los datos generados por esta son troceados en paquetes o "celdas", ahora sería conveniente indicar quien es el destinatario y remitente. Es decir, al igual que cuando escribimos una carta, tras redactarla (datos de aplicación) e introducirla en un sobre, escribimos el destinatario y remitente para que el cartero ("ROUTER") sepa que hacer con ella para hacerla llegar a su destino (en caso de problemas, remitirla al remitente), en comunicaciones informáticas, deberemos coger cada uno de los paquetes y añadirles mediante una nueva cabecera datos tales como el origen y el destino.



Tal como veremos posteriormente a lo largo del presente tema ó capítulo, en el caso de que

el modelo de comunicaciones utilizado sea el "**TCP/IP**", estas direcciones a incluir en la cabecera son las denominadas direcciones IP, las cuales estudiaremos en profundidad, con la finalidad de poder hacer un buen uso lógico de ellas (problemas de implementación lógica de redes y subredes).

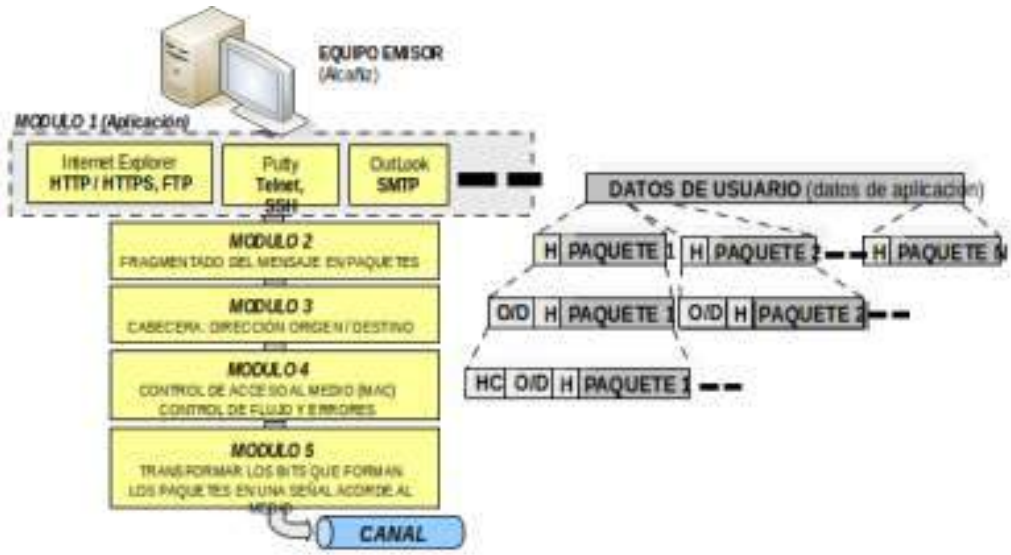
4) Conocida ya la dirección de destino de cada uno de los paquetes, tan sólo cabe pensar como mandar éstos por el medio de transmisión o canal de comunicaciones.

En la mayor parte de las ocasiones, el equipo emisor/receptor suele formar parte de una red de difusión donde el medio de comunicaciones es compartido. En estos casos, será necesario establecer un "**control de acceso al medio**" ("**MAC**") para evitar las ya comentadas colisiones (capítulo o tema 0), existiendo multitud de estrategias para ello, siendo la más utilizada "**Ethernet**" (**IEEE 802.3**). Esto permitirá que en cada momento sólo transfiera datos por el canal uno de los equipos de la red de difusión (si transmitiesen más de uno, se solaparían las señales y se produciría una colisión), y que esta transferencia sea equitativa para todos los equipos que la forman (reparto equitativo del ancho de banda de la red).

Junto con el "**MAC**", también es conveniente llevar a cabo un "**control de flujo y errores**", tanto por parte del emisor como del receptor.

En cuanto al "**control de flujo**", reseñar que consiste simplemente en controlar la velocidad de transmisión por parte del emisor, mediante un dialogo con el receptor, con la finalidad de evitar cuellos de botella. Es decir, es posible que durante el transcurso de una comunicación, se produzcan cambios en las características de los enlaces (cambios de ruta por congestión ó caída de líneas, cambios medio ambientales, etc.), que hagan que la velocidad del enlace tenga que disminuir (ó pueda aumentar) con la finalidad de evitar la saturación de las líneas de comunicación, evitando de esta forma la perdida de paquetes (cuando un "**ROUTER**" se satura, se pone a descartar paquetes sin contemplación), y el consecuente reenvió de estos, que no haría más que agravar el problema.

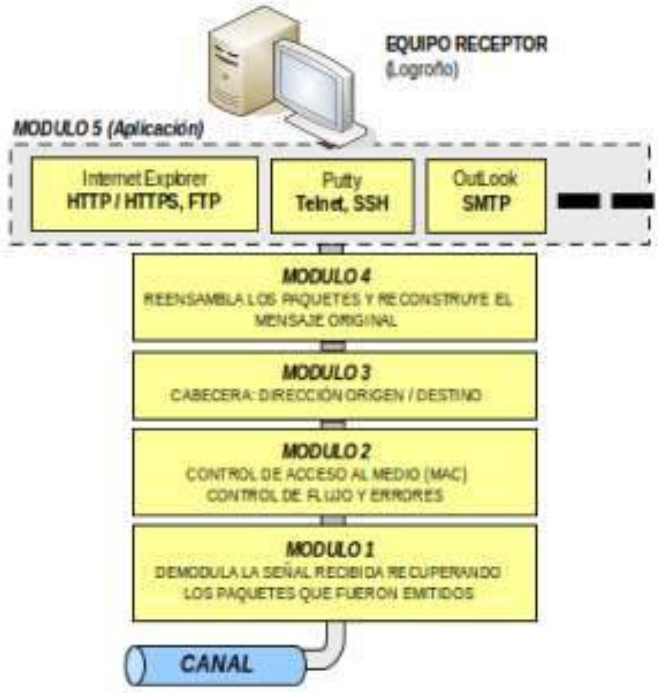
Respecto al "**control de errores**", con la finalidad de que los nodos intermedios, "**ROUTER**'s", y equipo destinatario, puedan comprobar, y en caso de ser posible, corregir, los posibles errores que se hayan producido durante la transmisión de los paquetes a través del medio, se añaden por parte del equipo emisor a través de una nueva cabecera un conjunto de bits ("**Checksum**") que servirán para tal finalidad.



En definitiva, en este nuevo módulo en el que podemos dividir el problema de la comunicación entre equipos informáticos (**divide y vencerás**) se concentran por tanto todas aquellas tareas relacionadas con el control de la comunicación: control de acceso, flujo y errores. En la realidad, siguiendo el modelo "**TCP/IP**", comprobaciones como la de errores, se hace prácticamente en todos los módulos, niveles o capas que la forman, y control de flujo puede llevarse a cabo simultáneamente por uno o varios niveles.

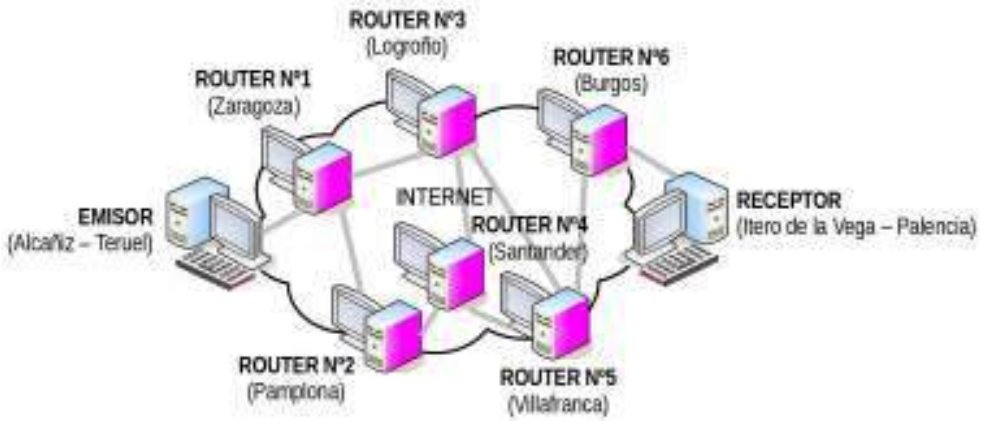
5) Una vez que se han seguido todos los pasos anteriores, ahora simplemente habrá que transformar los bits que forman parte de cada uno de los paquetes en una señal acorde al medio (par trenzado, cable coaxial, aire ó fibra óptica), proceso al cual se suele denominar modulación de la señal. Es decir, modular puede traducirse en dar forma a una señal (eléctrica, electromagnética u óptica, dependiendo del medio de transmisión) en función de la información. Esto permite que el receptor al recibir la señal enviada por el emisor, observando su forma, pueda interpretar los datos o información que la originaron, y por tanto, recuperar el paquete de bits que fue emitido.

Antes de acabar con esta larga introducción a este importantísimo capítulo, cabría reseñar que las figuras o imágenes mostradas a lo largo de ésta han hecho referencia únicamente al emisor del mensaje, sobreentendiendo que el receptor vuelve a hacer los mismos pasos pero de manera inversa: **(1)** A través de la forma de la señal que le llega, la demodula, y recupera el mensaje enviado, **(2)** después comprueba si se han producido errores en la transmisión, y en caso de ser necesario, ajustar el flujo de la transferencia, **(3)** se asegura de que el destinatario del mensaje es él, **(4)** reensambla todos los paquetes a través de la información del número de secuencia que llevan en su cabecera y el puerto de origen, **(5)** y por último, en función del puerto de destino, conducen el flujo de información a la aplicación adecuada para que pueda tratarla.



Ej.A.1.I Comparativa entre Conmutación de Mensajes y Paquetes

Dada la red que se muestra en la siguiente figura, determinar la diferencia temporal que existe entre utilizar la técnica de conmutación de mensajes, o conmutación de paquetes, en el caso de que se desee transferir una canción en formato "mp3" de tamaño 7.8MBytes, entre "Alcañiz" (Teruel - ESPAÑA), e "Itero de la Vega" (Palencia - ESPAÑA): Alcañiz > Zaragoza > Logroño > Villafranca > Burgos > Itero de la Vega, con un distancia total del enlace de unos 500 Km.



De cara a los cálculos asumiremos los siguientes datos:

(a) Tasa de transferencia a lo largo de todo el enlace en promedio es de 100Kbps.
(b) Velocidad de propagación: c≅300.000 Km/sg (asumimos como canal fibra óptica).
(c) Distancia total del enlace: d=500Km.
(d) Tamaño del mensaje: 7,8MBytes (fichero MP3).
(e) Cabecera del mensaje: 6 bytes (48 bits).
(f) Tamaño de los paquetes: 5 bytes de cabecera y 48 bytes de datos (424 bits en total).
(g) Tiempo de almacenamiento, procesamiento en el nodo (t_C), y reenvió, los asumiremos despreciables.
(h) Tiempo de establecimiento de la comunicación: 0.3 segundos.
(i) Tiempo de liberación de la comunicación despreciable.
(j) Tiempos de espera en cola despreciable.
(k) El numero de nodos intermedios o ROUTER's por los que se pasa es 4 (n=4).

Solución Ej.A.1.I.i: Conmutación de Mensajes frente a la de Paquetes

A) Opción Nº1: Conmutación de mensajes.

En esta primera opción asumiremos que mandamos el mensaje sin trocear. En este caso, el tiempo total invertido en la transferencia de fichero MP3 sería:

$$t_{TOTAL}=t_P+n\cdot t_C+(n+1)\cdot t_T = 5\cdot(\text{tamaño total del mensaje/velocidad de transferencia}) =$$
$$= 5\cdot(7{,}8\cdot1024\cdot1024 \text{ bits} / 100.000 \text{ bits/sg}) = 408{,}94 \text{ sg} = \mathbf{6{,}8 \text{ minutos.}}$$

Donde el tiempo de propagación, distancia/velocidad de propagación (500Km / 300.000Km/sg) resulta ser igualmente despreciable.

B) Opción Nº2: Conmutación de paquetes (tamaño del paquete 424 bits)**.**

Suponiendo que se trata de una comunicación orientada a conexión, donde como ya se ha explicado a lo largo de anterior introducción, requiere de un establecimiento de la comunicación previo, y una posterior liberación de esta, el cálculo resultaría el siguiente (se desprecia la cabecera del mensaje en el cálculo de su tamaño total, por considerarse este despreciable respecto al tamaño de los datos, 48 bits frente a $7{,}8 \cdot 1024^2$ bits):

$$t_{TOTAL}=t_{EST}+t_P+n \cdot t_C+n_{PAQ} \cdot t_T+ n \cdot t_T+t_{LIB}=$$

= 0,3 sg + (tamaño total del mensaje/tamaño del paquete)·(tamaño paquete/velocidad de transferencia) + 4·(tamaño paquete/velocidad de transferencia) = (7,8·1024·1024 bits / 424 bits)·(424 bits / 100.000 bits/sg) + 4·(424 bits / 100.000 bits/sg) = 81,8 sg = **1,36 minutos**.

Como puede advertirse, la diferencia es "bestial", lo que viene a cerciorarnos lo que ya hemos estado recalcando a la largo de toda la introducción del tema: la clave esta en que mientras en la conmutación de mensajes debemos esperar en cada nodo intermedio, la llegada de todo el mensaje, para su posterior comprobación y reenvío, en la conmutación de paquetes, una vez comprobado un paquete, se puede reenviar, sin la necesidad de tener que esperar a los demás paquetes que forman parte del mensaje, es decir, puede darse el caso de que parte de los paquetes hayan sido recibidos por el destinatario, mientras otros aún siguen siendo comprobados por los nodos intermedios.

Aunque no se ha comentado a lo largo de toda la introducción, como es lógico, el tamaño del paquete es crucial de cara a optimizar el resultado. Se puede comprobar, aunque no va a desarrollarse aquí, que el tamaño ideal para toda comunicación podría calcularse a través de la siguiente ecuación:

$$Tamaño_{PAQ\ OPT}=\sqrt{(m \cdot H/n)} + H$$

Donde "m" el tamaño total del mensaje a enviar, "n" el número de nodos intermedios por los que se pasa, y "H" el tamaño de la cabecera del paquete.

Aclaración: Matemáticamente se sabe que derivando la función tiempo total (T_{TOTAL}) respecto al tamaño del paquete, e igualando esta primera derivada a cero, se obtiene el valor de la variable respecto a la cual se ha derivado (el tamaño del paquete), que hace la función mínima: $\mathbf{dT_{TOTAL}/dTamaño_{PAQ}=0}$.

A.2.- Modelos de Referencia en Comunicaciones Informáticas.

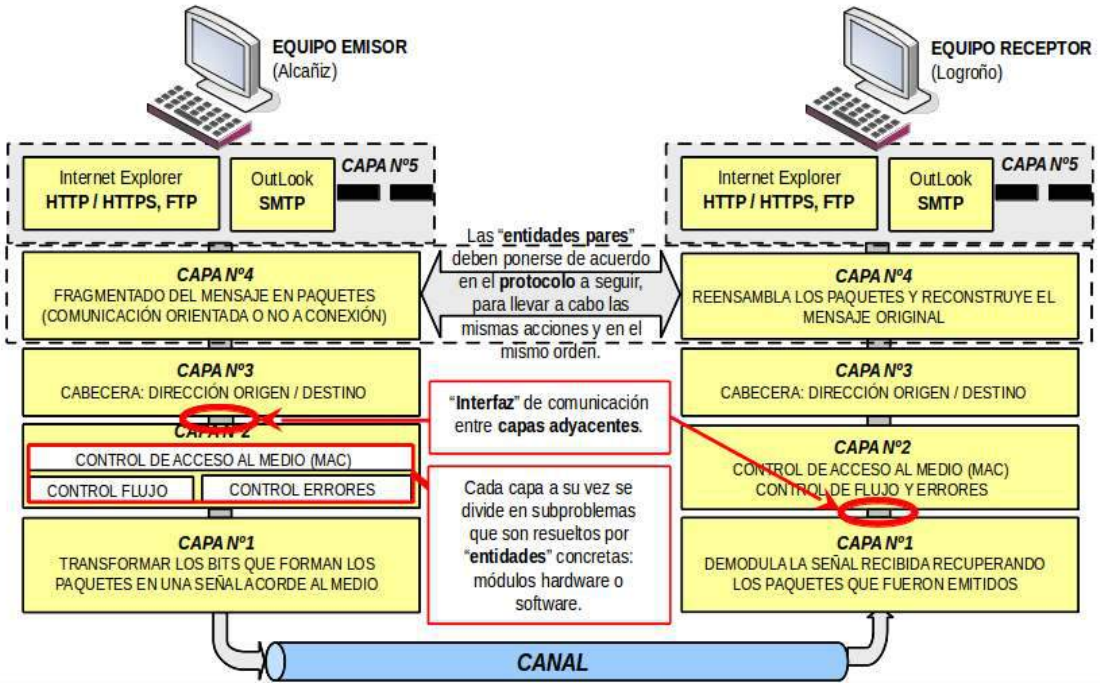
Tal como se ha tratado de explicar en el apartado introductorio anterior, el problema de comunicación entre equipos informáticos puede dividirse en subproblemas independientes, que posibilita el poder resolverlos de manera separada sin que se interfieran, facilitando su desarrollo (*es más fácil solucionar varios problemas cortos, que un problema más grande y complejo*) y garantizando un mantenimiento a posteriori más cómodo y versátil. En concreto, se ha planteado

una posible solución al problema de la comunicación mediante la división de éste en 5 bloques (*módulos, niveles ó capas*), pero podría pensarse otras alternativas con un número de divisiones ó módulos diferentes. Esta disparidad en las posibles soluciones, provoco en los inicios de las comunicaciones en red de computadores, una confusión como consecuencia de las múltiples alternativas existentes al mismo problema, y aunque hacían uso de algoritmos similares, estos se llevaban a cabo en distinto orden (*una estructuración en niveles diferentes*).

Con la finalidad de llegar a un consenso ante este problema que se había planteado, se establecieron una serie de modelos a seguir, denominados "**modelos de referencia**". En concreto, dos de ellos han cobrado un especial interés, el modelo "**OSI**" y el modelo "**TCP/IP**", de los cuales, el segundo de ellos, es el que actualmente esta implantado, y que ya se ha adoptado como un estándar "abierto".

La primera arquitectura de redes comercial fue presentada por IBM en 1974, y se denominó "**SNA**" ("**Systems Network Architecture**"). "**SNA**" se basa en la definición de una serie de módulos, niveles ó capas, donde cada uno de ellos ofrece sus servicios al siguiente, que los utiliza para implementar los suyos propios, y así sucesivamente. Es decir, tal como hemos visto en la introducción, el trabajo a realizar se distribuye entre las distintas capas, complementándose entre sí. "**SNA**" sigue todavía vigente en los entornos mainframe IBM, aunque en la actualidad se están llevando a cabo migraciones hacia redes "**TCP/IP**". Pero más importante que su relativa actualidad, su importancia reside en el modelo de capas que utiliza, el cual ha sido la base de todas las arquitecturas de redes actuales, incluidas tanto el modelo "**OSI**", como el "**TCP/IP**".

Las ideas básicas del modelo de capas son las siguientes:



(A) La capa "n" ofrece una serie de servicios a la capa "n+1" que los puede usar para implementar los suyos propios.

(B) Los elementos activos de cada capa se denominan entidades. Una entidad puede ser

software o hardware.

(C) Las entidades de la capa "n" en el equipo emisor y receptor (entidades pares), deben ponerse de acuerdo en las acciones a llevar a cabo, y en que orden. Es decir, tienen que seguir un protocolo previamente definido para dicho nivel. Esto es importante, ya que tal como se ha visto en la introducción, en el receptor, se tienen que establecer exactamente los mismos pasos, pero de manera inversa a como fueron llevados a cabo en el emisor, para devolver el mensaje a su estado original. Si en alguna de las capas o niveles se saltasen el protocolo, y llevarán a cabo las acciones en otro orden diferente, la fiabilidad de la comunicación no estaría garantizada.

(D) La comunicación entre dos capas adyacentes ("n" y "n+1") debe realizarse a través de un interfaz. Éste es el que oculta la complejidad de las acciones llevadas por una capa, garantizando la independencia (baja cohesión) entre las funciones realizadas por los módulos ó capas, definiendo al mismo tiempo, qué servicios va a ofrecer una capa a la otra.

(E) Gracias a la interfaz anterior, se nos permite sustituir la funcionalidad completa de una capa, sin que el resto de capas se entere, siempre y cuando, se respete como es lógico, las características de la interfaz y servicios ofrecidos, además del protocolo a seguir.

Un conjunto funcional concreto de capas y protocolos entre entidades pares forman una arquitectura de red. Una arquitectura de red queda perfectamente especificada cuando se enuncian las capas que lo componen, sus interfaces y la pila de protocolos que usa. En la figura anterior se puede observar una arquitectura de red de comunicaciones pentacapa (5 capas o niveles) que comunica un equipo emisor y otro receptor, donde cada una de las capas que la forman a su vez se divide en distintas entidades que llevan a cabo las tareas (protocolo) previo acuerdo con sus entidades homologas (entidades pares).

A el conjunto de protocolos que utiliza un determinado sistema se denomina pila de protocolos ("Protocol Stack"), de hay que sea frecuente oír hablar de la pila de protocolos "**OSI**" ó "**TCP/IP**".

A.3.- Modelo de Referencia OSI.

Como ya se ha comentado en el punto anterior (3.2), con la finalidad de evitar confusiones a los clientes de las redes de comunicaciones (en las primeras redes cada fabricante establecía sus propias normas de conexión y transmisión), las organizaciones internacionales de normalización establecieron una serie de modelos que sirvieran de referencia a los fabricantes. De entre lo distintos modelos propuestos destacó el modelo "**OSI**" ("**Open Systems Interconnection**", "Interconexión de Sistemas Abiertos"), especificado por la organización "**ISO**" ("**International Organization for Standardization**", "Organización Internacional para la Normalización o Estandarización"). A partir de 1977 la "**ISO**" fue definiendo una arquitectura de redes, "**OSI**", con el fin de promover la creación de una serie de estándares que especificaran un conjunto de protocolos que fueran independientes del fabricante que lo implementara, evitando de esta forma la dependencia del cliente con un fabricante concreto, y favoreciendo así la competitividad. De esta forma, con "**OSI**" la "**ISO**" además de facilitar las comunicaciones entre sistemas diferentes, pretendía impedir que ninguna de las arquitecturas de fabricante existentes adquiriese una posición hegemónica, especialmente "**SNA**" de IBM.

La aportación más importante del modelo "OSI" ha sido la definición teórica de su modelo arquitectónico de red. Este modelo ha servido como marco de referencia para describir y estudiar redes existentes, de hay que se le denomine "modelo de referencia OSI". En concreto, el modelo

"OSI" esta formado por siete capas ó niveles (igual que "SNA"):

(1) Capa Física. Esta ligada con la transmisión de los bits. Es la capa de más bajo nivel, y se encarga de adecuar la señal de información (modular/demodular) a las características del canal, junto con el transporte del flujo de bits entre emisor y receptor a través del medio de transmisión (pares trenzados, fibra óptica, etc.). Por tanto, tal como vimos en el capítulo anterior, se encargará de funciones tan importantes como la "codificación de canal", la modulación/demodulación de la señal en función de la información a transmitir, multiplexación de señales, etc. A este nivel se especifican todos los aspectos eléctricos, conectores y características de los medios de transmisión.

(2) Capa de Enlace. Se corresponde con la segunda capa más baja, y tal como se vio en el ejemplo introductorio, se encarga de funciones de control: garantizar una transmisión fiable libre de errores (control de errores), para que el resto de capas (capas superiores) puedan manipular la información que ha llegado sin el miedo de que ésta sea errónea, además de llevar a cavo un control de flujo para de adecuar ó ajustar la velocidad de transmisión de los datos en función de las características instantáneas de la red, y establecer un control de acceso al medio compartido. Debido a que existen tres funciones básicas he independientes dentro de este nivel, (a) control de flujo, (b) control de errores y (c) control de acceso al medio, normalmente se suele subdividir en dos subcapas: subcapa "**MAC**" ("**Medium Access Control**", "Control de Acceso al Medio") y subcapa "**LLC**" ("Logical Link Control", control de flujo y errores).

(3) Capa de Red. Se encarga de llevar a cabo funciones como puede ser la identificación de paquetes y su "routing" ó enrutamiento (ante un "cruce de caminos" decide que salida tomar) a través de la red. Además, si se cree conveniente, en función de las características de la red, puede segmentar los paquetes en trozos más pequeños, ya que tal como vimos en la introducción, esto puede redundar en un menor retardo de comunicación, y mayor aprovechamiento del ancho de banda. En ocasiones, en este nivel puede llevarse a cabo por parte de los ROUTER un "control de congestión", distribuyendo los paquetes que forman parte de la comunicación a través de distintos caminos, en caso de ser necesario (se llevan a cabo conteos y estadísticas), con la finalidad de evitar la saturación de ciertos enlaces.

(4) Capa de Transporte. Aunque su función esencial es la del troceado/ensamblado de los paquetes en que se divide la información, y garantizar su secuencialidad, lleva a cabo otra serie de funciones como el control de errores (no a nivel de bit), controlando los paquetes que son recibidos, y en caso de detectar el fallo de alguno, garantizar su retransmisión. En ocasiones también lleva a cabo funciones de control de flujo, multiplexación (varias sesiones multiplexadas en una misma comunicación), e identificación (tipo de aplicación que envía o debe recibir la información) a través de un número de puerto.

(5) Capa de Sesión. Se encarga de llevar a cabo un control de las sesiones que se establecen sobre los servicios ofrecidos en la red (HTTP, FTP, etc.), y emular una comunicación duplex. En esta capa también se llevan a cabo funciones de sincronismo, que permiten restablecer una comunicación en caso de que se produzca algún corte. También tiene la posibilidad de establecer prioridades a las distintas sesiones establecidas.

(6) Capa de Presentación. Define el formato de los datos que se van a intercambiar entre las aplicaciones, y ofrece a los programas de usuario (aplicaciones) un conjunto de servicios

relacionados con la transformación de los datos: compresión de los datos (garantiza un mayor aprovechamiento del ancho de banda, aunque dicha compresión puede provocar perdida de información) y su cifrado (codifica la información garantizando confidencialidad). En definitiva, se encarga de adaptar los datos a la forma en que serán tratados por la máquina.

(7) Capa de Aplicación. Proporciona un medio a los programas de aplicación para que accedan al entorno de comunicaciones "OSI", garantizando una transparencia de ello al usuario a través de una interfaz de usuario, que aunque normalmente es "GUI" ("Interfaz Gráfica de Usuario"), puede ser igualmente "LUI" ("Interfaz de Usuario de Línea de comandos").

Aunque como se puede ver, se trata de un modelo de referencia muy bien definido, en algunos puntos es cuestionable y redundante (se llevan a cabo funciones similares en distintas capas). También se le suele achacar que no existe un equilibrio entre las funciones que realizan las capas, estando alguna de ellas sobrecargada, y otras apenas hacen nada. No obstante, el error mayor que cometieron sus diseñadores, es que tardaron mucho tiempo en sacarlo a la luz, lo que provoco que otros modelos como es el "TCP/IP" se le adelantará en su implementación y se extendiera su utilización, desembocando en que el modelo "OSI" apenas se utilizase.

A.4.- Modelo de Referencia TCP/IP.

En 1969 la agencia norteamericana de proyectos de investigación avanzada ("**ARPA**", "**Advanced Research Projects Agency**") recibió el encargo de proyectar un sistema de interconexión de ordenadores que debía poder resistir la destrucción de parte de su infraestructura a causa de un posible ataque, de forma que dos nodos cualesquiera pudieran seguir comunicados siempre que hubiera alguna ruta que los uniera. La "ARPA" decidió que la red debía usar conmutación de paquetes y topología mallada (múltiples alternativas para poder ir de un origen a un destino), tras lo cual procedió a su licitación. Poco a poco, las empresas y universidades participantes en el proyecto fueron creando lo que se denominó "**ARPANET**", una red altamente fiable y robusta. La "ARPANET" fue creciendo paulatinamente, y pronto se hicieron experimentos utilizando otros medios de transmisión de datos, en particular enlaces por radio y vía satélite. En relación a los protocolos existentes, se detecto que no satisfacían las necesidades esperadas, por lo que se diseñó un nuevo conjunto o pila de protocolos, y con ellos una nueva arquitectura de red. Esta nueva arquitectura recibió el nombre "**TCP/IP**" ("**Transmisión Control Protocol**/**Internet Protocol**"), procedente de los dos protocolos más importantes que formaban la pila de protocolos: el protocolo "TCP" utilizado por las entidades pares de nivel 4, e "IP" por las entidades pares de nivel 3. "ARPANET" creció exponencialmente, sobre todo tras su fusión con la red de la fundación nacional de la ciencia de los Estados Unidos ("NSFNet"). Poco a poco se la fue conociendo por el nombre actual: "**Internet**".

Los diseñadores de "TCP/IP" actuaron de manera más pragmática que los del modelo "OSI". Mientras que en el caso de "OSI" se emplearon varios años en definir con sumo cuidado una arquitectura de capas donde la función y servicios de cada una estaban perfectamente definidas, y sólo después se planteó desarrollar los protocolos para cada una de ellas, en el caso de "TCP/IP" la operación fue a la inversa: primero se especificaron los protocolos, y luego se definió el modelo como una simple descripción de los protocolos ya existentes. Por este motivo el modelo "TCP/IP" es más simple que el modelo "OSI". No obstante, "OSI" es una especificación más coherente y académica por lo que se utiliza a menudo para describir otras arquitecturas, como por ejemplo la

"TCP/IP", mientras que el modelo "TCP/IP" nunca suele emplearse para describir otras arquitecturas que no sean la suya propia.

A continuación se describirá en mayor detalle el modelo de red más ampliamente utilizado en el mundo, aunque trataran de omitirse los aspectos de más enjundia, y que no se consideren relevantes para la comprensión de la solución que se adoptó.

A.5.- División en niveles del Modelo de Referencia TCP/IP.

En cuanto a su división en niveles ó capas, cabría reseñar que dependiendo de autores/docentes, hay quien lo divide en 5 capas, mientras otros, tras agrupar las dos capas más bajas del modelo, lo dividen en 4 capas. Aquí daremos por buenas ambas divisiones, ya que de cara a la explicación teórica del modelo, se cree más conveniente una mayor división del problema, lo que ayudará a comprender de una forma más pausada la cadena de acciones que se llevan a cabo, por lo que se hará uso de 5 niveles, mientras que desde el punto de vista práctico y de implementación, asumiremos como válida la división en 4 niveles.

Por tanto, desde un punto de vista teórico vamos a asumir que el modelo "TCP/IP" esta dividido en 5 niveles, que mantienen una correspondencia con el modelo de referencia "OSI" de la siguiente forma:

¡¡Aclaración!! Aunque según el estándar (OSI y TCP/IP) el estudio de las capas, y su numeración, se hace de abajo hacia arriba (física a aplicación), en este capítulo por cuestiones docentes se va a comenzar el estudio del modelo TCP/IP al revés. Se cree más razonable empezar su estudio por la capa que está más próxima al usuario (capa de aplicación), que no por la capa más distante y más compleja para su comprensión (capa física), tal como se hizo en la introducción del capítulo.

1) Nivel de Aplicación: Se corresponde con las capas, y por tanto, con las funciones ya comentadas de las capas de aplicación, presentación, y buena parte de la capa de sesión del modelo de referencia "OSI".
2) Nivel de Transporte: Se corresponde con la capa de transporte, y alguna de las funciones de la capa de sesión del modelo "OSI".
3) Nivel de Red: Se corresponde con la capa de red del modelo "OSI".
4) Nivel de Enlace: Se corresponde con la capa de enlace del modelo "OSI".
5) Nivel Físico: Se corresponde con la capa física del modelo "OSI".

Aunque las funciones básicas de las distintas capas ó niveles del modelo "TCP/IP" fueron descritas en el punto introductorio (*la división en niveles sugerida se corresponde con la estructura y funcionalidad a grandes rasgos del modelo "TCP/IP"*) de este capítulo, a continuación se detallarán por separado las características más interesantes que presenta cada uno de los niveles.

Antes cabría reseñar que existen una serie de documentos donde se detalla prácticamente todo lo relacionado con la tecnología de la que se sirve Internet (*protocolos, recomendaciones, comunicaciones, etc.*) conocida como "**RFC**" ("**Request For Comments**"), colección que comprende más de 3000 documentos. En concreto, todo el estándar "TCP/IP" esta publicado en "**RFCs**" ("**www.rfc-es.org**"). Entre otras cosas, estos "RFCs" están hechos para hacer compatibles los distintos programas entre sí, lo que nos permite ofrecer un mismo servicio, mediante el uso de diferentes aplicaciones. Aunque cualquiera puede proponer una recomendación "RFC", el "**IETF**" ("**Internet Engineering Task Force**", "Grupo de Trabajo en Ingeniería de Internet") es una de las principales fuentes, actuando en diversos aspectos, tales como transporte, encaminamiento ó seguridad.

En relación a estas "RFCs", se pueden clasificar en cinco grupos:

1) **Required** (requerido). Debe ser implementado en todas las máquinas que ejecuten "TCP/IP" inclusive los "gateway" y "routers".
2) **Recommended** (recomendada). Se estimula el que todas las máquinas que ejecuten TCP/IP utilicen esta especificación de la "RFC". Las "RFC" recomendadas, normalmente están siendo utilizadas en todas las máquinas.
3) **Elective**. El uso de esta "RFC" es opcional. No es ampliamente usada.
4) **Limited Use** (uso limitado). No esta pensada para un uso general.
5) **Not recommended** (no recomendada). No está aconsejada su uso.

Si uno de estos documentos "RFCs" comienza a ser considerado como un estándar, comienza a pasar por los diferentes estados de desarrollo, prueba y aceptación. Durante este proceso, estos procesos son formalmente llamados "maturity levels" (niveles de maduración). Hay tres niveles de maduración en los estándares de Internet:

A) Proposed Standard (propuesta). Una especificación de propuesta, es generalmente estable, ha resuelto las conocidas alternativas de diseño, está bien comprendida, ha recibido el visto bueno de la comunidad y parece un buen candidato a ser evaluado por la comunidad.

B) Draft Standard: (borrador). Un borrador, debe ser entendido y reconocido como estable, tanto semánticamente como su base para poder ser desarrollada correctamente.

C) Internet Standard: El estándar Internet, (muchas veces nos referimos a él como un 'estándar' simplemente) se caracteriza por un alto grado de madurez técnica y generalmente se reconoce como una ayuda al protocolo o al servicio que significa un beneficio para la comunidad Internet.

A.6.- Capa Nº 1: Aplicación.

Dentro de este nivel se resuelven todas las características relacionadas con la interfaz de usuario, y define la funcionalidad de la aplicación de red, implementándose en su totalidad por software.

En relación a la interfaz con el usuario cabría resaltar su gran importancia, ya que es la parte software encargada de interactuar con el usuario. Es decir, es el "intermediario" entre el usuario y la máquina, encargada de ocultar al usuario la complejidad del problema de la comunicación. De su diseño depende el éxito de la aplicación de red que el usuario este manipulando, condicionando el que pueda sacársele el mayor partido. Características deseables a la hora de llevar a cabo su diseño son: entorno amigable, cómodo, eficiente, poco lioso, etc. En la actualidad existen dos tipos de interfaz: modo consola ó de línea de comandos ("LUI", "Interfaz de Usuario mediante Línea de comandos") y modo gráfico ("GUI", "Interfaz Gráfica de Usuario"). Aunque el entorno "LUI" (también llamada "CUI", "Command-line User Interface") esta cayendo en desuso al tratarse de una interfaz más ruda y más compleja de utilizar, sigue utilizándose al ser más rápida y eficiente, ya que consume menos ancho de banda (sólo hay que transferir texto, no imágenes).

Como ya se ha comentado, dentro de este mismo nivel ó capa se programa (software) la función de red que va a desempeñar la aplicación: control de un equipo remoto, mensajería electrónica, transferencia de ficheros, etc. Todo este tipo de aplicaciones se caracterizan ya que para llevar a cabo una de las funciones mencionadas están normalizados los pasos que se deben seguir (protocolo), lo que permitirá ponerse de acuerdo a las entidades pares (emisor-receptor).

Definición: Se le llama "**protocolo de red**" ó "**protocolo de comunicación**" al conjunto de reglas que controlan la secuencia de acciones que ocurren durante una comunicación entre entidades pares. Los "protocolos de red" se encargan de aspectos tales como: **(1)** la sintaxis de los mensajes intercambiados, **(2)** estrategias para la detección y corrección de errores, **(3)** técnicas de control de flujo y congestión, **(4)** estrategias para la seguridad en la comunicación (autenticación y cifrado de información), y **(5)** control del establecimiento y finalización de la comunicación, en el caso en que esta sea orientada a conexión.

Entre los protocolos más destacados podrían destacarse los siguientes:

A) TELNET. Fue el primer protocolo que se creo, y surgió tras la necesidad de controlar equipos informáticos remotamente (reiniciar servicios, solucionar fallos, cambios de configuración, etc.) sin la necesidad de tener que desplazarse al lugar donde se encuentra el equipo concreto. Hoy en día

este tipo de protocolo está en desuso debido a los problemas de seguridad que presenta, siendo substituido por otros protocolos como "**SSH**"("**Secure SHell**", "SHell Segura"). "SSH" se caracteriza por usar técnicas de cifrado que hacen que la información que viaja a través del canal de comunicaciones viaje de manera segura. Su especificación se encuentra en el documento "**RFC-854/855**".

B) HTTP ("**HyperText Transfer Protocol**", "Protocolo de Transferencia de Hipertexto"). Es el protocolo utilizado por el servicio "WWW" ("World Wide Web") para la transferencia de sitios Web (conjunto de páginas Web entrelazadas). Se le denomina hipertexto, ya que la información que se transfiere no es sólo texto, sino que como todos sabemos, el contenido de una página Web puede estar formada por imágenes, sonidos ó videos que le pueden acompañar. Su especificación se encuentra en el documento "**RFC-2616**".

C) FTP ("**File Transfer Protocolo**", "Protocolo para la Transferencia de Ficheros"). Es el protocolo utilizado por las aplicaciones encargadas de llevar a cabo el intercambio de ficheros entre equipos distantes que se encuentran conectados en red. Su especificación se encuentra en el documento "**RFC-959**".

D) SMTP ("**Simple Mail Transfer Protocol**", "Protocolo Simple de Transferencia de Correo Electrónico"). Fue el primer diseño que se creo para "ARPANET" con la finalidad de intercambiar correos electrónicos. Es el protocolo utilizado entre servidores de correo, con la finalidad de transferir el correo electrónico al servidor destinatario. Su especificación se encuentra en el documento "**RFC-821/822**".

E) POP ("**Post Office Protocol**", "Protocolo de Oficina de Correos"). Protocolo de aparición posterior a "SMTP", se caracteriza por no necesitar una conexión permanente a Internet, ya que los mensajes son almacenados en un servidor de correo, que nos permite que los mensajes puedan ser leídos en cualquier momento, una vez conectados a dicho servidor. Es lo que se denomina el "correo Web". Su especificación se encuentra en el documento "**RFC-1725**".

F) SNMP ("**Simple Network Management Protocol**", "Protocolo Simple para la Gestión de Redes"). Es un protocolo ampliamente utilizado para la administración y gestión remota de los dispositivos de red ("SWITCH", "ROUTER" ó "servidores"). Su especificación se encuentra en el documento "**RFC-1157**".

G) DNS ("**Domain Name Server**" ó "**Domain Name System**"). Es el protocolo encargado de realizar la conversión de nombres de dominio a direcciones IP, aunque también puede llevar a cabo la operación inversa (informar que nombre de dominio se corresponde con una determinada dirección IP). Se dice que resuelve nombres de dominio. Esto es sumamente útil, ya que a un usuario informático le resulta mucho más sencillo aprenderse un nombre de dominio (p.e. "google.com"), que no la dirección IP de la máquina que ofrece el servicio (*p.e. "www.google.com" > "216.239.59.147"*). Su especificación se encuentra en el documento "**RFC-1034/1035**".

Es importante resaltar que las aplicaciones comerciales, al hacer uso de estos protocolos con la finalidad de ofrecernos los servicios comentados, se dividen entre aplicaciones cliente y aplicaciones servidor. Es decir, un usuario de Internet puede visualizar los sitios Web servidos por equipos remotos mediante el uso de aplicaciones como "Internet Explorer" ó "Mozilla Firefox" (aplicaciones cliente), pero estas no son las aplicaciones de las hacen uso los servidores Web para la

gestión de sus sitios Web, sino que utilizan "Apache" ó "IIS" (aplicaciones servidor).

Además de las funciones anteriores, dentro de la capa de aplicación también se llevan a cabo funciones relacionadas con la compresión de la información a transferir, la seguridad, y gestión de las sesiones abiertas, funciones asociadas a los niveles "OSI" de presentación y sesión (hay que recordar que estas capas quedaban aglutinadas dentro de la capa de aplicación en "TCP/IP"). De todas estas funciones, la más compleja e importante es la referida a seguridad, motivo por el cual en vez de tratarse sus aspectos más destacados en este capítulo, se reserva uno (capítulo nº 16) para tal finalidad.

A.7.- Capa Nº 2: Transporte.

Tal como ya se expuso en el punto introductorio del presente capítulo, tras la capa de aplicación, independientemente de la aplicación y protocolo correspondiente que haya sido utilizado, se genera una información (mensaje) en formato binario (0's y 1's) que hay que hacer llegar al destinatario. Relacionándolo con el modelo de referencia "OSI", dentro de este nivel se llevan a cabo todas las funciones de la capa de transporte, y parte de las del nivel de sesión.

De todas las funciones que lleva a cabo, podrían destacarse las siguientes:

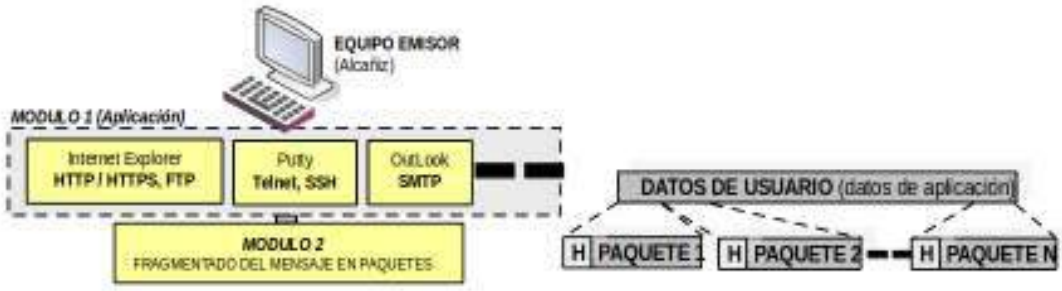
1) Con la finalidad de aprovechar el ancho de banda, y disminuir los retardos en la comunicación, tal como se vio en la introducción, el mensaje es troceado en paquetes (conmutación de paquetes). Con la finalidad de seguir un orden, y permitir su reensamblado en el destino, al dividir el mensaje, se les asigna un "**número de secuencia**" a los paquetes (1, 2, 3, …).

2) Para evitar confusiones, con la finalidad de identificar que aplicación (protocolo) generó la información a enviar se le añade un identificador, conocido como "**número de puerto de origen**". De igual forma, para identificar que aplicación en el equipo destinatario estará en disposición de tratar la información que se le envía, se añade otro identificador denominado "**número de puerto de destino**". Estos identificadores ó números de puerto tienen una longitud de 2 bytes (16 bits), por lo que pueden encontrarse entre 0 y 65535 (2^{16} = 65536). Los números de puerto de cara a "aplicaciones cliente" son dinámicamente asignados por el sistema operativo (no tienen un identificador concreto) en el mismo momento en que se lleva a cabo una solicitud o petición de servicio. En cambio, los números de puerto en las "aplicaciones servidoras" se les conoce como puertos "**well-known**" (bien conocidos), los cuales han sido preasignados por el "**IANA**" ("**Internet Assigned Numbers Authority**", "Agencia de Asignación de Números Internet"), hoy conocida como "**ICANN**" ("**Internet Corporation for Assigned Names and Numbers**", "Corporación de Internet para la Asignación de Nombres y Números"), y no pueden cambiarse. En concreto, estos puertos "well-known" están en el rango del 0 al 1023, cuya lista completa está documentada en la "RFC-1700", entre los que cabría conocer los siguientes: 21-FTP, 23-TELNET, 25-SMTP, 80-HTTP y 110-POP3. En cuanto al resto de números de puerto, del 1024 al 49151, se les conoce como "puertos registrados" al encontrarse reservados ya para determinados servicios, y del 49152 al 65535 como "puertos dinámicos ó privados", ya que pueden ser usados según se crea conveniente.

3) Determina el tipo de comunicación, orientada ó no a conexión, dependiendo del protocolo que se utilice "**TCP**" ("**Transmisión Control Protocol**") ó "**UDP**" ("**User Datagram Protocol**"). Es decir, mediante "**TCP**" podemos establecer una comunicación entre dos equipos informáticos

previo establecimiento de la conexión, lo que permite establecer un "circuito virtual" a priori, por donde viajarán los paquetes de la comunicación, garantizando una fiabilidad (los paquetes llegan ordenados). Es decir, gracias al "número de secuencia", puede darse cuenta de que se ha perdido algún paquete, en cuyo caso solicitará una retransmisión de éste al emisor. Para llevar un control de todo ello, tanto emisor como receptor se informan mediante el uso de paquetes "**ACK**" y "**NACK**" ya comentados en la introducción. En cambio "**UDP**" permite el intercambio de paquetes, denominados en este caso "**datagramas**", a través de la red sin que se haya establecido previamente una conexión, ahorrándonos por tanto, el tiempo invertido en su establecimiento y liberación, pero corriendo el riesgo de que los paquetes no lleguen ordenados (no tienen porque seguir el mismo camino ó circuito), pudiéndose traducir en una perdida de información. Según esto, utilizar el protocolo "**UDP**" tan sólo es recomendable utilizarlo en situaciones, donde la información a transmitir sea reducida (el tiempo de establecimiento y liberalización serían muy significativos), ó en aquellas situaciones donde aún perdiendo parte de los paquetes, la información recibida siguiera siendo entendible, como es el caso de la transmisión de voz ó video.

Toda esta información (número de secuencia, números de puerto e información del protocolo) se agrega a cada uno de los paquetes que forman el mensaje, junto con otra que puede ser de utilidad para el control de la comunicación, formando una cabecera ("**Header**") que acompaña al paquete de datos.



A.8.- Capa Nº 3: Red.

Dentro de este nivel se llevan a cabo dos funciones esenciales: identificación del equipo que origen/destino mediante el uso de "direcciones IP" y el encaminamiento (ó routing) de los paquetes, lo que garantiza que los paquetes que forman parte de la comunicación lleguen al destinatario. Para llevar a cabo estas funciones, se estableció el protocolo "**IP**" ("**Internet Protocol**"), responsable por tanto, del envío y enrutamiento de los paquetes entre máquinas.

Al igual que cuando nos comunicamos con una persona mediante correo postal, hemos de indicar la dirección de destino (ciudad-calle-numero-puerta), y del remitente, para que el cartero sepa que ruta llevar para hacer llegar la carta a su destino (y en caso de un posible error, remitirla), en comunicaciones informáticas ocurre exactamente lo mismo, debemos indicar la "ciudad-calle-número-puerta" donde se localiza el equipo informático destinatario. Si tenemos en cuenta, que los ordenadores solo comprenden cantidades binarias, lo que se hace es indicar mediante cuatro cantidades binarias (4 bytes, 32 bits) tal dirección, lo que hace un total de "2^{32} = 4.294.967.296 direcciones IP" posibles a asignar. No obstante, de cara a su manejo, debido a que a una persona le resultaría bastante latoso tener que manejar cantidades binarias, cada uno de los

grupos ó bytes que forman la dirección IP es traducida a decimal ("**w.x.y.z**").

De igual forma, si tenemos en cuenta que los ordenadores se agrupan en redes, no en ciudades, ni calles, lo que la dirección IP realmente indica es simplemente, la red donde se ubica el equipo (identificador de red), junto con un identificador que le diferencia al equipo dentro de la red donde se encuentra del resto que la forman (identificador de equipo).

Rango de direcciones IP:
00000000.00000000.00000000.00000000 hasta 11111111.11111111.11111111.11111111
0.0.0.0 hasta 255.255.255.255

32 bits (4 bytes)	
IDENTIFICADOR DE RED	IDENTIFICADOR DE EQUIPO

Este identificador debe ser único (direcciones IP públicas), ya que al igual que no existen en una misma ciudad, dos domicilios con la misma dirección postal, tampoco puede darse en el ámbito informático si lo queremos evitar confusiones.

Con la finalidad de facilitar el enrutamiento de los paquetes, las direcciones IP de todos los equipos se agrupan de una manera lógica, estableciendo una división en 5 grupos ó clases de direcciones IP, distinguiéndose entre ellas a través de los primeros bits de comienzo de la dirección:

	IDENTIFICADOR RED								IDENTIFICADOR EQUIPO																			
Clase A:	**0**	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

	IDENTIFICADOR RED																IDENTIFICADOR EQUIPO													
Clase B:	**1**	**0**	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

	IDENTIFICADOR RED																								ID. EQUIPO							
Clase C:	**1**	**1**	**0**	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Clase D:	**1**	**1**	**1**	**0**	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Clase E:	**1**	**1**	**1**	**1**	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Tal como se observará posteriormente, a través de un problema propuesto (problema nº 2), esta clasificación condiciona el número de redes de cada clase que pueden existir, y el número de equipos informáticos que puede albergar cada una de ellas.

De entre todos los valores posibles algunos tienen significados especiales:

1) "**0.0.0.0**" es utilizada por los equipos cuando arrancan (se refieren a si mismos) pero no se usa después.
2) "**0**" como identificador de red seguido de identificador de equipo se utiliza para hacer referencia a un host de la propia red.
3) "**255.255.255.255**" se utiliza para difusión en la red local. Tal como se explico en el capitulo nº 0, a veces puede resultar interesante mandar un mensaje a todos los equipos de la red, en lugar de a un equipo ó conjunto de estos.
4) Identificador de red seguido de todo ceros como identificador de host se utiliza para representar a

la propia red: "**dirección de red**", y por tanto, no se le puede asignar a un equipo de ésta. Esta dirección IP se corresponde con la dirección más baja de toda la red.
5) Identificador de red seguido de todo unos como identificador de host se utiliza para la difusión de información en otra red diferente a la propia. Es lo que se denomina la "**dirección de broadcast**" de la red, y al igual que la "dirección de red", no se le puede asignar a un equipo de ésta. Esta dirección IP se corresponde con la dirección más alta de toda la red.
6) "**127.x.y.z**" se utiliza para la realización de pruebas de bucle interno (referenciar a la máquina local, "localhost"), en las cuales las peticiones de servicio no se transmiten por el cable sino que se procesan localmente. Esto resulta útil, cuando nuestro equipo ó host, hace las veces de servidor (servidor HTTP, FTP, ó cualquier otro tipo), y queremos comprobar su funcionalidad llamando desde una aplicación cliente a nosotros mismos.
7) Las direcciones de clase D son usadas para uso en grupos multicast, y las de clase E son direcciones experimentales, por lo que no están disponibles para uso general. De ahí, que de aquí en adelante tan sólo vayamos a trabajar con direcciones de clase A, B y C.

A modo de ejemplo, esto significa que la dirección IP "64.187.23.201", una vez traducida en binario, "**0**1000000.10111011.00010111.11001001", se advierte que es una dirección de clase A, que se identifica ante el resto de redes de clase A, por ser la número "64", y que a su vez hace referencia al equipo cuyo identificador es el "187.23.201". Además la dirección IP "64.**00000000**.**00000000**.**00000000**" ("64.0.0.0") es la dirección de red, y "64.**11111111**.**11111111**.**11111111**" ("64.255.255.255") la dirección de broadcast de la red.

Todo lo anterior sólo afecta a las máquinas que pertenecen a redes públicas, es decir, aquellas máquinas a las que todo el mundo puede tener acceso a través de Internet. En este ámbito, la asignación de direcciones IP esta regulada por "InterNIC" ("Internet Network Information Center"), la cual se encarga de organizar su asignación de tal forma que el encaminamiento de los paquetes por parte de los "enrutadores" sea lo más sencillo posible. Si por el contrario, el equipo al que le vamos a asignar una dirección IP, es un equipo al cual no se tiene acceso desde Internet, y por tanto, pertenece a una red privada ("IntraNET"), se recomiendan los siguientes identificadores de red, pudiéndose repetir en redes privadas distintas:

- Clase A: "10.0.0.0".
- Clase B: "172.16.0.0" a "172.31.0.0".
- Clase C: "192.168.1.0" a "192.168.254.0".

En el caso en que esta organización de las direcciones IP en cuatro clases no se acomode a nuestros requisitos, y se requiera una mayor división, es posible establecer subredes. Al formar subredes se nos permite (1) mezclar redes de topología diferente ("Ethernet" y "Token Ring"), y (2) reducir la congestión de red redireccionando el tráfico, reduciendo el "broadcasting". Para ello se hará uso de los bits encargados de identificar el equipo, repartiéndolos de tal forma, que los bits de más peso identifiquen la subred y los más bajos, identifiquen al equipo dentro de la subred. Éste y otros aspectos, se verán en mayor profundidad a continuación mediante problemas propuestos.

En cuanto al encaminamiento, indicar que es la acción que se desencadena ante la llegada de un paquete IP a un cruce de caminos (distintas opciones de salida) ante el cual se debe decidir, en base a un criterio previamente establecido (p.e. "Elegir el camino que garantice llegar al destino en menos tiempo, independientemente del coste que conlleve."), el camino de salida por el cual

reenviar el paquete con la finalidad de llegar a su destino. Este cruce de caminos, se presenta continuamente, incluso en el mismo equipo donde se genera, al existir como poco dos alternativas, (1) enviar el paquete hacia el lazo interno, ó (2) enviar el paquete a la red sacándolo por la interfaz ó tarjeta de red.

Para llegar a comprender como este encaminamiento se lleva a cabo, es fundamental conocer la importancia que tiene la "**máscara de red** ó **subred**". Esta máscara es una "dirección" de 32 bits que permite, tras aplicársele a una dirección IP de un equipo, desenmascarar la dirección IP de la red a la cual pertenece. Esto es muy útil, ya que un "enrutador" ó "ROUTER" al recibir un paquete, una vez comprobado que éste no contiene errores, observa cual es la dirección IP del destinatario, y tras aplicarle la máscara reconoce la red a la que pertenece, y por tanto, por que interfaz debe reenviarlo.

Concretamente, esta máscara esta compuesta por "1's" en las posiciones de más peso, y "0's" en las de menor peso, de tal forma, que al hacer una "AND" (función lógica) con la dirección IP, se obtiene como resultado la dirección de su red ó subred. Siguiendo con el ejemplo anterior, si un paquete va destinado al equipo cuya dirección IP es "64.187.23.201" ("**0**1000000.10111011.00010111.11001001"), su máscara deberá ser "255.0.0.0" ("11111111.00000000.00000000.00000000"), para que tras hacer la función lógica AND entre las dos, de cómo resultado la dirección de red a la que pertenece: "**01000000**.**00000000**.**00000000**.**00000000**" ("64.0.0.0").

Para llegar a comprender en mayor profundidad estos aspectos, se remite igualmente al lector a realizar los problemas que a continuación se proponen, cuya solución seguramente ayudará a dar respuesta a muchas dudas que se puedan presentar.

Todo lo anterior, provoca que dentro de este nivel de red sea necesario incluir una nueva cabecera ó "Header" que acompañe a todos los paquetes que forman parte de la comunicación, donde se indicarán las direcciones IP de origen y destino, junto con otra serie de información complementaria.

Por último, indicar que dentro de este nivel, también puede llevarse a cabo una nueva fragmentación de los paquetes, en el caso en que su longitud no sea la aconsejable para su transmisión por el canal. Esta función es ampliamente utilizada por los "enrutadores" que hacen función de pasarela entre dos redes de características diferentes "TCP/IP" (tamaño del paquete variable) y "ATM" (tamaño de paquete fijo e igual a 53 bytes), donde las características de los paquetes que viajan por ambas redes no coinciden.

A.9.- Problemas básicos de repaso sobre Direccionamiento IP

A continuación se resolverán un conjunto de ejercicios de direccionamiento IP que pueden servir de base a los que se presentan en el primer capitulo de este manual, correspondiente al repaso de conceptos básicos.

Ej.A.9.I Clases de Redes

Identifica la clase de dirección a la que pertenecen las siguientes direcciones IP: "200.58.20.165", "128.167.23.20", "16.196.128.50", "150.156.10.10" y "250.10.24.96".

Solución Ej.A.9.I.i: Identificación de la Clase de Red de una dirección IP

Tal como se ha expuesto a lo largo del presente capítulo, la manera de identificar la clase de dirección IPv4 de que se trata (A, B, C, D ó E), consiste en traducir a binario la primera de las cifras decimales que componen la dirección IP, y observar el valor de los bits de más peso:

A) "200.58.20.165": 200 ⇒ **110**01000 ⇒ **Clase C**
B) "128.167.23.20": 128 ⇒ **10**000000 ⇒ **Clase B**
C) "16.196.128.50": 16 ⇒ **0**0010000 ⇒ **Clase A**
D) "150.156.10.10": 150 ⇒ **10**010110 ⇒ **Clase B**
E) "250.10.24.96": 250 ⇒ **1111**1010 ⇒ **Clase E**

Ej.A.9.II Rangos de direcciones IP en función de su Clase de Red

Indica el rango y cantidad de direcciones IP (número de redes y equipos dentro de cada red) ocupado por la Clase A, la Clase B y la Clase C.

Solución Ej.A.9.II.i: Rangos de direcciones IP en las clases A, B y C

Teniendo en cuenta, al igual que se hizo en el problema anterior, que aquellas direcciones IP cuya primera cantidad decimal traducida en binario empiece por "**0**" es **clase A**, si lo es por "**10**" es **clase B**, si es por "**110**" es **clase C**, "**1110**" **clase D**, y "**1111**" **clase E**:

A) Clase A: "**0**XXXXXXX.X.X.X"
⇒ dirección IP de clase A más baja: "**0**0000000.0.0.0" ("0.0.0.0")
⇒ dirección IP de clase A más alta: "**0**1111111.255.255.255" ("127.255.255.255")

ID DE LA RED DE CLASE A	IDENTIFICADOR DEL EQUIPO DENTRO DE LA RED

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Si ahora tenemos en cuenta que en una dirección de clase A, la primera de las cantidades decimales que la forman, identifica a la red de clase A, y las otras tres cifras, el equipo ó host dentro de la red de clase A correspondiente:

➲ Nº de Redes: $2^{8-1} -2 = 2^7 – 2 =$ **126 redes de clase A**.

De todas las combinaciones posibles se han descontado dos, ya que la red "0" no puede usarse por convenio, y la red "127" esta reservada para bucle interno.

➲ Nº de Equipos dentro de una red de clase A: $2^{24} – 2 =$ **16.777.214 equipos** (≈ 17 millones).

De todas las combinaciones posibles, hay que descontar dos, ya que la combinación más baja de todas entraría en conflicto con la dirección de red de clase A correspondiente, y la más alta de todas, con la dirección de broadcast de la red. Por ejemplo, la red de clase A "00001010.X.X.X" ("10.X.X.X"), tiene como dirección de red "10.0.0.0", y como dirección de broadcast ó multidifusión "10.255.255.255", por lo que no se les puede asignar a ningún equipo de la red para que no haya confusiones ni conflictos.

B) Clase B: "**10**XXXXXX.X.X.X"

⇒ dirección IP de clase B más baja: "**10**000000.0.0.0" ("128.0.0.0")

⇒ dirección IP de clase B más alta: "**10**111111.255.255.255" ("191.255.255.255")

IDENTIFICADOR DE LA RED DE CLASE B																**IDENTIFICADOR DEL EQUIPO DENTRO DE LA RED**															
1	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Si ahora tenemos en cuenta que en una dirección de clase B, las dos primeras cantidades decimales que la forman, identifican a la red de clase B, y las otras dos cifras, el equipo ó host dentro de la red de clase B correspondiente:

➲ Nº de Redes: $2^{16-2} = 2^{14} =$ **16.384 redes de clase B**.

➲ Nº de Equipos dentro de una red de clase B: $2^{16} – 2 =$ **65.534 equipos**.

Al igual que en las redes de clase A, de todas las combinaciones posibles, hay que descontar dos, ya que la combinación más baja de todas entraría en conflicto con la dirección de red de clase B correspondiente, y la más alta de todas, con la dirección de broadcast de la red. Por ejemplo, la red de clase B "**10**111111.11111111.X.X" ("191.255.X.X"), tiene como dirección de red "191.255.0.0", y como dirección de broadcast ó multidifusión "191.255.255.255", por lo que no se les puede asignar a ningún equipo de la red para que no haya confusiones ni conflictos.

C) Clase C: "**110**XXXXX.X.X.X"

⇒ dirección IP de clase C más baja: "**110**00000.0.0.0" ("128.0.0.0")

⇒ dirección IP de clase C más alta: "**110**11111.255.255.255" ("191.255.255.255")

IDENTIFICADOR DE LA RED DE CLASE C																							**ID. DEL EQUIPO**								
1	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	**1**	**0**	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Si ahora tenemos en cuenta que en una dirección de clase C, las tres primeras cantidades decimales que la forman, identifican a la red de clase C, y la última de ellas, al equipo ó host dentro de la red de clase C

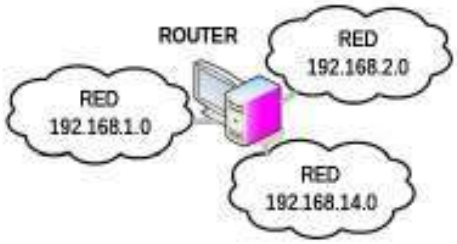
correspondiente:

- Nº de Redes: $2^{24-3} = 2^{21}$ = **2.097.152 redes de clase B** (≈ 2 millones de redes de clase C).
- Nº de Equipos dentro de una red de clase B: $2^8 - 2$ = **254 equipos**.

Por el mismo motivo que en las redes de clase A y B, hay que descontar la dirección más baja y más alta de cada red de clase C, al no poderse asignar a ningún equipo de ésta, con la finalidad de evitar confusiones.

Ej.A.9.III Direccionamiento IP en el diseño de una Red I

Si un ROUTER dentro de una red recibe un paquete IP por una de sus interfaces destinado al equipo con dirección IP "192.168.1.13", ¿Cuál sería la máscara que debería aplicar el ROUTER a la dirección IP anterior para conocer por cual de sus interfaces de red sacarlo, teniendo en cuenta que interconecta las redes "192.168.1.0", "192.168.2.0" y "192.168.14.0"?

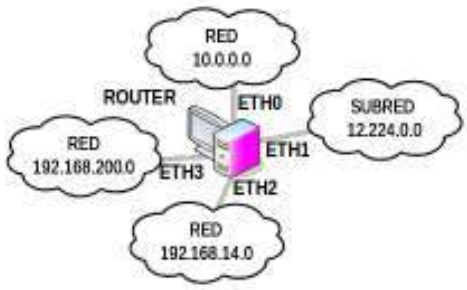


Solución Ej.A.9.III.i: Calculo de la Máscara de Red

Al interconectar el ROUTER tres redes de clase C, la máscara que aplicaría a todo paquete que le llegase, con la finalidad de conocer la red de destino, sería la misma: "**255.255.255.0**". De esta forma, al aplicar dicha máscara a la dirección IP "192.168.1.13" del destinatario, da como resultado la dirección de red "192.168.1.0" donde éste se encuentra, informando al ROUTER porque interfaz sacar el paquete.

Ej.A.9.IV Direccionamiento IP en el diseño de una Red II

Según la red que se muestra a continuación, describe la tabla de enrutamiento ó "routing" (máscara-destino-interfaz de salida) que debería tener programado el ROUTER para que encaminara adecuadamente todo paquete que le llegase a su destino.



Solución Ej.A.9.IV.i: Tabla de enrutamiento básica de un Router

Tal como se ha explicado a lo largo del presente capítulo, la tabla de enrutamiento indica al ROUTER cual es la interfaz de red (**ETH**X) por la que debe encaminar un paquete en función de la red a la que vaya destinado.

Teniendo en cuenta que un paquete IP se incluye la dirección IP de destino, pero no la red de destino, en la correspondiente tabla de enrutamiento será necesario indicar, tal como hemos visto en el apartado anterior, la máscara que será necesaria aplicar a la dirección IP a la que va destinado el paquete, para reconocer la red a la cual dirigirlo. En el caso que se muestra, no hay grandes problemas, ya que en las redes de clase A, B y C, la máscara de red esta predefinida. El único problema que nos podemos encontrar es con la máscara a aplicar a los paquetes que vayan destinados a la subred "12.224.0.0". Si tenemos en cuenta que esa es la dirección de la subred, y por tanto, la dirección IP más baja de la subred, propondremos una posibilidad: "mascara de subred: **11111111.11100000.0.0**"

IDENTIFICADOR DE LA RED DE CLASE A	ID. DE SUBRED	IDENTIFICADOR DE EQUIPO DENTRO DE LA SUBRED "111" DENTRO DE LA RED DE CLASE A "00001100"		
0 0 0 0 1 1 0 0	1 1 1	0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
12	224		0	0

Hay que resaltar, que lo anterior no es más que una posibilidad, ya que lo único que se sabe de una dirección de red o subred, es que al pasarla a binario los bits que se utilizan para identificar a un equipo dentro de una red o subred están a cero, con lo cual otras posibilidades podrían haber sido las siguientes:

Posibilidad Nº 2: "mascara de subred: **11111111.11110000.0.0**"

IDENTIFICADOR DE LA RED DE CLASE A	ID. DE SUBRED	IDENTIFICADOR DE EQUIPO DENTRO DE LA SUBRED "1110" DENTRO DE LA RED DE CLASE A "00001100"		
0 0 0 0 1 1 0 0	1 1 1 0	0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
12	224		0	0

Posibilidad Nº 3: "mascara de subred: **11111111.11111000.0.0**"

IDENTIFICADOR DE LA RED DE CLASE A	ID. DE SUBRED	IDENTIFICADOR DE EQUIPO DENTRO DE LA SUBRED "11100" DENTRO DE LA RED DE CLASE A "00001100"		
0 0 0 0 1 1 0 0	1 1 1 0 0	0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
12	224		0	0

Posibilidad Nº 4: "mascara de subred: **11111111.11111100.0.0**"

IDENTIFICADOR DE LA RED DE CLASE A	ID. DE SUBRED	IDENTIFICADOR DE EQUIPO DENTRO DE LA SUBRED "111000" DENTRO DE LA RED DE CLASE A "00001100"		
0 0 0 0 1 1 0 0	1 1 1 0 0 0	0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
12	224		0	0

Y así podrían plantearse otras posibles soluciones. De aquí se puede deducir la importancia que tiene el definir a priori la máscara a toda red o subred que se forme, evitando así cualquier posible confusión.

En la realidad, como es lógico la solución es única, donde dicha máscara de subred se escoge en

función del número de subredes a formar (contra más bits cojamos para identificar a la subred, más subredes se podrán formar: $2^{nºbits}$ – 2 subredes), y el número de equipos que queramos agregar dentro de cada subred (contra más bits destinemos a identificar el equipo dentro de la subred formada, más equipos podrán agregarse: $2^{nºbits}$ – 2 equipos / subred).

Asumiendo como válida la primera de las posibles máscaras de subred planteadas, "**11111111.11100000.0.0**" ("255.224.0.0"), para la dirección de subred planteada ("12.224.0.0"), la tabla de enrutamiento quedaría de la siguiente forma:

Mascara Red/Subred	Red Destino	Interfaz de red de salida
255.0.0.0	**10.0.0.0**	**ETH0**
255.224.0.0	**12.224.0.0**	**ETH1**
255.255.255.0	**192.168.14.0**	**ETH2**
255.255.255.0	**192.168.200.0**	**ETH3**

A modo de ejemplo, para comprender lo que haría el ROUTER anterior supondremos que le llega un paquete por su interfaz "ETH3" con dirección de destino "192.168.14.167":

1) Aplica la máscara de red que aparece en la primera regla de la tabla de enrutamiento programada, "255.0.0.0", obteniendo como resultado "192.0.0.0".
2) Al observar que "192.0.0.0" no se corresponde con la red de destino "10.0.0.0" de la primera de las reglas, pasa a la siguiente regla de enrutamiento programada.
3) Aplica la máscara de subred de la segunda regla a la dirección de destino "192.168.14.167" ("192.168.14.167" **AND** "255.224.0.0"), obteniendo como resultado: "192.160.0.0".
4) Al observar que "192.160.0.0" no se corresponde con la red de destino "12.224.0.0" programada en la segunda de sus reglas de enrutamiento, se da cuenta que el paquete tampoco debe sacarlo ó encaminarlo hacía la interfaz de red "ETH1". Así pues, pasa a la siguiente regla de la tabla de enrutamiento.
5) Aplica la máscara de subred de la tercera regla "255.255.255.0" a la dirección de destino ("192.168.14.167" **AND** "255.255.255.0"), obteniendo como resultado "192.168.14.0".
6) Al percatarse el ROUTER de que la dirección IP obtenida tras aplicar la máscara, "192.168.14.0", coincide con la dirección IP de la red de destino de dicha regla, advierte que el paquete debe sacarlo a través de la interfaz o tarjeta de red "ETH2".

De esta forma, tras encaminar el paquete recibido hacia "ETH2", si el equipo "192.168.14.167" de la red "192.168.14.0" está encendido, recogerá el paquete.

Ej.A.9.V Análisis de una dirección de Broadcast

Sabiendo que la dirección IP 155.210.157.111 es de broadcast, calcular las posibles mascaras y sus correspondientes direcciones de subred.

Solución Ej.A.9.V.i: Máscara y dirección de Subred a través de la de Broadcast

Al igual que en el ejercicio anterior, a través de este problema vamos a ver la importancia que tiene la asignación de una máscara de red con la finalidad de evitar ambigüedades, ó posibles confusiones.

De momento, a través de la información proporcionada mediante el enunciado del problema sabemos que:

1) Si pasamos la primera cantidad decimal que compone la dirección IP de broadcast suministrada, "155", a binario: "155" ⇒ "**10**011011**"**, podemos advertir que se trata de una dirección IP de clase B. Esto significa que las dos primeras cantidades "155.210" identifican a la red de clase B, y las dos últimas, "157.111" son utilizadas para formar subredes e identificar equipos dentro de estas.
2) Por otro lado sabemos que la dirección IP "155.210.157.111" es una dirección de broadcast, y por tanto, se trata de la dirección más alta de la subred, lo que significa que los bits utilizados dentro de las dos últimas cantidades decimales "157.111" utilizados para identificar a un equipo dentro de la subred deben estar todos a "1", o lo que es lo mismo, al traducir la dirección IP anterior a binario, los últimos bits deben ser todos "1's". Esto significa que la dirección IP anterior podría corresponderse con la siguiente estructuración:

IDENTIFICADOR DE LA RED DE CLASE B																ID. DE SUBRED												ID. DEL EQUIPO DENTRO DE LA SUBRED				
1	**0**	0	1	1	0	1	1	1	1	0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	1	0	1	1	1	1	1
155								**210**								**157**								**111**							

Donde en ese caso (**Posibilidad Nº1**):

- **Máscara de subred**: "255.255.11111111.11100000" ("**255.255.255.224**").
- **Dirección de subred** (dirección más baja): "155.210.10011101.110**00000**" ("**155.210.157.192**").

Y recalcamos el "podría corresponderse", ya que al igual que ocurría en el problema anterior, no es la única posibilidad. Es decir, lo único que sabemos es que por tratarse de una dirección IP de broadcast debe acabar en todo "1's", pero eso no significa que todos sus "1's" finales sean bits utilizados para identificar un equipo dentro de la subred, sino que alguno de esos "1's" podría formar parte, porque no, de los bits utilizados para definir subredes.

Posibilidad Nº 2:

IDENTIFICADOR DE LA RED DE CLASE B																ID. DE SUBRED												ID. EQUIPO DENTRO DE LA SUBRED				
1	**0**	0	1	1	0	1	1	1	1	0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	1	0	1	1	1	1	1
155								**210**								**157**								**111**							

- **Máscara de subred**: "255.255.11111111.11110000" ("**255.255.255.224**").
- **Dirección de subred** (dirección más baja): "155.210.10011101.1101**0000**" ("**155.210.157.208**").

Posibilidad Nº 3:

IDENTIFICADOR DE LA RED DE CLASE B																ID. DE SUBRED												ID. EQUIPO SUBRED				
1	**0**	0	1	1	0	1	1	1	1	0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	1	0	1	1	1	1	1

155	210	157	111

- **Máscara de subred**: "255.255.11111111.11111000" ("**255.255.255.224**").
- **Dirección de subred** (dirección más baja): "155.210.10011101.11011**000**" ("**155.210.157.216**").

Posibilidad Nº 4:

IDENTIFICADOR DE LA RED DE CLASE B																ID. DE SUBRED														ID. EQ. SUB.	
1	**0**	0	1	1	0	1	1	1	1	0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	1	0	1	1	1	1	1
155								210								157								111							

- **Máscara de subred**: "255.255.11111111.11111100" ("**255.255.255.224**").
- **Dirección de subred** (dirección más baja): "155.210.10011101.11011**00**" ("**155.210.157.220**").

Siguiendo esta dinámica, podría pensarse en que existe una última posibilidad, en la cual, de los últimos 16 bits (los 16 bits primeros identifican a la red de clase B), 15 son utilizados para formar subredes dentro de la red de clase B, y el último bit para identificar equipos dentro de cada una de las subredes que se forme (dos posibles direcciones IP a asignar dentro de cada subred):

1	**0**	0	1	1	0	1	1	1	1	0	1	0	0	1	0	1	0	0	1	1	1	0	1	1	1	0	1	1	1	1	1
155								210								157								111							

Esta no es una solución correcta, ya que hay que pensar que dentro de toda subred que formemos hay que descontar la dirección más alta y más baja, con lo cual no nos quedan direcciones para poder asignar a equipos que formen parte de la subred.

Resumiendo, la dirección IP de broadcast suministrada, al no verse acompañada por una máscara de subred, puede provocar confusiones, al poder corresponderse con la dirección más alta de cuatro posibles subredes. Esto significa, que de aquí en adelante, deberemos tener presente la importancia de definir una máscara de red ó subred a la hora de configurar cualquier dispositivo de red.

Ej.A.9.VI Direccionamiento IP en el diseño de una Red III

Una empresa asociada al sector del automóvil desea reestructurar su red informática, departamentándola, organizando los equipos informáticos de trabajo de una manera más idónea. Suponiendo que partiendo de una red de clase C, "192.168.20.0", se quieren formar 3 subredes con 24 equipos (PC's e impresoras) cada una. ¿Cuál sería la máscara de subred más ajustada en relación al número de equipos que van a formar cada subred? ¿Cuál sería la máscara de subred más ajustada en relación al número de subredes que desean crearse?

Solución Ej.A.9.VI.i: División de una Red Lógica en Subredes

Al tratarse de una dirección de clase C, debemos tener en cuenta que las tres primeras cantidades decimales que forman la dirección IP son inamovibles, ya que son las que identifican a la red de clase C, lo que implica, que tan sólo disponemos de la última cantidad decimal para implementar la estructura lógica

solicitada.

IDENTIFICADOR DE LA RED DE CLASE C 10101000				BITS USADOS PARA FORMAR SUBREDES E IDENTIFICAR EQUIPOS DENTRO DE ELLAS
1 1 0 0 0 0 0 0	1 0 1 0 1 0 0 0	0 0 0 1 0 1 0 0	0 0 0 0 0 0 0 0	
192	**168**	**20**	**0**	

Esto significa que disponemos de 8 bits para repartirlos de tal forma, que con parte de ellos (los bits más altos) formemos subredes, y con el resto (los bits de menor peso, o más bajos) identificar equipos dentro de la subred. Si tenemos en cuenta que el número de equipos informáticos que va a acoger cada subred es de 24, esto implica que al menos necesitamos de 5 bits (con 4 bits tan sólo podríamos identificar "$2^4 – 2 = 14$ equipos"):

➲ $2^5 – 2 = 30$ equipos (cota máxima). Se han restado 2, ya que hay que recordar que la combinación binaria más baja y más alta no pueden utilizarse, ya que entraríamos en conflicto con la dirección IP de la subred y de broadcast.

De esta forma tan sólo nos quedarían 3 bits para formar subredes.

Si ahora atendemos al número de subredes a formar, teniendo en cuenta que se desean crear 3 subredes, necesitaremos de al menos 3 bits (con 2 bits tan sólo podríamos crear "$2^2 – 2 = 2$ subredes")

➲ $2^3 – 2 = 6$ subredes (cota máxima). Se han restado 2, ya que hay que tener en cuenta, que la combinación más baja y más alta no pueden utilizarse, ya que entraríamos en conflicto con la dirección de red y de broadcast de la red de clase C.

Por tanto, parece ser que tan sólo hay una única posibilidad, y por tanto, la máscara más ajustada, tanto desde el punto de vista del número de equipos como de subredes a formar sería:
⇒ Máscara de Subred: "**255.255.255.11100000**" ("**255.255.255.224**").

Ej.A.9.VII Direccionamiento IP en el diseño de una Red IV

Dentro del rango de direcciones IP que va desde 155.210.157.16 hasta 155.210.157.143, ¿Cuál es el número máximo de subredes IP de tamaño 32 direcciones que podemos generar? Para cada una de las subredes propuestas dar la dirección de subred, de "broadcast" y la máscara de subred.

Solución Ej.A.9.VII.i: Número máximo de Subredes en base a un criterio dado

Podría pensarse, simplemente por observación de las direcciones IP suministradas, que la cuenta a realizar para dar respuesta al problema planteado, sería burda como hacer "(143-16) / 32 ≈ 4 subredes o subconjuntos de 32 direcciones IP", pero tal como se va a demostrar a continuación, la respuesta no es correcta. Para dar solución al problema planteado, e indicar las direcciones de tales subredes, máscaras y direcciones de broadcast, lo más correcto sería seguir los siguientes pasos:

1) Atendiendo a que la primera cantidad decimal que forma parte del rango de direcciones IP disponible es

155 ("155" ⇒ "**10**011011"), sabemos que se trata de un rango de direcciones IP de clase B: "Dos primeras cantidades decimales identifican a la red de clase B, y mediante el uso de las dos últimas, podemos formar subredes e identificar equipos dentro de estas".

IDENTIFICADOR DE LA RED DE CLASE B "155.210"																BITS USADOS PARA FORMAR SUBREDES E IDENTIFICAR EQUIPOS DENTRO DE ELLAS													
1	**0**	0	1	1	0	1	1	1	0	1	0	1	0	0	0	1	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0
155								**210**								**157**								**16**							
1	**0**	0	1	1	0	1	1	1	0	1	0	1	0	0	0	1	0	0	1	1	1	0	1	1	0	0	0	1	1	1	1
155								**210**								**157**								**143 10001111**							

2) Si tenemos en cuenta que dentro de cada subred deseamos agrupar 32 direcciones (incluidas las direcciones de la subred, y su correspondiente dirección de broadcast), para un máximo de 30 equipos informáticos a poder identificar dentro de estas, será necesario reservar al menos 5 bits:

- ➲ $2^5 = 32$ direcciones IP.
- ➲ $2^5 - 2 = 30$ equipos (cota máxima). Se han restado 2, ya que hay que recordar que la combinación binaria más baja y más alta no pueden utilizarse, ya que entraríamos en conflicto con la dirección IP de la subred y de broadcast.

Según esto, de los 16 bits que se corresponden con las dos últimas cantidades decimales, los últimos 5 bits (los de menor peso) serán reservados para identificar equipos dentro de cada una de las subredes que formemos dentro de la red de clase B "155.210". Los 11 bits restantes, los utilizaremos para formar subredes:

IDENTIFICADOR DE LA RED DE CLASE B "155.210"																IDENTIFICADOR DE SUBRED											ID. EQUIPO				
1	**0**	0	1	1	0	1	1	1	0	1	0	1	0	0	0	1	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0
155								**210**								**157**											**16**				
1	**0**	0	1	1	0	1	1	1	0	1	0	1	0	0	0	1	0	0	1	1	1	0	1	1	0	0	0	1	1	1	1
155								**210**								**157**											**143**				

3) Por último, para conocer el número de subredes que se pueden formar dentro del rango marcado, tan sólo será necesario buscar las posibles combinaciones binarias que se pueden establecer a través de los 11 bits utilizados para formar subredes. Para ello, lo único que hay que hacer es incrementar la cantidad binaria asociada a esos 11 bits, hasta apurar el rango:

IDENTIFICADOR DE LA RED DE CLASE B "155.210"																IDENTIFICADOR DE SUBRED											ID. EQUIPO				
155								**210**								**157**											**16**				
1	**0**	0	1	1	0	1	1	1	0	1	0	1	0	0	0	1	0	0	1	1	1	0	1	0	0	0	1	0	0	0	0
																1	**0**	**0**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	**1**					
																1	**0**	**0**	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**0**					
																1	**0**	**0**	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**1**					
1	**0**	0	1	1	0	1	1	1	0	1	0	1	0	0	0	1	0	0	1	1	1	0	1	1	0	0	0	1	1	1	1
155								**210**								**157**								**143**							

De esta forma, se comprueba que aunque la intuición nos decía que se podían formar 4 subredes con 32 posibles direcciones IP, en realidad sólo es posible formar 3 subredes. Podría pensarse según lo anterior que son 5 pero no es cierto. En concreto, la primera subred, cuyo "IDENTIFICADOR DE SUBRED" es

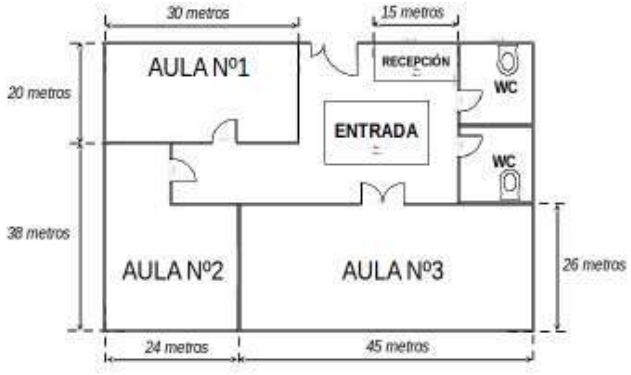
"10011101000", su "ID. EQUIPO" sólo admite como posibles combinaciones binarias válidas el rango comprendido entre "10000" y "11111", lo que hace que sean menos de 32, y por tanto no nos sirve. En relación a la última de las subredes, cuyo "IDENTIFICADOR DE SUBRED" es "10011101100", le ocurre algo similar, ya que su "ID. EQUIPO" sólo admite combinaciones en el rango "00000" a "01111", lo que vuelve a dar un valor inferior a 32.

Concluyendo, a continuación indicaremos las direcciones de cada subred (dirección IP más baja de la subred), la dirección de broadcast (dirección IP más baja de la subred), y su correspondiente máscara:

- **Subred Nº 1.-**
  - ⇒ Dirección IP de la subred: "155.210.10011101.001**00000**" ("**155.210.157.32**").
  - ⇒ Dirección IP de broadcast de la subred: "155.210.10011101.001**11111**" ("**155.210.157.63**").
  - ⇒ Máscara de la subred: "255.255.11111111.111**00000**" ("**255.255.255.224**").
- **Subred Nº 2.-**
  - ⇒ Dirección IP de la subred: "155.210.10011101.010**00000**" ("**155.210.157.64**").
  - ⇒ Dirección IP de broadcast de la subred: "155.210.10011101.010**11111**" ("**155.210.157.95**").
  - ⇒ Máscara de la subred: "255.255.11111111.111**00000**" ("**255.255.255.224**").
- **Subred Nº 3.-**
  - ⇒ Dirección IP de la subred: "155.210.10011101.011**00000**" ("**155.210.157.96**").
  - ⇒ Dirección IP de broadcast de la subred: "155.210.10011101.011**11111**" ("**155.210.157.127**").
  - ⇒ Máscara de la subred: "255.255.11111111.111**00000**" ("**255.255.255.224**").

Ej.A.9.VIII Direccionamiento IP en el diseño de una Red V

Suponiendo que somos los técnicos de una empresa dedicada a la implantación de redes de ordenadores en Zaragoza, y que se nos ofrece por parte de una academia de inglés que esta cambiando sus estrategias docentes (pretenden instruir a sus alumnos mediante clases guiadas vía PC) diseñar su red interna, la cual debe presentar las siguientes características:



- **1º.- Debe constar de cinco *subredes* privadas independientes: una para cada una de las aulas, otra para los administrativos que trabajen en la recepción de la academia, más una última subred que se formará mediante los ordenadores que se colocarán en el salón de entrada dedicados a la búsqueda de información en Internet para los alumnos de la academia.**
- **2º.- El Aula número 1 debería acoger 20 ordenadores.**

- **3º.- El aula número 2 debería acoger otros 20 ordenadores.**
- **4º.- El aula número 3 debería acoger al menos 28 ordenadores (con idea de poder ampliarlo hasta un número de 34 en un futuro).**
- **5º.- Cada una de las aulas anteriores además tiene su propia impresora conectada en RED.**
- **6º.- En recepción se dispone de dos ordenadores, un impresora conectada en red, más la fotocopiadora que también esta conectada en red.**
- **7º.- En la sala de entrada de la academia se dispondrá de 5 equipos.**
- **8º.- En principio, se debe poder salir desde cualquiera de las aulas hacia Internet, al igual que desde recepción y el salón de entrada, a través de una línea ADSL de banda ancha que piensan contratar, y cuyo punto de conexión (roseta) estará en recepción.**
- **9º.- Todas las subredes anteriores, estarán conectadas a un equipo que hará las veces de ROUTER/GATEWAY, situado igualmente en recepción, que hará de puerta de enlace hacía Internet. De igual forma, garantizará que las subredes sean independientes, de tal forma, que un ordenador situado en un aula, por ejemplo, no pueda acceder a los equipos de recepción.**

A) Indica la clase de dirección IP de la red que sería necesaria para la reestructuración lógica de la academia que permita formar las subredes solicitadas. Elige una concreta, la cual usarás en el resto de apartados.
B) Indica la dirección IP de las subredes formadas, sus direcciones IP de broadcast y máscaras de subred.
C) Según lo anterior, indica los rangos de dirección IP a poder asignar a los distintos equipos informáticos que forman parte de cada subred.
D) A través del enunciado del problema, plantea un esquema de interconexión de todos los equipos y dispositivos de interconexión que forman parte de la red.
E) Indica el número de interfaces de red de que debería disponer el PC que hace las veces de ROUTER/GATEWAY para cumplir los requisitos solicitados, indicando la dirección IP a asignar y máscara de subred para cada una de sus interfaces, si asumimos que para la puerta de enlace de cada subred se reserva la dirección IP más alta asignable dentro de cada rango. Para ello supondremos que el MODEM/ROUTER ADSL suministrado por el proveedor de Internet al que hemos alquilado sus servicios, dispone de dos interfaces, una con una IP pública a través de la cual accede a Internet, y otra con una IP privada "212.97.123.65/24".
F) Describe como sería la tabla de enrutamiento que debería tener un equipo que se encontrase en el aula nº 1, asumiendo que debe poder comunicarse con cualquier equipo de su misma subred, y además tiene permiso para salir a Internet a través del ROUTER/GATEWAY.
G) Describe la tabla de enrutamiento del equipo que hace ROUTER/GATEWAY en la red, para permitir la salida a Internet de todos los ordenadores de la red.

Solución Ej.A.9.VIII.i: Diseño de una Red Lógica en base a una Red Física

A) Según las características técnicas indicadas en el diseño de la red de la academia, necesitaríamos selección una clase de dirección IP de tal forma que admitiera la creación de 5 subredes, lo que significa que de los bits atizados en la dirección IP de la clase de red, 3 bits (los tres bits de mayor peso) debemos reservarlos para la creación e identificación de subredes:

- $2^2 - 2 = 2$ subredes, por lo que no es suficiente con 2 bits.
- $2^3 - 2 = 6$ subredes (recordar que la combinación más baja, y más alta se desechan con la finalidad de

evitar el conflicto que se podría generar con la dirección de red y de broadcast de la red).

Por otro lado, si tenemos en cuenta que en el peor de los casos (aula nº3) las subredes deben estar preparadas para acoger un total de hasta 34 equipos informáticos, nos condiciona el tener que reservar al menos 6 bits para identificar los equipos dentro de la subred correspondiente:

- $2^5 - 2 = 30$ equipos, por lo que no es suficiente.
- $2^6 - 2 = 62$ equipos informáticos se pueden identificar con 6 bits como máximo (recordar que la combinación más baja, y más alta se desechan con la finalidad de evitar el conflicto que podría generarse con las direcciones de subred y broadcast de la subred).

Según lo expuesto anteriormente, para no complicar el diseño, deberíamos elegir una clase de dirección IP para la red, de tal forma que los bits destinados a identificar equipos dentro de la red fueran como poco 9 (al menos 3 para identificar subredes dentro de la red, y 6 para identificar equipos dentro de ellas). Esto significa que una dirección de clase C para el diseño de la red quedaría descartada, al contar con tan sólo 8 bits para identificar equipos dentro de la red, con lo cual la opción más razonable podría ser elegir una dirección de clase B: 16 bits para identificar a la red de clase B, y 16 bits para identificar equipos dentro de dicha red, pudiendo organizar éstos en subredes.

Aunque tal como se ha expuesto, elegir una dirección de clase B sería una opción correcta para el diseño, podríamos advertir de que se trata de una solución poco eficiente, debido al bajo aprovechamiento que haríamos del rango de direcciones IP disponibles. Es decir, en una red de clase B se pueden identificar hasta 65.000 equipos ($\approx 2^{16}$) dentro de ella, mientras que si hacemos un recuento del número de equipos que va a formar la academia de inglés no alcanza los 90 equipos (entre PC's, impresoras y fotocopiadora). Según esto, lo más apropiado desde el punto de optimización del rango de direcciones IP disponibles sería escoger una clase C, ya que esta nos permite identificar en torno a 250 equipos ($\approx 2^8$), cantidad que nos es suficiente para nuestras pretensiones. El problema como vamos a ver, es que el diseño o reparto lógico de las direcciones va a ser un poco más complejo. Concretando, la dirección de red que se usará para diseñar la red será: "**192.168.1.0/255.255.255.0**".

IDENTIFICADOR DE LA RED DE CLASE C "192.168.1"			**BITS USADOS PARA FORMAR SUBREDES E IDENTIFICAR EQUIPOS DENTRO DE ELLAS**
1 1 0 0 0 0 0 0	1 0 1 0 1 0 0 0	0 0 0 1 0 1 0 0	X X X X X X X X
192	**168**	**1**	**X**

B) Teniendo en cuenta que en el caso más critico la subred del aula nº 3 debe acoger hasta 35 equipos informáticos (34 PC's + Impresora en red), deberemos contar con 6 bits (los de menor peso) para identificarlos ($2^6 - 2 = 62$ equipos), lo que nos reduce a 2 bits los disponibles para formar subredes ($2^2 - 2 = 2$ subredes).

Si tenemos en cuenta que con esos 2 bits sólo vamos a poder formar 2 subredes, ya que las subredes "**00**" y "**11**" podrían darnos problemas de conflicto con las direcciones de red (dirección IP más baja de la red) y de broadcast (dirección IP más alta de la red) de la clase C "**192.168.1**":

10) Dirección de red:

1 1 0 0 0 0 0 0	1 0 1 0 1 0 0 0	0 0 0 1 0 1 0 0	0 0 **0 0 0 0 0 0**

11) Dirección de broadcast:

1	**1**	**0**	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	1	1	**1**	**1**	**1**	**1**	**1**	**1**

Con lo cual:

IDENTIFICADOR DE LA RED DE CLASE C "192.168.1"																				**BITS USADOS PARA FORMAR SUBREDES E IDENTIFICAR EQUIPOS DENTRO DE ELLAS**									
1	**1**	**0**	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	X	X	X	X	X	X
1	**1**	**0**	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	**0**	**1**	X	X	X	X	X	X
1	**1**	**0**	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	**1**	**0**	X	X	X	X	X	X
1	**1**	**0**	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	1	1	X	X	X	X	X	X
192								**168**								**1**								**X**							

Es decir, al requerir para el diseño del aula nº 3 de 6 bits para identificar equipos dentro de ella, podemos pensar que esto nos condiciona el diseño de la red al parecernos que nos deja con tan sólo 2 bits para formar subredes, con lo cual no podríamos formar las 5 subredes que nos solicitan. A continuación vamos a demostrar que esto no es cierto.

Tal como se muestra a través del razonamiento anterior, estaríamos desechando dos subredes, la "**00**" y la "**11**", y sus 2^6 direcciones IP, por el mero hecho de que ambos rangos contienen una dirección IP (¡¡Una dirección de las 2^6 = 64 direcciones que consta la subred!!) que nos puede crear conflictos. La solución es sencilla: volver a subdividir ambas subredes, desechando aquellas agrupaciones que contengan las direcciones que crean conflicto, y el resto aprovecharlos. Por ejemplo, a continuación dividiré las subredes de 64 direcciones IP, en dos nuevas subredes de 32 direcciones IP, para lo cual haremos uso de un tercer bit:

1	**1**	**0**	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	**0**	**0**	**0**	X	X	X	X	X
																								0	**0**	**1**	X	X	X	X	X
1	**1**	**0**	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	0	1	X	X	X	X	X	X
1	**1**	**0**	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	1	0	X	X	X	X	X	X
1	**1**	**0**	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	**1**	**1**	**0**	X	X	X	X	X
																								1	**1**	**1**	X	X	X	X	X

Tras la división anterior, la red de clase C "192.168.1.0" queda dividida en 6 subredes, de las cuales, cuatro de ellas están formadas por un rango de 32 direcciones IP ($2^5 – 2$ = 30 equipos por subred), y las otras dos, por 64 (2^6 -2 = 62 equipos por subred):

1ª Subred: "**000**". 5 bits para identificar equipos dentro de la subred (2^5 -2 = 30 equipos).

IDENTIFICADOR DE LA RED DE CLASE C "192.168.1"																				**ID. SUBRED**			**ID. EQUIPO DE LA SUBRED**				
1	**1**	**0**	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	**0**	**0**	**0**	X	X	X	X	X

⇒ **¡¡Problema!!** La dirección más baja de la subred, es decir, la dirección de subred, "192.168.1.**00000000**" entra en conflicto con la dirección de red de la clase C, por lo que no es conveniente utilizarla.

2ª Subred: "**001**". 5 bits para identificar equipos dentro de la subred (2^5 -2 = 30 equipos).

IDENTIFICADOR DE LA RED DE CLASE C "192.168.1"																				ID. SUBRED			ID. EQUIPO DE LA SUBRED				
1	**1**	**0**	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	**0**	**0**	**1**	X	X	X	X	X

Al poder albergar hasta 30 equipos dentro de esta subred, podemos usarla para la implementación de cualquiera de las subredes que forman parte de la academia menos la correspondiente al aula nº 3 que requiere de 35 ⇒ Por ejemplo: "**subred del Aula Nº 1**" (requiere 21 direcciones IP asignables, 20 para equipos más una para una impresora en red).

3ª Subred: "**01**". 6bits para identificar equipos dentro de la subred (2^6 -2 = 62 equipos).

IDENTIFICADOR DE LA RED DE CLASE C "192.168.1"																				ID. SUB		ID. EQUIPO DE LA SUBRED					
1	**1**	**0**	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	**0**	**1**	X	X	X	X	X	X

Al poder albergar hasta 62 equipos dentro de esta subred, podemos usarla para implementar la "**subred del Aula Nº 3**" (requiere 34 direcciones IP asignables, y disponemos de 62).

4ª Subred: "**10**". 6bits para identificar equipos dentro de la subred (2^6 -2 = 62 equipos).

IDENTIFICADOR DE LA RED DE CLASE C "192.168.1"																				ID. SUB		ID. EQUIPO DE LA SUBRED					
1	**1**	**0**	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	**1**	**0**	X	X	X	X	X	X

Al disponer de 62 direcciones IP asignables, la subred va más que sobrada de cara a implementar cualquiera de las subredes restantes. El problema es que estaríamos desperdiciando muchas direcciones IP, ya que del resto de subredes que nos quedan por implementar, la que más (aula Nº 2) tan sólo requiere 21 (20 PC's + Impresora en red), es decir, ni la mitad. Según esto, al igual que hemos hecho con la las subredes "00" y "11", para un mayor aprovechamiento del rango de direcciones IP disponibles, lo más lógico sería dividir la subred en dos de 30 direcciones:

IDENTIFICADOR DE LA RED DE CLASE C "192.168.1"																				ID. SUBRED			ID. EQUIPO DE LA SUBRED				
1	**1**	**0**	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	**1**	**0**	**0**	X	X	X	X	X
1	**1**	**0**	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	**1**	**0**	**1**	X	X	X	X	X

De esta forma la subred "**100**" la podríamos utilizar para implementar la "**subred del Aula Nº 2**".

En cuanto a la subred "**101**" la podríamos utilizar para implementar cualquiera de las dos redes restantes. Por ejemplo, la "**subred de la Sala de Entrada**" (5 equipos).

5ª Subred: "**110**". 5 bits para identificar equipos (2^5 -2 = 30 equipos).

IDENTIFICADOR DE LA RED DE CLASE C "192.168.1"	ID. SUBRED	ID. EQUIPO DE LA SUBRED

1	**1**	**0**	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	**1**	**1**	**0**	X	X	X	X	X

Mediante esta subred podríamos implementar la última de las subredes que nos queda: "**subred de Recepción**" (2 PC's + Impresora en red + Fotocopiadora en red).

Aunque ya tenemos implementadas todas las subredes necesarias en la academia, analizaremos la última subred que nos resta, "**111**".

6ª Subred: "**111**". 5 bits para identificar equipos (2^5 -2 = 30 equipos).

IDENTIFICADOR DE LA RED DE CLASE C "192.168.1"																						ID. SUBRED			ID. EQUIPO DE LA SUBRED				
1	**1**	**0**	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	**1**	**1**	**1**	X	X	X	X	X

⇒ **¡¡Problema!!** La dirección más alta de la subred, es decir, la dirección de broadcast de la subred, "192.168.1.**11111111**" entra en conflicto con la dirección de broadcast de la red de la clase C, por lo que no es conveniente utilizarla.

Antes de pasar a definir las direcciones de subred, broadcast y mascara de subred de cada una de las subredes formadas, sería importante resaltar, que siguiendo la estrategia mostrada a lo largo del ejercicio, en caso de haber sido necesario podríamos haber seguido subdividiendo las subredes formadas. En concreto, aquellas que disponen de un rango de direcciones IP más amplio del necesario para su implementación, como puede ser el caso de la subred "**110**" correspondiente a recepción, donde se dispone de 30 direcciones IP asignables, y tan sólo necesitábamos 5 (mucho menos de la mitad), lo que nos permitiría formar nuevas subredes para que fueran aprovechadas por otros agrupamientos de equipos, y así conseguir un mayor aprovechamiento del rango de direcciones IP. En caso de haber sido necesario, que no es el caso, la subdivisión podría haberse realizado tal como se muestra a continuación (división de la subred "110" en dos subredes: "1100" y "1101" con 16 direcciones IP cada una de ellas, de las cuales 14 serian asignables):

IDENTIFICADOR DE LA RED DE CLASE C "192.168.1"																						ID. SUBRED				ID. EQUIPO SUBRED			
1	**1**	**0**	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	**1**	**1**	**0**	**0**	X	X	X	X
1	**1**	**0**	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	0	0	**1**	**1**	**0**	**1**	X	X	X	X

Resumiendo:

➲ **Subred Aula Nº 1:**

⇒ Dirección de Subred: "192.168.1.**001**00000" ("**192.168.1.32**") (dirección más baja de la subred).

⇒ Dirección de Broadcast: "192.168.1.**001**11111" ("**192.168.1.63**") (dirección más alta de la subred).

⇒ Máscara de la subred: "255.255.255.11100000" ("**255.255.255.224**").

➲ **Subred Aula Nº 2:**

⇒ Dirección de Subred: "192.168.1.**100**00000" ("**192.168.1.128**") (dirección más baja de la subred).

⇒ Dirección de Broadcast: "192.168.1.**100**11111" ("**192.168.1.159**") (dirección más alta de la subred).

⇒ Máscara de la subred: "255.255.255.11100000" ("**255.255.255.224**").

➲ **Subred Aula Nº 3:**

⇒ Dirección de Subred: "192.168.1.**01**000000" ("**192.168.1.64**") (dirección más baja de la subred).

⇒ Dirección de Broadcast: "192.168.1.**01**111111" ("**192.168.1.127**") (dirección más alta de la subred).
⇒ Máscara de la subred: "255.255.255.11000000" ("**255.255.255.192**").

➲ **Subred Sala de Entrada:**

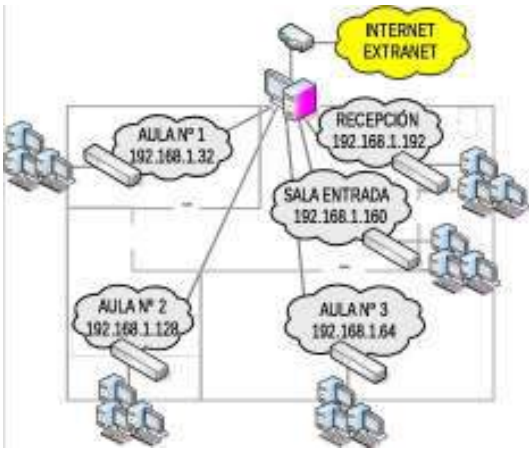
⇒ Dirección de Subred: "192.168.1.**101**00000" ("**192.168.1.160**") (dirección más baja de la subred).
⇒ Dirección de Broadcast: "192.168.1.**101**11111" ("**192.168.1.191**") (dirección más alta de la subred).
⇒ Máscara de la subred: "255.255.255.11100000" ("**255.255.255.224**").

➲ **Subred de Recepción:**

⇒ Dirección de Subred: "192.168.1.**110**00000" ("**192.168.1.192**") (dirección más baja de la subred).
⇒ Dirección de Broadcast: "192.168.1.**110**11111" ("**192.168.1.223**") (dirección más alta de la subred).
⇒ Máscara de la subred: "255.255.255.11100000" ("**255.255.255.224**").

C) En cuanto a los rangos de direcciones a poder asignar dentro de cada subred, no son más que el conjunto de direcciones situadas entre la dirección de subred y la de broadcast de éstas.

D) El esquema de interconexión podría reflejarse de la siguiente forma:



E) Tal como se puede advertir a través de la figura anterior, el equipo que hace las veces de "ROUTER/GATEWAY" requiere tener instaladas al menos 6 tarjetas de red, donde a través de 5 de sus tarjetas establecerá comunicación con las distintas subredes que se han formado, y mediante una última tarjeta de red dará salida a Internet a través del "MODEM/ROUTER" del proveedor de Internet.

Teniendo en cuenta que al haber una conexión directa entre el equipo "ROUTER/GATEWAY" y cada una de las subredes, este equipo pasa a formar parte de ellas, para ser identificado dentro de éstas requiere igualmente que se le asigne una dirección IP, que normalmente suele ser la última dirección IP asignable dentro de la subred. Es decir, se van gastando direcciones IP de manera consecutiva a medida que se agregan nuevos equipos a la red, reservando para no entorpecer este proceso, la última dirección para el "gateway" ó "puerta de enlace" hacia el exterior.

Denominando por "ETH***X***" a las distintas tarjetas de red (donde "***X***" indica el número de interfaz concreta) que posee el "gateway", la asignación de direcciones IP de cara a su configuración sería:

➲ **ETH0:** Teniendo en cuenta que la IP del "MODEM/ROUTER" es "**212.97.123.65/24**" (el "24" indica el número de "1's" que contiene la máscara, por lo que es equivalente a "255.255.255.0").

⇒ Dirección IP/Máscara de red: "**212.97.123.64/255.255.255.0**" ó "**212.97.123.64/24**" (como no se han establecido requisitos, y no hay más equipos en esa red que el "gateway" y el "MODEM/ROUTER", se ha escogido una dirección cualquiera dentro de la misma red).

⇒ Dirección de Broadcast: "212.97.123.**11111111**" ("**212.97.123.255**") (dirección más alta de la red).

➲ **ETH1:**

⇒ Dirección IP/Máscara de subred: "**192.168.1.62/255.255.255.224**" ó "**192.168.1.62/27**" (dirección asignable más alta de la subred).

⇒ Dirección de Broadcast: "**192.168.1.63**".

➲ **ETH2:**

⇒ Dirección IP/Máscara de subred: "**192.168.1.158/255.255.255.224**".

⇒ Dirección de Broadcast: "**192.168.1.159**".

➲ **ETH3:**

⇒ Dirección de Subred: "**192.168.1.126/255.255.255.192**" ó "**192.168.1.126/26**".

⇒ Dirección de Broadcast: "**192.168.1.127**".

➲ **ETH4:**

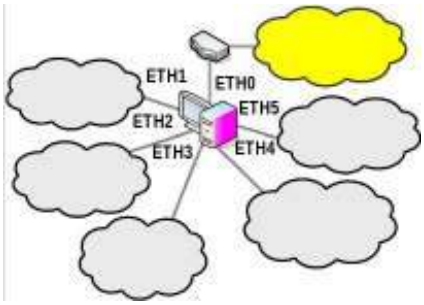
⇒ Dirección de Subred: "**192.168.1.190/255.255.255.224**" ó "**192.168.1.190/27**".

⇒ Dirección de Broadcast: "**192.168.1.191**".

➲ **ETH5:**

⇒ Dirección de Subred: "**192.168.1.222/255.255.255.224**" ó "**192.168.1.222/27**".

⇒ Dirección de Broadcast: "**192.168.1.223**".



F) En cuanto a la tabla de enrutamiento de un ordenador del aula Nº 1, asumiendo que tan sólo dispone de una tarjeta de red (eth0), tan sólo contendrá tres reglas:

1ª regla de enrutamiento: En el caso en que la dirección IP de destino comience por "127" (p.e. "127.0.0.1"), el paquete será reenviado al propio equipo (equipo local ó "**localhost**") a través de su interfaz de lazo interno (interfaz virtual "**lo**", "**loopback**"), sin necesidad de enviarlo a través de la tarjeta de red "**eth0**".

2ª regla de enrutamiento: En el caso en que la dirección IP de destino se corresponda con una dirección IP de un equipo de la red propia, el paquete será enviado a través de la interfaz de red "**eth0**" a través del canal que los comunica.

3ª regla de enrutamiento: En el caso en que la dirección IP de destino no se correponda con ninguna de las dos anteriores, significará que se trata de un equipo externo a nuestra propia red, y por tanto, inalcanzable para el propio equipo. En ese caso, el paquete se le enviará por defecto a través de "**eth0**" al equipo que hace de "gateway" ("**192.168.1.62**") ó "puerta de enlace" hacia el exterior de la red, al tener este capacidad de

enrutamiento entre redes.

Según esto, la tabla de enrutamiento quedaría de la siguiente forma:

Mascara de red/subred	Red Destino	Dirección Gateway	Interfaz de salida
255.0.0.0	**127.0.0.0**	**0.0.0.0**	**lo**
255.255.255.224	**192.168.1.32**	**0.0.0.0**	**eth0**
0.0.0.0	**0.0.0.0**	**192.168.1.62**	**eth0**

Es decir, si traducimos la tabla de "**routing**" anterior, utilizada cuando un equipo del aula Nº 1 va a enviar un paquete: **(1)** se aplica la primera regla, forzando la máscara "**255.0.0.0**" a la dirección IP de destino del paquete, y a si comprobando si va dirigido a la red "**127.0.0.0**". En caso afirmativo, el paquete se envía a través de la interfaz "**lo**", sin la necesidad de la ayuda de ningún "gateway". En caso de respuesta negativa, **(2)** se aplica la segunda regla, y tras imponerle a la dirección IP de destino la máscara "**255.255.255.224**", se comprueba ahora si el destinatario pertenece a la red "**192.168.1.32**". En caso de respuesta afirmativa, el paquete se envía a la red a través de la interfaz "**eth0**" sin la necesidad de "gateway", ya que el destinatario es alcanzable. En caso contrario, **(3)** se aplica la última de la reglas, y tras aplicar la máscara "**0.0.0.0**" a la dirección IP de destino, se obtiene que la red de destino es "**0.0.0.0**" (la "AND" entre cualquier dirección IP, y una máscara que vale todo ceros, el resultado son todo ceros), y al coincidir con la red de destino de la tercera regla, automáticamente el paquete es reenviado al "**gateway**" ("**default gateway**", ó gateway por defecto) con dirección "**192.168.1.62**" a través de "**eth0**".

G) Siguiendo los mismos razonamientos que en apartado anterior (F), la tabla de "routing" del equipo que hace las veces de "ROUTER/GATEWAY" sería la siguiente:

Mascara de red/subred	Red Destino	Dirección Gateway	Interfaz de salida
255.0.0.0	**127.0.0.0**	**0.0.0.0**	**lo**
255.255.255.0	**212.97.123.0**	**0.0.0.0**	**eth0**
0.0.0.0	**0.0.0.0**	**212.97.123.65**	**eth0**

Donde se ha presupuesto que el "ROUTER/GATEWAY" sólo debe enrutar hacia el exterior (Internet), pero no entre las distintas subredes, garantizando el aislamiento de estas.

En el caso en que nos hubiese interesado comunicar las distintas subredes entre sí, deberíamos añadir las correspondientes reglas de enrutamiento hacia cada una de ellas:

Mascara de red/subred	Red Destino	Dirección Gateway	Interfaz de salida
255.0.0.0	**127.0.0.0**	**0.0.0.0**	**lo**
255.255.255.0	**212.97.123.0**	**0.0.0.0**	**eth0**
255.255.255.224	**192.168.1.32**	**0.0.0.0**	**eth1**
255.255.255.224	**192.168.1.128**	**0.0.0.0**	**eth2**
255.255.255.192	**192.168.1.64**	**0.0.0.0**	**eth3**
255.255.255.224	**192.168.1.160**	**0.0.0.0**	**eth4**
255.255.255.224	**192.168.1.192**	**0.0.0.0**	**eth5**
0.0.0.0	**0.0.0.0**	**212.97.123.65**	**eth0**

Tal como se puede advertir a través de las tablas anteriores, el equipo "ROUTER/GATEWAY" para

salir hacia Internet, requiere a su vez de otro "gateway" ó "puerta de enlace", que en este caso es el "MODEM/ROUTER".

Por último señalar, aunque se verá en la última de las prácticas propuestas (capítulo Nº 15), que en el caso de que las funciones de "ROUTER/GATEWAY" las lleve a cabo un PC normal y corriente, será necesario activar el reenvío de paquetes IP, también llamado "**ipforwarding**". En el caso de que no se implemente a través de PC, sino mediante hardware específico (p.e. "Router Cisco"), las cosas cambian un poco, pero la filosofía es idéntica.

Ej.A.9.IX Propuesta Práctica

Divide la red de equipos donde te encentres en subredes, y comprueba que aunque los equipos están interconectados entre sí a través de un "HUB" ó un "SWITCH", la comunicación entre equipos pertenecientes a subredes distintas es imposible (p.e. intentando hacer un "ping"), ya que como estamos advirtiendo, el dispositivo encargado de interconectar equipos pertenecientes a subredes diferentes no es ninguno de los anteriores, sino un "ROUTER".

A.10.- IPv6 (Protocolo IP versión 6).

La versión del protocolo IP que hemos visto en el punto anterior es la cuarta, "**IPv4**". Tal como se ha mostrado, las direcciones IP dentro de esta versión están compuestas por 32 bits, lo que permite identificar 2^{32} equipos informáticos diferentes, esto es, algo más de cuatro mil millones. Este número puede parecer grande, pero si atendemos a la cantidad de direcciones que no pueden utilizarse por ser especiales y el vertiginoso aumento de la demanda existente para conectarse a Internet a través de dirección IP pública, la cantidad anterior se queda corta. No obstante, gracias a la formación de redes privadas por parte de proveedores de Internet (dirección IP asignada no es pública), y a fuertes restricciones del "InterNIC" a la hora de asignar redes completas (llegan a asignar incluso subredes de clase C), ha permitido utilizar este sistema hasta nuestros días (vigente desde 1970).

Sin embargo, desde hace varios años se van desarrollando nuevas versiones del protocolo IP que evita estas restricciones además de hacerlo más rápido en los enrutamientos y menos pesado (que tenga menos información añadida).

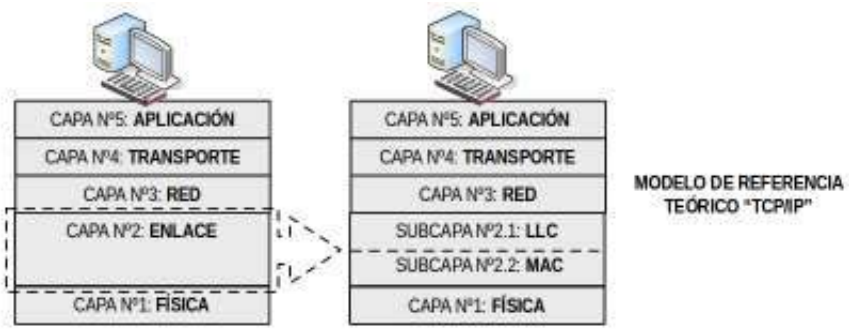
La versión que parece destinada a sucederlo es la sexta, "**IPv6**". Vamos a ver alguna de sus características más notables:

- Utiliza 128 bits en lugar de 32, por tanto dispondremos de 2^{128} direcciones distintas que parecen suficientes para todas las necesidades presentes y futuras.
- Estos 128 bits, al igual que ocurría con los 32 de la "IPv4", se agrupan para identificar de una forma más compacta. En este caso crearemos 8 grupos de 16 bits cada uno de ellos, donde cada uno de los grupos se escribe de forma hexadecimal (16 bits 4 dígitos hexadecimales), separándolos por dos puntos, "**:**". Por ejemplo: "**21a9:b6cc:688d:a901:323e:8266:ef18:bb2f**".
- Para simplificar, se pueden omitir los ceros a la izquierda de cada grupo e incluso los grupos que sean todo ceros. Por ejemplo: "…:52af:0000:b1c9:..." se podría escribir "…:52af::b1c9:...".

Por tanto, "IPv6" es una nueva estructura de paquetes que es incompatible (mediante encapsulamiento del "IPv4" se solucionan las incompatibilidades) con los sistemas "IPv4", pero que nos ofrece muchos beneficios como son un espacio de direcciones extendido, una cabecera más simplificada, soporte para el tráfico dependiente del tiempo (voz y video), y la posibilidad de añadir nuevas características.

A.11.- Capa Nº 4: Enlace.

Dentro de esta capa ó nivel, se llevan a cabo todas aquellas tareas relacionadas con el control de la comunicación y gestión del enlace: control de acceso al medio, control de flujo y control de errores. Al igual que en el modelo "OSI", en el modelo "TCP/IP", vuelve a subdividir este nivel con la finalidad de que las funciones se llevan a cabo de la manera más independiente: "subcapa LLC" (control de flujo y de errores) y "subcapa MAC" (control de acceso al medio).



A.12.- Subcapa LLC: Control de Enlace Lógico.

Aunque la complejidad de esta capa es mucho mayor, a continuación indicaremos las características más relevantes, sin entrar en disquisiciones ó detalles excesivamente técnicos.

Con la finalidad de garantizar una comunicación a través del enlace entre emisor y receptor libre de errores, a nivel hardware, mediante unos chips electrónicos preparados para tal función, se llevan a cabo codificaciones (emisor) y decodificaciones (receptor), que permiten detectar los errores que se hayan podido producir durante la transferencia de los paquetes a través del canal de comunicaciones por culpa del ruido ó las interferencias. Las técnicas matemáticas más ampliamente usadas para llevar a cabo dicha detección son la introducción de bits de paridad (paridad lineal de un bit, paridad longitudinal y transversal ó bidimensional) ó el uso de códigos cíclicos (CRC), técnicas ya vistas en el capítulo anterior. Una vez detectado un posible error, lo más habitual es que la propia circuitería (hardware) trate de corregir el error mediante la utilización de códigos de "Hamming", y en caso de dicha corrección no pueda llevarse a cabo, es cuando es solicitada desde el receptor, una retransmisión del paquete concreto al emisor.

Es importante tener en cuenta, que aunque en otras capas del modelo "TCP/IP" se llevan a cabo comprobaciones de la existencia de errores, mediante la ayuda de campos introducidos en la cabecera correspondiente de ese nivel (**Checksum**), dicha comprobación sólo es a nivel de cabecera, no afectando a los datos de información, que es lo que realmente se hace aquí.

Por otro lado, con la finalidad de que el emisor no sobrecarga al receptor como consecuencia de que se emitan datos una velocidad mayor de la que no pueden ser tratados por el

receptor, se establecen "**técnicas de control de flujo**". En este caso emisor y receptor, pueden ser cualquier equipo (nodos intermedios ó enrtuadores) que nos encontremos a lo largo del enlace que se establece entre el equipo que genero los datos, y el destinatario.

El procedimiento más sencillo para establecer un control de flujo es mediante la utilización de "**protocolos de parada y espera**". En esta técnica, una vez que el emisor envía una trama (en este nivel, al conjunto formado por el paquete de datos y las "Headers" impuestas se le denomina trama), no envía la siguiente hasta no recibir una confirmación (ACK) por parte del receptor e indicación de que envíe una nueva trama. De esta forma, el receptor controla el ritmo o flujo con que se transmiten los datos, evitando de esta forma su sobrecarga. Aunque es una técnica ampliamente utilizada y muy sencilla, esta totalmente desaconsejado su uso, cuando el tamaño de las tramas de información es reducido, ya que invertiríamos un tiempo excesivamente apreciable en controlar el flujo en relación al tiempo invertido en transferir lo que realmente es información, lo cual se traduciría en un enlace con una eficiencia bajísima.

Para solucionar el problema anterior presentado por los protocolos de parada y espera, surgen los "**protocolos de ventana deslizante**". En este caso, es posible emitir la trama "n+1" antes de haber recibido la confirmación de la trama "n", pudiendo viajar simultáneamente a través del mismo enlace varias tramas y paquetes de confirmación (ACK), en concreto tantos como admita el ancho temporal de la ventana utilizada. Es decir, el emisor, una vez enviada una trama, durante un intervalo de tiempo (ventana temporal) determinado a priori, esta a la espera del "ACK" procedente del receptor, durante el cual asume que no ocurre nada anómalo y que por tanto, puede seguir enviando el resto de las tramas. Si transcurrido dicho tiempo no es recibida una contestación, se retransmite la correspondiente trama.

Otra mejora que presenta el uso de "ventana deslizante", es que el receptor no se ve obligado a tener que mandar inmediatamente un "ACK", sino que puede esperarse un tiempo a ver si el receptor tiene que también mandar datos al emisor, y aprovechar este envío para mandar también el "ACK". A esta técnica también se le conoce como "piggybacking".

A.13.- Subcapa MAC: Control de Acceso al Medio.

En muchos casos, se da la situación de que uno ó los dos equipos que forman parte de la comunicación (emisor y receptor), se encuentran dentro de una red de área local donde el canal de comunicaciones es compartido (red de difusión). Hoy en día esta situación suele darse, cuando el equipo pertenece a una red de equipos interconectados a través de un "HUB" (repetidor multipuerto), ó redes inalámbricas (wireless).

En estos casos, donde el canal de comunicaciones es compartido por varios equipos simultáneamente, tal como ya se insistió en los primeros capítulos (0 y 1), es necesario establecer unas normas que controlen el acceso al medio de forma equitativa, ya que sino corremos el riesgo de que se produzcan colisiones, como resultado de que más de un equipo trate de emitir al mismo tiempo. Dichas normas han llegado a convertirse en estándares de "IEEE", existiendo tantos, como

topologías existen donde se utilizan técnicas de difusión ("Ethernet IEEE 802.3", "Token Bus IEEE 802.4", "Token Ring 802.5", etc.). De entre todos estos estándares, a continuación tan sólo estudiaremos los que en la actualidad pueden tener mayor repercusión: "IEEE 802.3" y "IEEE 802.11" (Wireless).

A.14.- IEEE 802.3: Ethernet.

Aunque en al actualidad, con la implantación de redes conmutadas punto a punto mediante la utilización de "SWITCH" (conmutador) como elementos de interconexión, ha dejado a un lado los problemas de colisión que se producen en redes de difusión, el estándar "**Ethernet**" sigue estando vigente debido al abaratamiento de los "HUB" (repetidor multipuerto), claro esta, a costa de una disminución apreciable del ancho de banda.

Ethernet, para solucionar los grandes problemas que provocan las colisiones utiliza la técnica "**CSMA/CD 1 persistente**". Mediante esta técnica, cuando una estación tiene datos para enviar, en primer lugar, comprueba si alguien esta haciendo uso del canal. Si éste está ocupado espera hasta que se desocupe, y entonces transmite la correspondiente trama. En el caso en que de la casualidad de que dos equipos ó estaciones de trabajo se encontrasen en la misma situación, y ambas al ver el canal desocupado, transmitan simultáneamente, se detectará como resultado una colisión, situación ante la cual, las estaciones esperan un tiempo aleatorio, tras el cuentan intentan de nuevo la transmisión.

La terminación "CD", indica "Collision Detect", lo que advierte que las estaciones de trabajo que forman parte de la red Ethernet son capaces de detectar una colisión, lo que les permite terminar sus transmisiones inmediatamente. De este modo se ahorra tiempo y ancho de banda.

"Persistente" informa de que una vez detectado por parte de un equipo que el canal esta ocupado, persiste, y prueba constantemente tras un tiempo aleatorio, si el canal ya ha quedado libre.

Esta técnica "CSMA/CD 1 persistente" esta totalmente desaconsejada en aquellos casos donde el tiempo de propagación de la trama sea apreciable (redes de difusión extensas), ya que puede ocurrir, que poco después de mandar una trama por parte de una estación de trabajo, otra estación al no haberse percatado de ello debido al retardo en la propagación de la señal, se ponga igualmente a transmitir su correspondiente trama, provocando como resultado una colisión, y por tanto, una nueva retransmisión, lo que se traduce en una disminución de la eficiencia de la red, y un desaprovechamiento del ancho de banda. Por este motivo, en redes Ethernet están acotadas las distancias entre equipos, en función del tipo de canal utilizado.

Al igual que "CSMA/CD 1 persistente", existen otras técnicas más depuradas y más eficientes, como puede ser "protocolos de contención limitada" ó "protocolos de paso de testigo", pero no son contempladas por el estándar.

En cuanto al tipo de canal compartido a utilizar, puede ser pares trenzados, cable coaxial ó fibra óptica, de hay que existan distintos tipos de especificaciones Ethernet en función del tipo concreto:

Especificaciones Ethernet cuando el medio es cable coaxial / pares trenzados						
Denominación	**Tipo de Cable**	**Pares**	**Conector**	**Longitud**	**Nº Estaciones**	**Dúplex**

				Segmento	por Segmento	
10BASE5	Coaxial grueso	-	"vampiros"	500 metros	100	No
10BASE2	Caxial fino (RG 58)	-	BNC	185 metros	30	No
10BASE-T	UTP categoría 3	2	RJ-45	100 metros	1	Sí
10BASE-T	UTP categoría 5	2	RJ-45	150 metros	1	Sí
100BASE-TX	UTP categoría 5	2	RJ-45	100 metros		Sí
100BASE-TX	STP	2	9 pin D sub.	100 metros	1	Sí
1000BASE-CX	STP	2	8 pin HSSDC 9 pin D sub.	25 metros	1	Sí
1000BASE-T	UTP categoría 5	4	RJ-45	100 metros	1	Sí
10GBASE-T (en proyecto)	UTP clase E (cat. 6) UTP clase F (cat. 7)	4	RJ-45	55 metros 100 metros	1	Sí

Especificaciones Ethernet cuando el medio es Fibra Óptica					
Denominación	**Ventana Trabajo**	**Emisor Luz**	**Conector**	**Distancia**	**Tipo de Fibra**
10BASE-FL	1ª	LED	ST	2 Kilómetros	Multimodo 62,5/125 µm
100BASE-FX	2ª	LED	SC	2 Kilómetros	Multimodo 62,5/125 µm
100BASE-SX (propuesto)	1ª	LÁSER	SC ó ST	500 metros 500 metros	Multimodo 62,5/125 µm Multimodo 50/125 µm
1000BASE-SX	1ª	LÁSER	SC	275 metros 550 metros	Multimodo 62,5/125 µm Multimodo 50/125 µm
1000BASE-LX	2ª	LÁSER	SC	550 metros 550 metros 5 Kilómetros	Multimodo 62,5/125 µm Multimodo 50/125 µm Monomodo 9 µm
10GBASE-SX	1ª	LÁSER	SC	33 metros 300 metros	Multimodo 62,5/125 µm Multimodo 50/125 µm
10GBASE-LX	2ª	LÁSER	SC	Hasta 15 Km.	Monomodo 10 µm
10GBASE-EX	3ª	LÁSER	SC	Hasta 100 Km.	Monomodo 10 µm

Donde en todas las especificaciones, la primera cantidad, expresa la máxima velocidad de transferencia de información en "Mbps" (Megabits por segundo). Por ejemplo, "1000BASE-T" admite una tasa de 1000Mbps (1Gbps), y "10GBASE-LX" hasta 10.000Mbps (10Gbps). La palabra

"BASE" significa que la señal se envía al canal de comunicaciones sin modular en frecuencia, es decir, se dice que está en "banda base".

Por último, indicar que la norma "IEEE 802.3" en realidad se encuentra dividida en diferentes apartados, que se han ido agregando a medida que los avances tecnológicos han provocado cambios apreciables en algún aspecto del estándar. Concretamente podrían destacarse:

- **802.3u**: Fast Ethernet (10Mbps 100Mbps).
- **802.3x**: Ethernet full duplex y control de flujo.
- **802.3z**: Gigabit Ethernet (100Mbps 1.000Mbps).
- **802.3ab**: Gigabit Ethernet en cable UTP-5.
- **802.3ad**: Agregación de enlaces.
- **802.3ae**: 10 Gigabit Ethernet (1.000Mbps 10.000Mbps).
- ...

A.15.- IEEE 802.11: Wireless LANs (WLAN).

Las "**WLAN**" son redes de área local que se caracterizan por interconectar equipos informáticos (ordenadores, impresoras, etc.) sin la necesidad cablearlos ("Wire-Less", "sin cable). La ventaja de éste tipo de redes es indiscutible, al ahorrarnos el coste de infraestructura que conlleva el tendido de todo el cableado de red, y permitir una movilidad de los equipos (no existe una roseta fija donde conectar el equipo) sin provocar cambios en ella. Por otro lado, presenta dos grandes inconvenientes, **(1)** ya que la equipación de red es más cara, y **(2)** la velocidad de transferencia (ancho de banda) es más reducido que en redes cableadas, al tratarse de un medio compartido por multitud de servicios (radios, televisiones, radioaficionados, etc.), y presentar una tasa de errores muy elevada (los errores se traducen en retransmisiones, y estas a su vez en una disminución del ancho de banda útil).

En la realidad, las redes no suelen ser inalámbricas, sino que son redes híbridas, donde hay ciertos segmentos de red que están cableados, y otros son inalámbricos. En relación a estos segmentos, ó secciones de red inalámbricos, la tecnología utilizada puede ser de dos tipos: **(1)** infrarrojos y **(2)** microondas (ondas de radio ó electromagnéticas), aunque en la actualidad hablar de redes inalámbricas esta asociado a las (2), ya que debido a los fuertes inconvenientes de los infrarrojos (cualquier obstáculo, paredes, niebla, lluvia, etc., entorpece la comunicación, resultando efectivas únicamente para distancias muy cortas) están en desuso. Por este motivo a partir de aquí, cualquier referencia a redes "**wireless**" estará asociada a redes inalámbricas vía radio.

En cuanto a las técnicas utilizadas para modular la señal de información en frecuencia, y convertirla en una onda de radio (al ser la frecuencia de trabajo mayor a 1GHz recibe el nombre de microondas) podrían destacarse las siguientes:

A) "**FHSS**" ("Frequency Doping Spread Spectrum"): Sistema de bajo rendimiento, poco utilizado actualmente.
B) "**DSSS**" ("Direct Sequence Spread Spectrum"): Buen rendimiento y alcance. Es la modulación más utilizada hoy en día (IEEE 802.11b).
C) "**OFDM**" ("Orthogonal Frequency Division Multiplexing"): Máximo rendimiento (IEEE 802.11g).

Si en Ethernet la técnica utilizada para controlar el acceso al medio era "CSMA/CD", en este caso, la usada es "**CSMA/CA**" ("**CS Multiple Access with Collision Avoidance**"), el cual se caracteriza por detección de portadora y elusión de colisiones. Por tanto, se trata de un sistema "CSMA" a secas, ya que aunque las colisiones pueden seguir produciéndose, no tiene capacidad de detectarlas ("CD"). Con la finalidad de corroborar que la emisión no provoco una colisión, una vez que es enviada una trama al medio por parte de una estación de trabajo, se espera transcurrido un tiempo, que el destinatario le envíe una seña de confirmación "ACK". En caso de no ser recibida, se vuelve a reenviar la trama.

Al igual que con Ethernet, el estándar "IEEE 802.11" ha ido generando nuevas especificaciones que se han ido correspondiendo con los avances tecnológicos que se iban produciendo dentro de este campo. A modo de resumen, se muestra la siguiente tabla:

Nivel Físico	Banda	Velocidades	Alcance a máxima vel.	Uso	Características
Infrarrojos	850950 nm	1 y 2Mbps	20 metros	Muy raro	Cortas distancias sin obstáculos
FHSS	2,4 GHz	1 y 2Mbps	150 metros	Reducido	Sistemas Bluetooth y hornos microondas le interfieren fácilmente
DSSS	2,4 GHz	1 y 2Mbps (**802.11**) 5,5 y 11Mbps (**802.11b**)	30 metros	Muy utilizado	Buen rendimiento y alcance
OFDM	5 GHz 2,4 GHz	6, 9, 12, 18, 24, 36, 48 y 54Mbps (**802.11a**) Hasta 54Mbps (**802.11g**)	50 metros 100 metros	Utilizado y en auge	Máximo rendimiento

A.16.- Capa Nº 5: Física.

Al igual que en el modelo "OSI", esta capa se encarga de llevar a cabo la transmisión de los bits de información a través del canal ("codificación de canal", la "modulación/demodulación de la señal", "multiplexación de señales", etc.), aspectos ya tratados en el capítulo Nº2 del libro de "Instalación y Mantenimiento de Servicios de Redes de Área Local, Ed. Mira" editado por los mismos autores que han creado este tutorial.

De igual forma, en este nivel se especifican todos los aspectos eléctricos, conectores y características de los medios de transmisión, que igualmente ya han sido tratados en el capítulo Nº1.

En realidad, este nivel esta íntimamente relacionado con la capa de enlace, donde todos los aspectos (eléctricos, ópticos, etc.) relacionados con el medio ya son considerados en sus especificaciones, tal como se ha podido advertir en el punto anterior. Por este motivo, esta capa no se suele diferenciar a nivel práctico, y suele encontrarse integrada en la anterior.

A.17.- Dispositivos de red dentro del modelo tcp/ip.

En este apartado serán descritos de una manera brebe los elementos y dispositivos de red que juegos un papel esencial dentro de cada uno de los niveles del modelo "TCP/IP":

1) Nivel de Aplicación y Transporte. Dentro de este nivel juega un papel importantísimo los "**FIREWALL**" ("Muro de Fuego"). Estos dispositivos son capaces de realizar un filtrado del tráfico de la red, pudiendo cribar en función del puerto de origen/destino (no dejar pasar aquellos paquetes IP que vayan destinados/provengan a/de un puerto en concreto), ó aplicación concreta.

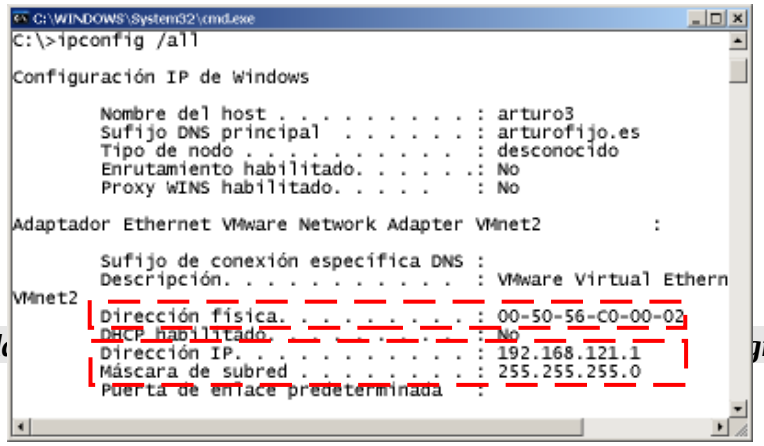
2) Nivel de Red. El elemento esencial dentro de este nivel son los "**ROUTER**". Estos son los dispositivos encargados de encaminar ó enrutar los paquetes IP de tal forma que puedan alcanzar su destino, además de hacer las veces de puerta de enlace ("**GATEWAY**") y conversor de protocolos ("**PASARELA**"). Otro elemento de red que también interviene dentro de este nivel son los "FIREWALLs" que llevan a cabo la función de filtrado en función de la dirección IP de origen/destino (en el caso en que la dirección IP de destino sea la correspondiente a un determinado servidor, no permitir el establecimiento de la comunicación). Normalmente, la mayoría de los "ROUTER" integran funciones de filtrado, lo que le convierten en "ROUTER/FIREWALL". Todos estos aspectos serán tratados en la última práctica propuesta del libro (capítulo Nº 15), donde implementaremos un "ROUTER/FIREWALL" a través de un equipo informático GNU/Linux.

3) Nivel de enlace. Dentro de este nivel juegan un papel fundamental las "**tarjetas**, **interfaces** ó **adaptadores de red**". Estos son los módulos dentro de nuestro equipo informático encargados de controlar el acceso al medio (MAC), junto a un control de los flujo y errores (LLC).

En relación a estas interfaces de red, señalar que poseen un identificador, conocido como "**dirección física/hardware**" ó "**dirección MAC**" que les permite identificarse dentro de una red de ordenadores (no confundir con la dirección IP), y que es utilizada por los protocolos de nivel de enlace para hacer referencia al equipo origen y destino de una comunicación.

Por tanto, esta dirección MAC no es más que un número de identificación único para cada uno de los adaptadores de red que les permite identificarse de forma inequívoca, e indicar a quien va dirigida la información ó quien la envía. Consta de 48 bits (una dirección IP 32 bits) que suelen representarse en forma hexadecimal (12 dígitos). Los 6 primeros dígitos hexadecimales identifican al fabricante y los 6 últimos al adaptador correspondiente de ese fabricante, el cual no puede utilizar dos veces el mismo número.

Mediante el uso de estas direcciones MAC también pueden llevarse a cabo difusiones de mensajes dentro de la red (mensajes destinados a todos los equipos que forman parte de la red) haciendo uso de la dirección "FF-FF-FF-FF-FF-FF".



Al igual que los adaptadores de red, otro tipo de dispositivos que trabajan dentro de este nivel, y que hacen uso de las direcciones MAC anteriores son los "**BRIDGE**" ("**puente**", "switch" de dos puertos) y "**conmutadores**" ó "**SWITCH**". Es decir, estos dispositivos son capaces de tratar las tramas de nivel de enlace, y reconocer quien es equipo destinatario a través de su dirección MAC.

Según todo lo anterior, podemos empezar a preguntarnos el porque de dos identificadores para un mismos equipo: **(1)** identificador hardware a nivel de enlace, conocido como dirección MAC, e **(2)** identificador lógico a nivel de red, conocido como dirección IP. La razón radica en razones de enrutamiento (redes de área extensa). Es decir, si pretendiéramos comunicar dos ordenadores distantes únicamente mediante el uso de direcciones MAC, resultaría prácticamente imposible de establecer dicha comunicación, ya que una dirección MAC no aporta ningún tipo de información en relación a donde se encuentra ubicado geográficamente un equipo (sólo informa de quien fabricó la interfaz y su número de serie), por lo que sería liadísimo establecer la ruta entre el origen y destino (se requerirían tablas de enrutamiento enormes que presentarían una complejidad de cara a su gestión y administración). En cambio una dirección IP se asigna de manera lógica, en función de la ubicación del equipo (es tenido en cuenta por el InterNIC), lo que permite que haga las veces de dirección postal "ciudad-calle-número-puerta" dentro del ámbito informático, lo que permite establecer las rutas entre origen y destino de una manera sencilla.

Resumiendo, las direcciones MAC sólo sirven para comunicar equipos que se encuentran muy próximos (redes LAN), donde la rutas que se necesitan establecer no requiere de complejos algoritmos de enrutamiento.

4) Nivel Físico. Dentro de este nivel se ubican los dispositivos "**HUB**" ó "**repetidores multipuertos**", al comportarse de manera transparente a la información que contienen las tramas de información, limitándose a recoger la señal que circula por el medio de comunicación (canal), amplificarla, y repetirla por todos sus puertos de salida.