Administración de servidores

Remo Suppi Boldrito

PID_00212473

UOC
Universitat Oberta de Catalunya

Índice

Introducción

La interconexión entre máquinas y las comunicaciones de alta velocidad han permitido que los recursos que se utilicen no estén en el mismo sitio geográfico del usuario. UNIX (y por supuesto GNU/Linux) es probablemente el máximo exponente de esta filosofía, ya que desde su inicio ha fomentado el intercambio de recursos y la independencia de dispositivos. Esta filosofía se ha plasmado en algo común hoy en día como son los servicios. Un servicio es un recurso (que puede ser universal o no) y que permite, bajo ciertas condiciones, obtener información, compartir datos o simplemente procesar la información a distancia. Nuestro objetivo es analizar los servicios que permiten el funcionamiento de una red. Generalmente, dentro de esta red existirá una máquina (o varias, según las configuraciones) que hará posible el intercambio de información entre las demás. Estas máquinas se denominan servidores y contienen un conjunto de programas que permiten que la información esté centralizada y sea fácilmente accesible. Estos servicios permiten la reducción de costes y amplían la disponibilidad de la información, pero se debe tener en cuenta que un servicio centralizado presenta inconvenientes, ya que puede quedar fuera de servicio y dejar sin atención a todos los usuarios. En este módulo se verán los principales servicios que permiten que una máquina GNU/Linux juegue un papel muy importante en una infraestructura tecnológica, tanto en centralizar y distribuir datos como en ser punto de información, acceso o comunicación. Por otro lado y con el avance de las arquitecturas (software) orientada a servicios (SOA - *Service Oriented Architecture*), y las tecnologías de desarrollo de aplicaciones que se han estandarizado en este paradigma de diseño de sistemas distribuidos, GNU/Linux se ha transformado en la infraestructura por excelencia que da soporte a la creación de sistemas de información altamente escalables. Este tipo de arquitectura (SOA) se ha transformado en una parte esencial del desarrollo de software distribuido, ya que permite la creación de sistemas distribuidos eficientes, que aprovechan toda la infraestructura subyacente, y establece una interfaz bien definida a la exposición e invocación de servicios web (de forma común pero no exclusivamente) facilitando la interacción entre los sistemas propios y externos.

Servicios replicados

Una arquitectura de servidores debe tener los servicios replicados *(mirrors)* para solventar los inconvenientes que supone.

Objetivos

En los materiales didácticos de este módulo encontraréis los contenidos y las herramientas procedimentales para conseguir los objetivos siguientes:

1. Presentar los aspectos más relevantes de los conceptos involucrados, tanto a nivel teórico como práctico, en la estructura de servidores/servicios en un sistema GNU/Linux.
2. Analizar los conceptos relativos a servicios y servidores específicos de un sistema GNU/Linux.
3. Experimentar con la configuración y adaptar la instalación de servicios a un entorno determinado.
4. Analizar y participar en discusiones sobre las posibilidades actuales y futuras de nuevos servicios y los obstáculos que existen básicamente en aspectos de seguridad en los diferentes entornos de trabajo GNU/Linux (servidor, escritorio multimedia, escritorio ofimàtica, enrutador o *router*,...).

1. Administración de servidores

Los servicios se pueden clasificar en dos tipos: de vinculación ordenador-ordenador o de relación hombre-ordenador. En el primer caso, se trata de servicios requeridos por otros ordenadores, mientras que en el segundo, son servicios requeridos por los usuarios (aunque hay servicios que pueden actuar en ambas categorías). Dentro del primer tipo se encuentran servicios de nombres, como el *Domain Name System* (DNS), el servicio de información de usuarios (NIS-YP), el directorio de información LDAP o los servicios de almacenamiento intermedio (*proxies*). Dentro de la segunda categoría se contemplan servicios de conexión interactiva y ejecución remota (ssh, telnet), transferencia de ficheros (ftp), intercambio de información a nivel de usuario, como el correo electrónico (MTA, IMAP, POP), *news*, *World Wide Web*, *Wiki* y archivos (NFS). Para mostrar las posibilidades de GNU/Linux Debian-FC, se describirá cada uno de estos servicios con una configuración mínima y operativa, pero sin descuidar aspectos de seguridad y estabilidad.

1.1. Sistema de nombres de dominio (*Domain Name System, DNS*)

La funcionalidad del servicio de DNS es convertir nombres de máquinas (legibles y fáciles de recordar por los usuarios) en direcciones IP o viceversa.

A la consulta de cuál es la IP de *nteum.remix.cat*, el servidor responderá 192.168.0.1 (esta acción es conocida como *mapping*); del mismo modo, cuando se le proporcione la dirección IP, responderá con el nombre de la máquina (conocido como *reverse mapping*).

El *Domain Name System* (DNS) es una arquitectura arborescente que evita la duplicación de la información y facilita la búsqueda. Por ello, un único DNS no tiene sentido sino como parte del árbol. Una de las aplicaciones más utilizadas que presta este servicio se llama *named*, se incluye en la mayoría de distribuciones de GNU/Linux (`/usr/sbin/named`) y forma parte de un paquete llamado `bind` (actualmente versión 9.x), coordinado por el ISC (Internet Software Consortium). El DNS es simplemente una base de datos, por lo cual, es necesario que las personas que la modifiquen conozcan su estructura, ya que, de lo contrario, el servicio quedará afectado. Como precaución debe tenerse especial cuidado en guardar las copias de los archivos para evitar cualquier in-

terrupción en el servicio. Los servidores DNS, que pueden convertir la mayoría de los nodos de DNS en su IP correspondiente y se les llama servidores DNS recursivos. Este tipo de servidor no puede cambiar los nombres de los nodos de DNS que hay, simplemente se preguntan otros servidores DNS la IP de un nodo DNS dado. Los servidores DNS autorizados pueden gestionar/cambiar las direcciones IP de los nodos DNS que gestionan y normalmente son con los cuales contactará un servidor DNS recursivo con el fin de conocer la IP de un nodo DNS dado. Una tercera variante de los servidores DNS son los DNS caché, que simplemente almacenan la información obtenida de otros servidores DNS recursivos.

1.1.1. Servidor de nombres caché

En primer lugar, se configurará un servidor de DNS para resolver consultas, que actúe como caché para las consultas de nombres *resolver, caching only server*). Es decir, la primera vez consultará al servidor adecuado porque se parte de una base de datos sin información, pero las veces siguientes responderá el servidor de nombres caché, con la correspondiente disminución del tiempo de respuesta. Para instalar un servidor de esta características, instalamos los paquetes `bind9 y dnsutils` (p. ej., en Debian `apt-get bind9 dnsutils`).

Como se puede observar en el directorio */etc/bind* encontraremos una serie de archivos de configuración entre ellos el principal es named.conf que incluye a otros *named.conf.options, named.conf.local, named.conf.default-zones* (podemos encontrar que esta configuración puede variar entre distribuciones pero es similar). El archivo */etc/bind/named.conf.options* contiene las opciones generales para el servidor y los otros dos archivos nos servirán para configurar nuestras zonas en el servidor de nombres que veremos en el apartado siguiente. Para configurar nuestro *caching only server* debemos considerar que las pregunta en primer lugar las resolveremos nosotros pero como es evidente que no tenemos la respuesta se deberá redirigir la pregunta a servidores de DNS en la red. Para ello, es interesante considerar los de servicios como OpenDNS (http://www.opendns.com/opendns-ip-addresses/) que son las IP: 208.67.222.222 y 208.67.220.220 o bien servicios como los de Google (https://developers.google.com/speed/public-dns/?csw=1) que responden a las direcciones IP: 8.8.8.8 y 8.8.4.4. A continuación modificamos el archivo */etc/bind/named.conf.options* para quitar los comentarios de la sección *forwarders* poniendo lo siguiente:

```
forwarders {
    // OpenDNS servers
     208.67.222.222;
     208.67.220.220;
    // Podríamos incluir nuestro ISP/router -verificar la IP-
     192.168.1.1;
    };
```

Se podrían agregar en el mismo archivo al final y reemplazando las que hay (se puede dejar esta configuración para un segundo paso) opciones de seguridad para que solo resuelva las peticiones de nuestra red:

```
// Security options
 listen-on port 53 { 127.0.0.1; 192.168.1.100; };
 allow-query { 127.0.0.1; 192.168.1.0/24; };
 allow-recursion { 127.0.0.1; 192.168.1.0/24; };
 allow-transfer { none; };
// Las siguientes dos opciones ya se encuentran por defecto en el
// archivo pero si no tenemos IPv6 podemos comentar la última.
 auth-nxdomain no; # conform to RFC1035
 // listen-on-v6 { any; };
```

Se puede verificar la configuración con `named-checkconf`. A continuación modificamos el archivo */etc/resolv.conf* para añadir `nameserver 127.0.0.1` para que las preguntas a DNS la librería *gethostbyname(3)* la realice al propio *host* y el archivo */etc/nsswitch.conf* verificando que la línea *hosts* quede como: `hosts: files dns` indicará el orden que se resolverán las peticiones (primero localmente al */etc/hosts* y luego al servidor DNS). Por último solo resta reiniciar el servidor con `service bind9 restart` y hacer las pruebas de funcionamiento. En nuestro caso hemos ejecutado `dig www.debian.org` -se han omitido líneas para resumir la información-:

```
; «» DiG 9.8.4-rpz2+rl005.12-P1 «» www.debian.org
...
;; QUESTION SECTION:
;www.debian.org. IN A

;; ANSWER SECTION:
www.debian.org. 300 IN A 130.89.148.14
www.debian.org. 300 IN A 5.153.231.4
...
;; Query time: 213 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
...
```

Donde podemos observar que la *query* ha tardado 213 milisegundos. Si la realizamos por segunda vez (ya ha quedado la consulta almacenada en nuestro servidor caché):

```
; «» DiG 9.8.4-rpz2+rl005.12-P1 «» www.debian.org
...
;; QUESTION SECTION:
;www.debian.org. IN A

;; ANSWER SECTION:
www.debian.org. 293 IN A 5.153.231.4
www.debian.org. 293 IN A 130.89.148.14
...
;; Query time: 1 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
...
```

En este caso la consulta ha tardado 1 milisegundo verificando que el servidor caché ha funcionado. También podemos hacer la consulta inversa con `nslookup 130.89.148.14` lo cual nos dará el nombre de la máquina que

tiene asignada esta IP (en este caso es el servidor al cual está asignado el dominio www.debian.org)

```
Server: 127.0.0.1
Address: 127.0.0.1#53
Non-authoritative answer:
14.148.89.130.in-addr.arpa name = klecker4.snt.utwente.nl.
Authoritative answers can be found from:
. nameserver = l.root-servers.net.
. nameserver = b.root-servers.net.
. nameserver = j.root-servers.net.
...
```

Es importante considerar si se desea instalar el servidor de DNS en una máquina virtual, al ser un servicio que se debe acceder desde fuera, el servidor DNS (y la mayoría de los servidores) no puede estar en NAT sobre un *host* sino que debe tener IP propia del segmento de red en la cual presta el servicio y por lo tanto el adaptador de red de la máquina virtual debe estar en modo puente (*bridged*). Para configurar los clientes dentro de nuestra red bastará con modificar el archivo */etc/resolv.conf* si son GNU/Linux (o modificar */etc/network/interfaces* para agregarlos como `dns-nameservers IP` si se tiene instalado el paquete `resolvconf`) y en Windows modificar las propiedades IPv4 del adaptador para introducir la IP de nuestro servidor DNS caché.

1.1.2. Configuración de un dominio propio

DNS posee una estructura en árbol y el origen es conocido como "." (ver `/etc/bind/db.root`). Bajo el "." existen los TLD (*Top Level Domains*) como **org**, **com**, **edu**, **net**, etc. Cuando se busca en un servidor, si este no conoce la respuesta, se buscará recursivamente en el árbol hasta encontrarla. Cada "." en una dirección (por ejemplo, nteum.remix.cat) indica una rama del árbol de DNS diferente y un ámbito de consulta (o de responsabilidad) diferente que se irá recorriendo en forma recursiva de izquierda a derecha.

Otro aspecto importante, además del dominio, es el `in-addr.arpa` (*inverse mapping*), el cual también está anidado como los dominios y sirve para obtener nombres cuando se consulta por la dirección IP. En este caso, las direcciones se escriben al revés, en concordancia con el dominio. Si nteum.remix.world es la 192.168.0.30, entonces se escribirá como 30.0.168.192, en concordancia con nteum.remix.world para la resolución inversa de IP ->nombre. En este ejemplo utilizaremos un servidor de DNS atiende a una red interna (192.168.0.0/24) y luego tiene una IP pública pero a a fines del ejemplo y poder hacer las pruebas internas la configuraremos en un IP privada de otro segmento (172.16.0.80) y un dominio ficticio (remix.world) pero en el caso de llevar este servidor a producción se debería cambiar por la IP externa que tenga y el dominio correspondiente que se haga obtenido y tengamos a nuestra disposición.

Para configurar un servidor de nombres propio en primer lugar cambiaremos el archivo */etc/bind/named.conf* para definir dos zonas (una interna y otra ex-

terna). El archivo quedará como (observad que hemos comentado la línea que incluye named.conf.default-zones ya que la incluiremos después):

```
include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
//include "/etc/bind/named.conf.default-zones";
include "/etc/bind/named.conf.internal-zones";
include "/etc/bind/named.conf.external-zones";
```

Modificamos el archivo *named.conf.internal-zones* para indicarle los archivos donde se encontrarán realmente los nombres de las máquinas que en este caso será dos: uno para para resolución nombre ->IP llamado remix.world.lan y otro para la resolución inversa IP ->nombre llamado 0.168.192.db.

```
view "internal" {
    match-clients {
        localhost;
        192.168.0.0/24;
    };
// set zone for internal
    zone "remix.world" {
        type master;
        file "/etc/bind/remix.world.lan";
        allow-update { none; };
    };
// set zone for internal reverse
    zone "0.168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/0.168.192.db";
        allow-update { none; };
    };
include "/etc/bind/named.conf.default-zones";
};
```

De la misma forma modificamos el archivo *named.conf.external-zones* para indicarle los archivos que resolverán esta zona y que también serán dos: remix.world.wan y 80.0.16.172.db para la resolución directa e inversa respectivamente.

```
view "external" {
// define for external section
    match-clients { any; };
// allow any query
    allow-query { any; };
// prohibit recursion
    recursion no;
// set zone for external
    zone "remix.world" {
        type master;
        file "/etc/bind/remix.world.wan";
        allow-update { none; };
    };
// set zone for external reverse
    zone "80.0.16.172.in-addr.arpa" {
        type master;
        file "/etc/bind/80.0.16.172.db";
        allow-update { none; };
    };
};
```

También se debe modificar el *named.conf.options*:

```
options {
    directory "/var/cache/bind";
// ...
// forwarders {
// 158.109.0.9;
// 158.109.0.1;
// };
// query range you permit
    allow-query { localhost; 192.168.0.0/24; };
// the range to transfer zone files
    allow-transfer { localhost; 192.168.0.0/24; };
// recursion range you allow
// allow-recursion { localhost; 192.168.0.0/24; };


    dnssec-validation auto;
    auth-nxdomain no; # conform to RFC1035
//listen-on-v6 { any; };
};
```

Ahora es necesario definir los archivos que realmente resolverán las IP/nombres de las zonas. Comenzamos por *remix.server.lan*:

```
$TTL 86400
@ IN SOA nteum.remix.world. root.remix.world. (
    2013050601 ;Serial
    3600 ;Refresh
    1800 ;Retry
    604800 ;Expire
    86400 ;Minimum TTL
)

    IN NS nteum.remix.world.
    IN A 192.168.0.30
    IN MX 10 nteum.remix.world.

nteum IN A 192.168.0.30
```

Se debe tener en cuenta el "." al final de los nombres de dominio. El significado de los registros indican: **SOA** (*Start of Authority*) debe estar en todos los archivos de zona al inicio, después de `TTL` (*Time to Live*) que indica el tiempo en segundos que los recursos de una zona serán válidos, el símbolo `@` significa el origen del dominio; **NS**, el servidor de nombres para el dominio, **MX** (*Mail eXchange*), indica a donde se debe dirigir el correo para una determinada zona, y **A** es la Ip que se debe asignar a un nombre. Para el archivo de resolución inversa *0.168.192.db*:

```
$TTL 86400
@ IN SOA nteum.remix.world. nteum.remix.world. (
    2013050601 ;Serial
    3600 ;Refresh
    1800 ;Retry
    604800 ;Expire
    86400 ;Minimum TTL
)

    IN NS nteum.remix.world.
    IN PTR remix.world.
    IN A 255.255.255.0
30 IN PTR nteum.remix.world.
```

Donde podemos observar que los registros tienen un formato como el siguiente: `<último-dígito-IP>IN PTR <FQDN-of-system>` donde PTR (*Registro PoinTeR*) crea un apuntador hacia el nombre de la IP que coincide con el registro. De la misma forma procedemos para la zona externa con el archivo *remix.world.wan*:

```
$TTL 86400
@ IN SOA nteum.remix.world. root.remix.world. (
    2013050601 ;Serial
    3600 ;Refresh
    1800 ;Retry
    604800 ;Expire
    86400 ;Minimum TTL
)

    IN NS nteum.remix.world.
    IN A 172.16.0.82
    IN MX 10 nteum.remix.world.
nteum IN A 172.16.0.82
```

Y su resolución inversa *80.0.16.172.db*:

```
$TTL 86400
@ IN SOA nteum.remix.world. nteum.remix.world. (
    2013050601 ;Serial
    3600 ;Refresh
    1800 ;Retry
    604800 ;Expire
    86400 ;Minimum TTL
)

    IN NS nteum.remix.world.
    IN PTR remix.world.
    IN A 255.255.255.248
82 IN PTR nteum.remix.world.
```

Ahora solo restaría cambiar el archivo */etc/resolv.conf* para incluir `nameserver 192.168.0.30` y reiniciar el servidor `service bind9 restart`. Se debe mirar */var/log/syslog* para observar si hay errores en el arranque del servidor y corregirlos hasta que su reinicio esté libre de estos (los más comunes son los símbolos utilizados como comentarios, los puntos al final de dominio o espacios antes de los registros A-PTR). Posteriormente podemos probar con `dig nteum.remix.world` y tendremos una salida como:

```
; «» DiG 9.8.4-rpz2+rl005.12-P1 «» nteum.remix.world
...
;; QUESTION SECTION:
;nteum.remix.world. IN A
;; ANSWER SECTION:
nteum.remix.world. 86400 IN A 192.168.0.30
;; AUTHORITY SECTION:
remix.world. 86400 IN NS nteum.remix.world.
;; Query time: 0 msec
...
```

Si hacemos la petición inversa con `dig -x 192.168.0.30` tendremos:

```
...
;; QUESTION SECTION:
;30.0.168.192.in-addr.arpa. IN PTR
;; ANSWER SECTION:
30.0.168.192.in-addr.arpa. 86400 IN PTR nteum.remix.world.
;; AUTHORITY SECTION:
0.168.192.in-addr.arpa. 86400 IN NS nteum.remix.world.
;; ADDITIONAL SECTION:
nteum.remix.world. 86400 IN A 192.168.0.30
...
```

Un aspecto interesante es poner alias lo cual significa incluir un registro CNAME en *remix.world.lan* como `ftp IN CNAME nteum.remix.world.` Dado que es un servicio esencial es necesario muchas veces disponer de un servicio secundario (*slave*) y bind9 permite crear muy fácilmente servicios de este tipo a partir del servidor primario por transferencia de las tablas (http://www.server-world.info/en/note?os=Debian_7.0&p=dns&f=5).

1.1.3. Otros DNS

MaraDNS es un servidor DNS que puede funcionar como DNS recursivo (caché) o como *authoritative name server* y que se caracteríza por su extremadamente fácil configuración. MaraDNS ha sido optimizado para un pequeño número de dominios en forma simple y eficiente y que permite crear un dominio propio y servirlo sin grandes esfuerzos aunque contiene todos los elementos para configurarlo en forma similar a bind9. En su diseño se ha puesto especial cuidado a la seguridad y utiliza una librería de gestión de cadenas de caracteres que resisten a los principales ataques a los DNS por desbordamiento (*buffer overflows*)[6]. EL paquete MaraDNS incluye un DNS autorizado (`maradns`), un servidor DNS recursivo con posibilidades de caché y que se llama Deadwood. La decisión de poner un servidor autorizado o recursivo dependerá de la función que se busque. Si simplemente es para obtener otros sitios en Internet se deberá configurar un servidor recursivo, en cambio se se desea registrar dominios y se tienen una red de ordenadores dentro de este dominio hay que configurar un servidor DNS con autoridad.

Como pruebas de este servicio configuraremos diferentes opciones de MaraDNS para Debian. Si bien el paquete está incluido en el repositorio* es una versión que tiene problemas de seguridad y su autor recomienda bajar el código fuente y compilarlo/instalarlo. Para ello debemos obtener el archivo desde http://maradns.samiam.org/download.html, descomprimirlo (`tar xjvf maradns-x.x.xx.tar.bz2` donde x.x.xx es la versión del software descargado) y dentro de directorio ejecutar `./configure; make` y si no hay errores ejecutar `make install`. Los servidores se instalarán en */usr/local/sbin* y otros archivos complementarios en */usr/local/bin*. Los archivos de configuración en */etc/mararc y /etc/dwood3rc* y el directorio de configuración */etc/maradns*.

***http://maradns.samiam.org/debian.html**

En primer lugar para configurar un DNS recursivo y caché modificar el archivo */etc/dwood3rc* para incluir:

```
bind_address="127.0.0.1"
chroot_dir = "/etc/maradns"
#upstream_servers = {}
#upstream_servers["."]="8.8.8.8, 8.8.4.4"
recursive_acl = "127.0.0.1/16" # Quien puede utilizar la caché

# Parámetros de configuración del servidor
maxprocs = 8 # Maximum number of pending requests
handle_overload = 1 # Send SERVER FAIL when overloaded
maradns_uid = 99 # UID Deadwood runs as
maradns_gid = 99 # GID Deadwood runs as
maximum_cache_elements = 60000

# File cache (readable and writable by the maradns_uid/gid)
cache_file = "dw_cache"
```

También es posible que Deadwood pueda contactar con otros DNS recursivos antes que con los root DNS servers. Para ello quitar el comentario de las líneas upstream_servers (en este caso se han puestos los dns públicos de Google). Luego podremos ejecutar `/usr/local/sbin/Deadwood` o si se quiere ejecutar como *daemon* se deberá utilizar el programa `duende` incluido en el paquete: `/usr/local/bin/duende /usr/local/sbin/Deadwood`.

Para configurar un DNS autorizado debemos modificar el archivo */etc/mararc* con los siguiente valores: cvs2 el dominio que gestionará este servidor y el archivo que tendrá la información en este caso db.local, el directorio donde se encontrará los archivos de configuración (/etc/maradns) y la IP de donde responderá.

```
csv2 = {}
csv2["remix.world."] = "db.local"
ipv4_bind_addresses = "127.0.0.1"
chroot_dir = "/etc/maradns"
```

El archivo /etc/maradns/db.local simplemente tendrá una línea por cada registro que deseamos que transforme el dns por ejemplo, `nteum.remix.world. 192.168.0.30 ~`. Luego se debe poner en marcha el servidor* o si se desea ejecutar como *daemon* habrá que usar el programa `/usr/local/bin/duende /usr/local/sbin/maradns`. Una de las cuestiones interesantes es cómo puedo dar servicio de forma *authoritative* para mis dominios en una Intranet pero además ser recursivo cuando la petición sea a dominios externos? Es decir tengo unos pocos nombres (FQDN) que pertenecen a mi dominio (por ejemplo remix.world) sobre mi Intranet y deseo al mismo tiempo asegurar la resolución de dominio FQDN externos (por ejemplo debian.org). La solución planteada por el autor (S. Trenholme**) pasa por una configuración combinada de MaraDNS 2.x y Deadwood 3.x poniendo MaraDNS que escuche sobre localhost (127.0.0.1) mientras Deadwood sobre la IP del servidor (192.168.1.37 en nuestro caso). Los archivos de configuración */etc/mararc* y */etc/dwood3rc* serán:

***/usr/local/sbin/maradns**

****http://woodlane.webconquest.com/pipermail/list/2011-August/000928.html**

```
# Archivo /etc/mararc
# Simplemente indicar que escuche sobre localhost y el dominio que será responsable
ipv4_bind_addresses = "127.0.0.1"
timestamp_type = 2
verbose_level = 3
hide_disclaimer = "YES"
csv2 = {}
csv2["remix.world."] = "db.local"
chroot_dir = "/etc/maradns"

#Archivo /etc/dwood3rc
# Indicarle que además de nuestro dominio donde queremos consultar el resto
root_servers = {}
root_servers["remix.world."] = "127.0.0.1"
root_servers["."]="198.41.0.4, 192.228.79.201, 192.33.4.12, 199.7.91.13,"
root_servers["."]+="192.203.230.10, 192.5.5.241, 192.112.36.4, 128.63.2.53, "
root_servers["."]+="192.36.148.17, 192.58.128.30, 193.0.14.129, 199.7.83.42, "
root_servers["."]+="202.12.27.33"
# Donde escuchará la peticiones = IP del servidor Deadwood
bind_address="192.168.1.37"
# Habilitar la respuesta de IP privadas
filter_rfc1918=0
# IPs que podrán hacer peticiones
recursive_acl = "192.168.1.0/24"
# Caché de disco para Deadwood
cache_file = "dw_cache"
# Directorio raíz
chroot_dir = "/etc/maradns"
```

Finalmente en nuestras máquinas poner como */etc/resolv.conf* la IP de nuestro servidor `nameserver 192.168.1.37` y poner en marcha ambos servidores (luego de verificar que funciona se pueden poner como *daemons* y bajar el nivel de log con `verbose_level = 1`.

Otro servidor interesante que viene en muchas distribuciones es **Dnsmasq** que permite en forma simple configurar un servidor DNS (*forwarder*) y un servidor DHCP en el mismo paquete para redes pequeñas/medianas. este puede atender peticiones de nombres de máquinas que no están en los DNS globales e integra el servidor de DHCP para permitir que máquinas gestionadas por el DHCP aparezcan en el DNS con nombres ya configurados y además soporta gestión de IP dinámicas y estáticas para el DHCP y permite el acceso BOOTP/TFTP para máquinas sin disco y que hacen su inicialización por red.

La forma de configurar rápidamente un DNS para máquinas de dominio propio y que haga de *forwarder* para los dominios externos es `apt-get update; apt-get install dnsmasq`. Si lo que se desea es un simple DNS ya está configurado teniendo en cuenta que en */etc/resolv.conf* tendremos alguna cosa como `nameserver 8.8.8.8`. Con esto podemos probar los externos con `dig debian.org @localhost` (o simplemente `dig debian.org`) pero también responderá a todas las máquinas que tengamos definidas en */etc/hosts* como por ejemplo si tenemos una línea como `192.168.1.37 nteum.remix.world nteum` podremos ejecutar `dig nteum @localhost` y nos responderá con la IP correspondiente. Es por ello que si la red es simple podemos agregar las máquinas en este archivo pero si es más compleja podemos utilizar el archivo de configuración */etc/dnsmasq.conf* que incluye una gran cantidad de opciones para organizar los dominios internos y otros parámetros no solo para la configuración del DNS sino también para el servidor de DHCP.[7]

1.2. NIS (YP)

Con el fin de facilitar la administración y dar comodidad al usuario en redes de diferentes tamaños que ejecutan GNU/Linux (o algún otro sistema operativo con soporte para este servicio), se ejecutan servicios de *Network Information Service*, NIS (o *Yellow Pages*, YP, en la definición original de Sun). GNU/Linux puede dar apoyo como cliente/servidor de NIS. La información que se puede distribuir en NIS es: usuarios (*login names*), contraseñas (*passwords*, `/etc/passwd`), directorios de usuario (*home directories*) e información de grupos (*group information*, `/etc/group`), redes, hosts, etc., lo cual presenta la ventaja de que, desde cualquier máquina cliente o desde el mismo servidor, el usuario se podrá conectar con la misma cuenta y contraseña y al mismo directorio (aunque el directorio deberá ser montado anteriormente sobre todas las máquinas cliente por NFS o mediante el servicio de `automount`). [10, 11]

La arquitectura NIS es del tipo cliente-servidor, es decir, existe un servidor que dispondrá de todas las bases de datos y unos clientes que consultarán estos datos en forma transparente para el usuario. Por ello, se debe pensar en la posibilidad de configurar servidores "de refuerzo" (llamados secundarios) para que los usuarios no queden bloqueados ante la caída del servidor principal. Es por ello que la arquitectura se denomina realmente de múltiples servidores (*master+mirrors-clients*).

1.2.1. ¿Cómo iniciar un cliente local de NIS en Debian?

Un cliente local es el que anexa el ordenador a un dominio NIS ya existente: primero se debe verificar que se tienen instalados los paquetes `netbase` (red básica TCP/IP) y `rpcbind` (servidor que convierte números RPC –*Remote Procedure Call*– en puertos DARPA) y `nis` (específico). Es importante destacar que el paquete `rpcbind`, que ya está en la mayoría de las distribuciones, es un servicio que sustituye al tradicional `portmap`. Este ha quedado obsoleto por diferentes causas vinculadas a los cambios en el kernel de NFSV3/V4, pero especialmente, al no tener soporte para IPv6. Es necesario verificar la instalación de los dos primeros paquetes para todos los programas que ejecutan RPC, incluyendo NFS y NIS. Se recomienda usar el comando `apt-get` (o también `dpkg, aptitude, synaptic-pkexec`) y se puede verificar si está instalado con `apt-cache pkgnames` en modo texto.

El procedimiento de instalación del paquete NIS solicitará un dominio (`NIS domainname`). Este es un nombre que describirá el conjunto de máquinas que utilizarán el NIS (no es un nombre de *host*). Hay que tener en cuenta que `NISNteum` es diferente de `Nisnteum` como nombre de dominio. Para configurarlo, se puede utilizar el comando `nisdomainname`, dominio que se almacenará en `/proc/sys/kernel/domainname`. También se puede reconfigugar el paquete con `dpkg-reconfigure nis` y volveremos a comenzar, po-

demos ver que el rpcbind está en marcha con `ps -edaf | grep rpcbind` o también con `rpcinfo -p` y nos mostrará diferentes líneas -para diferentes programas, puertos/protocolos, versiones- como portmapper. Para reiniciar el rpcbind, podemos hacer `service rpcbind restart`. Es importante tener en cuenta que la primera vez que instalamos en NIS, que será el cliente en nuestro caso, el sistema de instalación intentará poner los daemons en marcha pero fallará, ya que no tenemos ningún servidor configurado (que será el siguiente paso) y dará un mensaje de *[....] Starting NIS services: ypbind[....] binding to YP server...failed (backgrounded)*.

El cliente NIS usa el comando `ypbind` para encontrar un servidor para el dominio especificado, ya sea mediante `broadcast` (no aconsejado por inseguro) o buscando el servidor indicado en el archivo de configuración `/etc/yp.conf` (recomendable) que puede tener básicamente dos tipos de configuración como las mostradas abajo -solo se debe indicar una de ellas aunque es recomendable la primera-.

Sintaxis del archivo `/etc/yp.conf`

`domain nisdomain server hostname`: indica que se utiliza el *hostname* para el dominio `nisdomain`. En nuestro caso podría ser `domain NISnteum server 192.168.1.30` Se podría tener más de una entrada de este tipo para un único dominio.

`ypserver hostname`: indica que se utiliza *hostname* como servidor introduciendo la dirección IP del servidor NIS. Si se indica el nombre, asegúrese de que se puede encontrar la IP por DNS o que la misma figura en el archivo `/etc/hosts`, ya que, de otro modo, el cliente se bloqueará.

Para iniciar el servicio se debe ejecutar:

```
/etc/init.d/nis stop      para detenerlo
/etc/init.d/nis start     para iniciarlo
```
o también con el comando `service nis start|stop`

A partir de este momento, el cliente NIS estará funcionando. Ello se puede confirmar con `rpcinfo -u localhost ypbind`, que mostrará las dos versiones del protocolo activo o, cuando el servidor esté funcionando, se puede utilizar el comando `ypcat mapname` (por ejemplo, `ypcat passwd`, que mostrará los usuarios NIS definidos en el servidor) donde las relaciones entre *mapnames* y las tablas de la base de datos NIS están definidas en `/var/yp/nicknames`.

1.2.2. ¿Qué recursos se deben especificar para utilizar el NIS?

A partir de la inclusión de lib6 en las distribuciones de GNULinux (esto es Debian5 o FC4) es necesario orientar la consulta del *login* a las bases de datos adecuadas con los pasos siguientes:

1) Verificar el fichero `/etc/nsswitch.conf` y asegurarse de que las entradas `passwd`, `group`, `shadow` y `netgroup` son similares a:

```
passwd: compat nis
group: compat nis
shadow: compat nis
netgroup: nis
hosts: files dns nis
```

2) Consultad la sintaxis de este archivo en `man nsswitch.conf`.

3) En algunas configuraciones puede ser necesario agregar al final del fichero */etc/pam.d/common-session* (si no existe ya la línea) `session optional pam_mkhomedir.so skel=/etc/skel umask=077` para crear de forma automática el directorio home si no existe pero no es lo habitual ya que generalmente se desea que este se monte del servidor NFS.

4) Con esta configuración se podrá realizar una conexión local (sobre el cliente NIS) a un usuario que no esté definido en el fichero `/etc/passwd`, es decir, un usuario definido en otra máquina (*ypserver*). Por ejemplo, se podría hacer `ssh -l user localhost`, donde `user` es un usuario definido en *ypserver*.

1.2.3. ¿Cómo se debe configurar un servidor?

Antes de instalar el servidor, hay que tener instalado el paquete nis y rpcbind sobre la máquina que hará de servidor (`apt-get install nis rpcbind`) y configurar el dominio -el mismo que hemos introducido cuando configuramos el cliente (NISnteum en nuestro caso). Luego se debe modificar el archivo */etc/default/nis* para modificar la línea `NISSERVER=master` para indicarle el rol de servidor. También (si bien se puede hacer en un segundo paso) se debe ajustar el archivo */etc/ypserv.securenets* para modificar la línea que indica `0.0.0.0 0.0.0.0` que da acceso a todo el mundo y restringirlo a nuestra red, por ejemplo `255.255.255.0 192.168.1.0`. Por último, se debe modificar el */etc/hosts* para que tengamos la máquina con el nombre definido en */etc/hostname* (se puede cambiar/visualizar con el comando `hostname` o editando el fichero) con una línea FQND como por ejemplo `192.168.1.30 nteum.remix.world nteum`. La configuración del servidor se realiza con el comando `/usr/lib/yp/ypinit -m`; en algunas distribuciones es necesario verificar que existe el archivo `/etc/networks`, que es imprescindible para este *script*. Si este archivo no existe, cread uno vacío con `touch/etc/networks`; este *script* nos solicitará cuáles serán los servidores NIS indicándonos la máquina local por defecto y deberemos terminar con Crtl+D. También se puede ejecutar el cliente `ypbind` sobre el servidor; así, todos los usuarios entran por NIS indicando en el archivo */etc/default/nis* que exista `NISCLIENT=true`.

Considerad que a partir de este momento los comandos para cambiar la contraseña o la información de los usuarios, como `passwd`, `chfn` o `adduser`, no son válidos. En su lugar, se deberán utilizar comandos tales como `yppasswd`, `ypchsh` y `ypchfn`. Si se cambian los usuarios o se modifican los archivos men-

cionados, habrá que reconstruir las tablas de NIS ejecutando el comando `make` en el directorio `/var/yp` para actualizar las tablas.

La configuración de un servidor esclavo es similar a la del maestro, excepto si `NISSERVER = slave` en `/etc/default/nis`. Sobre el maestro se debe indicar que distribuya las tablas automáticamente a los esclavos, poniendo `NOPUSH = "false"` en el archivo `/var/yp/Makefile`. Ahora se debe indicar al maestro quién es su esclavo por medio de la ejecución de:

```
/usr/lib/yp/ypinit -m
```

e introduciendo los nombres de los servidores esclavos. Esto reconstruirá los mapas pero no enviará los archivos a los esclavos. Para ello, sobre el esclavo, ejecutad:

```
/etc/init.d/nis stop
/etc/init.d/nis start
/usr/lib/yp/ypinit -s nombre_master_server
```

Así, el esclavo cargará las tablas desde el maestro. También se podría poner en el directorio `/etc/cron.d` el archivo NIS con un contenido similar a (recordad hacer un `chmod 755 /etc/cron.d/nis`):

```
20 * * * * root /usr/lib/yp/ypxfr_1perhour >/dev/null 2>&1
40 6 * * * root /usr/lib/yp/ypxfr_1perday >/dev/null 2>&1
55 6,18 * * * root /usr/lib/yp/ypxfr_2perday >/dev/null 2>&1
```

Con ello, se garantizará que todos los cambios del maestro se transfieran al servidor NIS esclavo.

Iniciad el servidor ejecutando primero el comando `/etc/init.d/nis stop` y luego `/etc/init.d/nis start`. Esta sentencia iniciará el servidor (`ypserv`) y el *password daemon* (`yppasswdd`), cuya activación se podrá consultar con `ypwich -d domain`.

Actualización de las tablas NIS

Es recomendable que después de usar `adduser o useradd` para agregar un nuevo usuario sobre el servidor, ejecutéis `cd /var/yp; make` para actualizar las tablas NIS (y siempre que se cambie alguna característica del usuario, por ejemplo la palabra clave con el comando `passwd`, solo cambiará la contraseña local y no la de NIS).

Para probar que el sistema funciona y que el usuario dado de alta está en el NIS, podéis hacer `ypmatch userid passwd`, donde `userid` es el usuario dado de alta con `adduser` antes y después de haber hecho el `make`. Para verificar el funcionamiento del sistema NIS podéis usar el *script* de http://tldp.org/HOWTO/NIS-HOWTO/verification.html que permite una verificación más detallada del NIS.

También se puede probar en la misma máquina que un usuario se puede conectar a un domino NIS haciendo: `adduser usuario` y respondiendo a las preguntas, luego `cd /var/yp; make`. En este momento lo veremos tanto en */etc/passwd* como en NIS con `ypcat passwd`. Luego lo podemos borrar con el comando `userdel usuario` (no utilizar el comando `deluser` sino el `userdel`) que solo lo borrará del */etc/passwd* y no del NIS (hay que tener cuidado de no hacer un `make`, ya que este también lo borrará del NIS). Luego nos podemos conectar como `ssh usuario@localhost` y con ello podremos verificar que el NIS funciona, ya que el acceso solo se podrá hacer por NIS porque por */etc/passwd* no existe este usuario. Con relación a NIS+ su desarrollo fue parado en el 2012 dada la falta de interés/recursos de la comunidad, no así NIS/YP, que sigue actualizado y presente en todas las distribuciones (http://www.linux-nis.org/nisplus/)

1.3. Servicios de conexión remota: telnet y ssh

1.3.1. Telnet y telnetd

Telnet es un comando (cliente) utilizado para comunicarse interactivamente con otro *host* que ejecuta el *daemon* `telnetd`. El comando `telnet` se puede ejecutar como `telnet host` o interactivamente como `telnet`, el cual pondrá el *prompt* "telnet>" y luego, por ejemplo, `open host`. Una vez establecida la comunicación, se deberá introducir el usuario y la contraseña bajo la cual se desea conectar al sistema remoto. Se dispone de diversos comandos (en modo interactivo) como `open`, `logout` y `mode` (se deben definir las características de visualización), `close`, `encrypt`, `quit`, `set` y `unset` o se pueden ejecutar comando externos con "!". Se puede utilizar el archivo `/etc/telnetrc` para definiciones por defecto o `.telnetrc` para definiciones de un usuario particular (deberá estar en el directorio *home* del usuario).

El *daemon* `telnetd` es el servidor de protocolo telnet para la conexión interactiva. Generalmente, es el *daemon* `inetd` que pone en marcha `telned`; se recomienda incluir un `wrapper tcpd` (que utiliza las reglas de acceso en `host.allow` y `host.deny`) en la llamada al `telnetd` dentro del archivo `/etc/inetd.conf`. Para incrementar la seguridad del sistema se debería, por ejemplo, incluir una línea como:

```
telnet stream tcp nowait telnetd.telenetd /usr/sbin/tcpd /usr/bin/in.telnetd
```

En las distribuciones actuales, la funcionalidad de `inetd` se ha reemplazado por la de `xinetd`, el cual requiere configurar el archivo `/etc/xinetd.conf` que se ha explicado en el apartado de configuración de red de la asignatura *Administración GNU/Linux*. Si se quiere poner en marcha `inetd` a modo de pruebas, se puede usar la sentencia `/etc/init.d/inetd.real start`. Si el archivo `/etc/uissue.net` está presente, el `telnetd` mostrará su contenido al inicio de la sesión. También se puede usar `/etc/security/access.conf` para habilitar y deshabilitar *logins* de usuario, *host* o grupos de usuarios, según se conecten.

Se debe recordar que, si bien el par `telnet-telnetd` puede funcionar en modo *encrypt* en las últimas versiones (transferencia de datos encriptados, que

deben estar compilados con la opción correspondiente), es un comando que ha quedado en el olvido por su falta de seguridad (transmite el texto en claro sobre la red, lo que permite la visualización del contenido de la comunicación desde otra máquina, por ejemplo, con el comando `tcpdump`); no obstante, puede utilizarse en redes seguras o situaciones controladas.

Si no está instalado, se puede utilizar (Debian) `apt-get install telnetd` y después verificar que se ha dado de alta, bien en `/etc/inetd.conf`, bien en `/etc/xinetd.conf` (o en el directorio en que estén definidos los archivos; por ejemplo, `/etc/xinetd.d` según se indique en el archivo anterior con la sentencia `include /etc/xinetd.d`). El `xinetd.conf` o el archivo `/etc/xinetd.d/telnetd` deberán incluir una sección como*:

***Cualquier modificación en `xinetd.conf` deberá arrancar nuevamente el servicio con `service xinetd restart`.**

```
service telnet {
disable = no
flags = REUSE
socket_type = stream
wait = no
user = root
server = /usr/sbin/in.telnetd
log_on_failure += USERID
}
```

SSL telnet(d)

Se recomienda que en lugar de usar `telnetd` se utilice `SSL telnet(d)`, que reemplaza a `telnet(d)` utilizando encriptación y autentificación por SSL, o que se utilice SSH (el cual ya es un estándar en todas las distribuciones y entornos). El `SSLTelnet(d)` puede funcionar con el `telnet(d)` normal en ambas direcciones ya que al inicio de la comunicación verifica si el otro lado (*peer*) soporta SSL, de lo contrario, continúa con el protocolo telnet normal. Las ventajas con respecto al `telnet(d)` son que sus contraseñas y datos no circularán por la red en modo de texto plano y nadie que utilice el comando antes mencionado (`tcpdump`) podrá ver el contenido de la comunicación. También `SSL telnet` se puede utilizar para conectarse, por ejemplo, a un servidor web seguro (por ejemplo `https://servidor.web.org`) simplemente haciendo `telnet servidor.web.org 443`.

1.3.2. SSH, *Secure shell*

Un cambio aconsejable hoy en día es utilizar `ssh` (comando que ya está en todos los sistemas operativos incluido en Windows con `putty`* o `moabterm`**) en lugar de `telnet`, `rlogin` o `rsh`. Estos tres últimos comandos son inseguros (excepto `SSLTelnet`) por varias razones: la más importante es que todo lo que se transmite por la red, incluidos nombres de usuarios y contraseñas, es en texto plano (aunque existen versiones de telnet-telnetd encriptados, debe coincidir que ambos los sean), de modo que cualquiera que tenga acceso a esa red o a algún segmento de la misma puede obtener toda esta información y luego suplantar la identidad del usuario. La segunda es que estos puertos (telnet, rsh, etc.) son el primer lugar donde un *cracker* intentará conectarse. El protocolo `ssh` (en su versión OpenSSH) provee una conexión encriptada y comprimida mucho más segura que, por ejemplo, `telnet` (es recomendable utilizar la versión 2.0 o versiones superiores del protocolo). Todas las distribu-

***http://www.putty.org**
****http://mobaxterm.mobatek.net**

ciones actuales incorporan el cliente `ssh` y el servidor `sshd` por defecto. Es necesario tener actualizada la librería OpenSSL, utilizada por estos programas, ya que recientemente se encontró un problema de seguridad llamado *heartbleed* y que ha sido rápidamente solucionado en la mayoría de las distribuciones GNU/Linux (http://www.kriptopolis.com/recomendaciones-heartbleed).

ssh

Para ejecutar el comando, haced:

```
ssh -l login name host o ssh user@hostname
```

A través de SSH se pueden encapsular otras conexiones como X11 o cualquier otra TCP/IP. Si se omite el parámetro `-l`, el usuario se conectará con el mismo usuario local y en ambos casos el servidor solicitará la contraseña para validar la identidad del usuario. SSH soporta diferentes modos de autentificación (ved `man ssh`) basados en algoritmo RSA y clave pública. Si el cliente no está instalado, hacer `apt-get install openssh-client`.

Usando el comando `ssh-keygen -t rsa|dsa`, se pueden crear las claves de identificación de usuario. El comando crea en el directorio del `.shh` del usuario los ficheros* `id_rsa` y `id_rsa.pub`, las claves privada y pública, respectivamente. El usuario podría copiar la pública (`id_rsa.pub`) en el archivo `$HOME/.ssh/authorized_keys` del directorio `.ssh` del usuario de la máquina remota. Este archivo podrá contener tantas claves públicas como sitios desde donde se quiera conectar a esta máquina de forma remota. La sintaxis es de una clave por línea aunque las líneas tendrán un tamaño considerable. Después de haber introducido las claves públicas del usuario-máquina en este archivo, este usuario se podrá conectar sin contraseña desde esa máquina. También para realizar esta copia se puede utilizar el comando `ssh-copy-id máquina` que copiará las llaves en forma automática y es mucho más simple de utilizar.

***Por ejemplo, para el algoritmo de encriptación RSA.**

En forma normal (si no se han creado las claves), se le preguntará al usuario una contraseña, pero como la comunicación será siempre encriptada, nunca será accesible a otros usuarios que puedan escuchar sobre la red. Para mayor información, consultad `man ssh`. Para ejecutar remotamente un comando, simplemente haced:

```
ssh -l login name host_comando_remoto
Por ejemplo: ssh -l user localhost ls -al
```

sshd

El `sshd` es el servidor (*daemon*) para el `ssh` (si no están instalados, se puede hacer con `apt-get install openssh-client openssh-server`. Juntos reemplazan a `rlogin`, `telnet`, `rsh` y proveen una comunicación segura y encriptada entre dos *hosts* inseguros de la red. Este se arranca generalmente a través de los archivos de inicialización (`/etc/init.d` o `/etc/rc`) y espera conexiones de los clientes. El `sshd` de la mayoría de las distribuciones actuales soporta las versiones 1, 2 y 3 del protocolo SSH. Cuando se instala el paquete, se crea una clave RSA específica del *host* y cuando el *daemon* se inicia, crea otra, la RSA para la sesión, que no se almacena en el disco y que cambia cada hora. Cuando un cliente inicia la comunicación, genera un número aleatorio de 256 bits que está encriptado con las dos claves del servidor y enviado. Este número se utilizará durante la comunicación como clave de sesión para encriptar la comunicación que se realizará a través de un algoritmo de encriptación estándar. El usuario puede seleccionar cualquiera de los algoritmos disponibles ofrecidos por el servidor. Existen algunas diferencias (más seguro) cuando se utiliza la versión 2 (o 3) del protocolo. A partir de este momento, se inician algunos de los métodos de autentificación de usuario descritos en el cliente o se le solicita la contraseña, pero siempre con la comunicación encriptada. Para mayor información, consultad `man sshd`.

Tanto ssh como sshd disponen de un gran conjunto de parámetros que pueden ser configurados según se necesite en los archivos `/etc/ssh/ssh_config` y `/etc/ssh/sshd_config`. En el servidor probablemente algunas de las opciones más utilizadas son `PermitRootLogin yes|no` para permitir la conexión del usuario root o no, `IgnoreRhosts yes` para evitar leer los archivos `$HOME/.rhosts` y `$HOME/.shosts` de los usuarios, `X11Forwarding yes` para permitir que aplicaciones Xwindows en el servidor se visualicen en la pantalla del cliente (muy útil en la administración remota de servidores con el comando `ssh -X host_a_administrar`).

Túnel sobre SSH

Muchas veces tenemos acceso a un servidor `sshd` pero por cuestiones de seguridad no podemos acceder a otros servicios que no están encriptados (por ejemplo, un servicio de consulta de mail POP3 o un servidor de ventanas X11) o simplemente se quiere conectar a un servicio al cual solo se tiene acceso desde el entorno de la empresa. Para ello, es posible establecer un túnel encriptado entre la máquina cliente (por ejemplo, con Windows y un cliente ssh llamado `putty` de software libre) y el servidor con `sshd`. En este caso, al vincular el túnel con el servicio, el servicio verá la petición como si viniera de la misma máquina. Por ejemplo, si queremos establecer una conexión para POP3 sobre el puerto 110 de la máquina remota (y que también tiene un servidor `sshd`) haremos:

```
ssh -C -L 1100:localhost:110 usuario-id@host
```

Este comando pedirá la contraseña para el usuario-id sobre *host* y una vez conectado, se habrá creado el túnel. Cada paquete que se envíe desde la máquina local sobre el puerto 1100 se enviará a la máquina remota *localhost* sobre el puerto 110, que es donde escucha el servicio POP3 (la opción `-C` comprime el tráfico por el túnel).

Hacer túneles sobre otros puertos es muy fácil. Por ejemplo, supongamos que *solo* tenemos acceso a un *remote proxy server* desde una máquina remota (*remote login*), no desde la máquina local; en este caso, se puede hacer un túnel para conectar el navegador a la máquina local. Consideremos que tenemos *login* sobre una máquina pasarela (*gateway*), que puede acceder a la máquina llamada *proxy*, la cual ejecuta el *Squid Proxy Server* sobre el puerto 3128. Ejecutamos:

```
ssh -C -L 8080:proxy:3128 user@gateway
```

Después de conectarnos, tendremos un túnel escuchando sobre el puerto local 8080 que reconducirá el tráfico desde *gateway* hacia *proxy* al 3128. Para navegar de forma segura solo se deberá hacer `http://localhost:8080/`.

1.3.3. VPN SSL (via tun driver)

OpenSSH v4.3 introduce una nueva característica que permite crear *on-the-fly* VPN vía el túnel driver (tun) como un puente entre dos interfaces en diferentes segmentos de red a través de Internet y por lo cual un ordenador (B en nuestro caso) "quedará dentro" de la red del otro ordenador (A) a pesar de estar dentro de otra red. Para analizar este mecanismo, consideraremos que el ordenador A y B están conectados a la red vía ethernet y a su vez a Internet a través de una *gateway* que hace NAT. A tiene IP sobre su red (privada) 10.0.0.100, y B sobre su red (privada) 192.168.0.100 y como dijimos cada uno tiene acceso a internet NAT *gateway*.

A través de OpenSSH conectaremos B a la red de A vía una interfaz túnel ssh. Como hemos definido A ya tiene IP sobre la red A (10.0.0.100) y que se ve desde el gateway externo como dirección 1.2.3.4, es decir esta es la IP pública del servidor `ssh` de A (el router de A deberá hacer un forwarding de 1.2.3.4:22 a 10.0.0.100:22 para que externamente se pueda contactar con servidor `ssh` de A). B deberá tener una IP sobre la red A que será su interfaz tun0 y que le asignaremos 10.0.0.200. B debe también debe tener acceso al servidor `ssh` de A (o bien acceso directo o el puerto 22 debe ser forwarded sobre la red A en la dirección 1.2.3.4). Cuando el túnel esté creado B accederá directamente a la red de A lo cual significa que B "estará" dentro de la red de A).

Los pasos de configuración son los siguientes:

1) Activar el IP forwarding: `echo 1 >/proc/sys/net/ipv4/ip_forward`

2) Túnel: Sobre B `ssh -w 0:0 1.2.3.4` se puede utilizar como opciones además `-NTCf`. Esto crea un tunnel interface tun0 entre cliente (B) y el servidor (A) , recordar que A tiene dirección pública (1.2.3.4). Se deberá tener acceso de root a ambos sistemas para que puedan crear las interfaces (en A verificar que se tiene en `sshd_config PermitRootLogin yes` y `PermitTunnel yes`). Se recomienda utilizar SSH *keys* (PKI) y por lo cual se deberá cambiar PermitRootLogin without-password y si sobre el cliente no se quiere dar acceso de root se puede utilizar el comando `sudo` (ver `man sudoers`).

3) Verificar las interfaces: `ip addr show tun0`

4) Configurar las interfaces:
Ordenador A: `ip link set tun0 up ip addr add 10.0.0.100/32 peer 10.0.0.200 dev tun0`
Ordenador B: `ip link set tun0 up ip addr add 10.0.0.200/32 peer 10.0.0.100 dev tun0`

5) Probar: sobre B `ping 10.0.0.100` y sobre A `ping 10.0.0.200`

6) Routing directo: tenemos un link que conecta B en la red A pero es necesario inicializar el routing.
Sobre B: `ip route add 10.0.0.0/24 via 10.0.0.200` (para permitir enviar desde B a cualquier máquina de la red A).

7) Routing inverso: También para que los paquetes vuelvan a B sobre A debemos hacer `arp -sD 10.0.0.200 eth0`

El routing nos asegurará que otras máquinas que están en la red A puedan enviar paquetes a B a través de A. Podemos probar haciendo desde B `ping 10.0.0.123`. Si se desea que todos los paquetes de B vayan a través de A deberemos cambiar el default *gateway* de B pero esto no es simple ya que el propio túnel va por internet. Primero debemos crear un *host-based route* hacia la máquina A (todos los paquetes excepto los que crean el *link* deben ir por el túnel -pero obviamente no los que crean el túnel-). Sobre B: `ip route add 1.2.3.4/32 via 192.168.0.1` En este caso 192.168.0.1 es el *default gateway* para B, es decir el *gateway* sobre la red B que provee conectividad a internet y que nos servirá para mantener el túnel. Luego se podemos cambiar el *default gateway* sobre B: `ip route replace default via 10.0.0.1`. Con lo cual tendremos que 192.168.0.1 es el *default gateway* de la red B y 10.0.0.1 *default gateway* de la red A. Ya que la máquina B está conectada a la red de A le estamos indicando utilizar la red A como *default gateway* en lugar de *default gateway* sobre la red B (como sería habitual). Se puede verificar esto con el comando `tracepath google.com`.

1.3.4. Túneles encadenados

Muchas veces el acceso a redes internas solo es posible a través de un servidor SSH (que están en la DMZ) y desde esta es posible la conexión hacia servidores

SSH internos para luego llegar a la máquina SSH de nuestro interés (y si queremos copiar archivos o hacer el *forwarding* de X11 puede ser todo un reto). La forma de hacerlo es primero conectarnos a la M1, luego de esta a la M2 (a la cual solo es posible conectarse desde M1) y desde esta a la M3 (a la cual solo es posible conectarse desde M2). Una forma de facilitar la autentificación es utilizar el *agent forwarding* con el parámetro -A y poniendo nuestra llave pública en todos los *$HOME/.ssh/authorized_keys* así cuando vayamos ejecutando los diferentes comandos la petición de llaves puede ser *forwarded* hacia la máquina anterior y así sucesivamente. los comandos serán `ssh -A m1.org` y desde esta `ssh -a m2.org` y a su vez `ssh -a m3.org`. Para mejorar esto podemos automatizarlo con `ssh -A -t m1.org ssh -A -t m2.org ssh -A m2.org` (se ha incluido la opción -t para que `ssh` le asigne una pseudo-tty y solo veremos la pantalla final). El applet SSHmenu puede ayudar a automatizar todo esto (http://sshmenu.sourceforge.net/).

Una forma de gestionar efectivamente esta conexión "multisaltos" es a través de la opción `ProxyCommand` de `ssh` (ver `man ssh_config`) que nos permitirá conectarnos de forma más eficiente. Para ello definiremos los siguientes comandos en *$HOME/.ssh/config*:

```
Host m1
    HostName m1.org
Host m2
    ProxyCommand ssh -q m1 nc -q0 m2.org 22
Host m3
    ProxyCommand ssh -q m2 nc -q0 m3.org 22
```

Donde el primer comando (cuando hagamos `ssh m1`) nos conectará a m1.org. Cuando ejecutemos `ssh m2` se establecerá una conexión ssh sobre m1 pero el comando ProxyCommand utilizará en comando nc para extender la conexión sobre m2.org. Y el tercero es una ampliación del anterior por lo cual cuando ejecutemos `ssh m3` es como si ejecutáramos una conexión a m1 y luego ampliáramos esta a m2 y luego a m3.[23]

1.4. Servicios de transferencia de ficheros: FTP

El FTP (*File Transfer Protocol*) es un protocolo cliente/servidor (bajo TCP) que permite la transferencia de archivos desde y hacia un sistema remoto. Un servidor FTP es un ordenador que ejecuta el *daemon* `ftpd`.

Algunos sitios que permiten la conexión anónima bajo el usuario *anonymous* son generalmente repositorios de programas. En un sitio privado, se necesitará un usuario y una contraseña para acceder. También es posible acceder a un servidor ftp mediante un navegador y generalmente hoy en día los repositorios de programas se sustituyen por servidores web (p. ej. Apache) u otras tecnologías como Bittorrent (que utiliza redes *peer to peer*, P2P) o servidores web con módulos de WebDav o el propio comando `scp` (*secure copy*) que forma parte

del paquete openssh-client o el par `sftp/sftpd` que forman parte de los paquetes `openssh-client` y `openssh-server` respectivamente. No obstante, se continúa utilizando en algunos casos y Debian, por ejemplo, permite el acceso con usuario/contraseña o la posibilidad de subir archivos al servidor (si bien con servicios web-dav también es posible hacerlo).

El protocolo (y los servidores/clientes que lo implementan) de ftp por definición no es encriptado (los datos, usuarios y contraseñas se transmiten en texto claro por la red) con el riesgo que ello supone. Sin embargo, hay una serie de servidores/clientes que soportan SSL y por lo tanto encriptación.

1.4.1. Cliente ftp (convencional)

Un cliente ftp permite acceder a servidores ftp y hay una gran cantidad de clientes disponibles. El uso del ftp es sumamente simple; desde la línea de comando, ejecutad:

```
ftp nombre-servidor
```

O también `ftp` y luego, de forma interactiva, `open nombre-servidor`

El servidor solicitará un nombre de usuario y una contraseña (si acepta usuarios anónimos, se introducirá *anonymous* como usuario y nuestra dirección de correo electrónico como contraseña) y a partir del *prompt* del comando (después de algunos mensajes), podremos comenzar a transferir ficheros.

El protocolo permite la transferencia en modo ASCII o binario. Es importante decidir el tipo de fichero que hay que transferir porque una transferencia de un binario en modo ASCII inutilizará el fichero. Para cambiar de un modo a otro, se deben ejecutar los comandos `ascii` o `binary`. Los comandos útiles del cliente ftp son el `ls` (navegación en el directorio remoto), `get nombre_del_fichero` (para descargar ficheros) o `mget` (que admite `*`), `put nombre_del_fichero` (para enviar ficheros al servidor) o `mput` (que admite `*`); en estos dos últimos se debe tener permiso de escritura sobre el directorio del servidor. Se pueden ejecutar comandos locales si antes del comando se inserta un "`!`". Por ejemplo, `!cd /tmp` significará que los archivos que bajen a la máquina local se descargarán en `/tmp`. Para poder ver el estado y el funcionamiento de la transferencia, el cliente puede imprimir marcas, o *ticks*, que se activan con los comandos `hash` y `tick`. Existen otros comandos que se pueden consultar en la hoja del manual (`man ftp`) o haciendo `help` dentro del cliente. Contamos con numerosas alternativas para los clientes, por ejemplo en modo texto: `ncftp`, `lukemftp`, `lftp`, `cftp`, `yafc Yafc` o, en modo gráfico: `gFTP`, `WXftp`, `LLNL XFTP`, `guiftp`.

Enlace de interés

En la Wikipedia disponéis de una comparativa de diversos clientes FTP: http://en.wikipedia.org/wiki/Comparison_of_FTP_client_software

1.4.2. Servidores FTP

El servidor tradicional de UNIX se ejecuta a través del puerto 21 y es puesto en marcha por el *daemon* `inetd` (o `xinetd`, según se tenga instalado). En `inetd.conf` conviene incluir el *wrapper* `tcpd` con las reglas de acceso en `host.allow` y el `host.deny` en la llamada al `ftpd` por el `inetd` para incrementar la seguridad del sistema. Cuando recibe una conexión, verifica el usuario y la contraseña y lo deja entrar si la autentificación es correcta. Un FTP *anonymous* trabaja de forma diferente, ya que el usuario solo podrá acceder a un directorio definido en el archivo de configuración y al árbol subyacente, pero no hacia arriba, por motivos de seguridad. Este directorio generalmente contiene directorios `pub/, bin/, etc/` y `lib/` para que el *daemon* de ftp pueda ejecutar comandos externos para peticiones de `ls`. El *daemon* `ftpd` soporta los siguientes archivos para su configuración:

Ved también

El tema de la seguridad del sistema se estudia en el módulo "Administración de seguridad".

- **/etc/ftpusers**: lista de usuarios que no son aceptados en el sistema. Un usuario por línea.
- **/etc/ftpchroot**: lista de usuarios a los que se les cambiará el directorio base `chroot` cuando se conecten. Necesario cuando deseamos configurar un servidor anónimo.
- **/etc/ftpwelcome**: anuncio de bienvenida.
- **/etc/motd**: noticias después del *login*.
- **/etc/nologin**: mensaje que se muestra después de negar la conexión.
- **/var/log/ftpd**: *log* de las transferencias.

Si en algún momento queremos inhibir la conexión al ftp, se puede incluir el archivo `/etc/nologin`. El `ftpd` muestra su contenido y termina. Si existe un archivo `.message` en un directorio, el `ftpd` lo mostrará cuando se acceda al mismo. La conexión de un usuario pasa por cinco niveles diferentes:

1) tener una contraseña válida;

2) no aparecer en la lista de `/etc/ftpusers`;

3) tener un *shell* estándar válido;

4) si aparece en `/etc/ftpchroot`, se le cambiará al directorio `home` (incluido si es *anonymous* o ftp);

5) si el usuario es *anonymous* o ftp, entonces deberá tener una entrada en el `/etc/passwd` con usuario ftp, pero podrá conectarse especificando cualquier contraseña (por convención se utiliza la dirección de correo electrónico).

Es importante prestar atención a que los usuarios habilitados únicamente para utilizar el servicio ftp no dispongan de un *shell* a la entrada correspondiente de dicho usuario en `/etc/passwd` para impedir que este usuario tenga conexión, por ejemplo, por `ssh` o `telnet`. Para ello, cuando se cree el usuario, habrá que indicar, por ejemplo:

```
useradd -d/home/nteum -s /bin/false nteum
```
y luego: `passwd nteum`,

lo cual indicará que el usuario nteum no tendrá *shell* para una conexión interactiva (si el usuario ya existe, se puede editar el fichero `/etc/passwd` y cambiar el último campo por `/bin/false`). A continuación, se debe agregar como última línea `/bin/false` en `/ect/shells`. En [28] se describe paso a paso cómo crear tanto un servidor ftp seguro con usuarios registrados como un servidor ftp anonymous para usuarios no registrados. Dos de los servidores no estándares más comunes son el ProFTPD y Vsftpd (ambos incluidos en Debian por ejemplo).

Enlaces de interés

Para saber más sobre ProFTPD y Vsftpd podéis visitar las siguientes páginas http://www.proftpd.org https://security.appspot.com /vsftpd.html.

Para instalar el Proftpd sobre Debian, ejecutad: `apt-get install proftpd`. Una vez descargado `debconf`, preguntará si se quiere ejecutar por `inetd` o manualmente (es recomendable elegir la última opción). Si se debe detener el servicio (para cambiar la configuración, por ej.): `/etc/init.d/proftpd stop` y para modificar el fichero: `/etc/proftpd.conf`. Un servidor (Debian) muy interesante es el PureFtpd (`pure-ftpd`), que es muy seguro, permite usuarios virtuales, cuotas, SSL/TSL y unas características interesantes. El paquete virtual de Debian ftp-server (https://packages.debian.org/wheezy/ftp-server) muestra todos los paquetes de este tipo incluidos en la última versión.

Enlaces de interés

Para configurar un servidor FTP en modo encriptado (TSL) o para tener acceso *anonymous* podéis consultar: http://www.debian -administration.org /articles/228.
Por otro lado, para saber más sobre la instalación y configuración de PureFtpd podéis consultar: http://www.debian -administration.org /articles/383.

1.5. *Active Directory Domain Controller* con Samba4

Uno de los aspecto más importantes en la integración de sistemas, es la gestión de usuarios e identificación centralizada así como las autoridades de autentificación y los permisos. En mucho sitios basados en Windows esta tarea es realizada por un *Active Directory* (AD, servicio de directorio en una red distribuida de ordenadores -a veces también llamado PDC por *Primary Domain Controller*-) y es importante contar con herramientas de la banda de *Open Source* que permitan gestionar este tipo de entornos. Samba versión 4 es una nueva distribución de este popular software que permite la integración con sistemas Windows actuando como servidor de archivos y/o impresoras y además, en esta última versión y de forma estable, puede actuar como controlador de un dominio Windows aceptando clientes, gestionando usuarios y directorios en forma centralizada y totalmente compatible.[29, 30]

De los expertos de *Active Directory* (que generalmente son consultores especializados y con un alto coste) sabemos que es AD es una unión (que puede ser muy complicada de entender para los administradores que nos estén dedicados al mundo Windows) de diferentes servicios y tecnologías como DNS, Kerberos, LDAP y CIFS. Algunos incluyen DHCP también pero como veremos no es necesario en nuestra instalación. Estaríamos en un error si pensamos

que configurando cada uno de estos servicios tendríamos el resultado del conjunto, pero Samba4 presenta una capa de abstracción, estable y eficiente, que las integra para ofrecer un producto complejo pero que se puede configurar y administrar siguiendo un conjunto de pasos sin grandes dificultades (aunque no se debe pensar que es simple). El elemento crítico (base de los mayores problemas y errores) del AD es el DNS (en realidad Windows utilizará el AD como DNS) ya que este lo hará servir para 'agregar extraoficialmente' a la lista de nombres habituales en un DNS, los servidores AD para que los clientes puedan encontrarlos y tener acceso a ellos.

En relación Kerberos y LDAP los administradores de GNU/Linux saben de su potencialidad y complejidad, pero en el caso de AD están integrados en el paquete y si bien les otorga una cierta estandarización del sistema no son integrables con servidores u otros clientes, solo se utilizan para sus objetivos y particularmente cuando utilizamos Samba4, será este quien los configurará y gestionará en nuestro nombre con pequeñas modificaciones por parte del administrador. La versión actual de Samba (V4) no difiere de las anteriores en cuanto a compartición de archivos e impresoras (incluso se ha simplificado su gestión/administración) y además con la implementación de AD permitirá que aquellos administradores que utilicen Windows puedan continuar utilizando sus herramientas de gestión y administración de dominio solo con apuntar al servidor Samba. Toda la configuración de Samba pasará ahora por el archivo `smb.conf` (normalmente en */etc/samba/smb.conf*) con definiciones simplificadas pero permitiendo la gestión compleja de un dominio Windows a través de las herramientas (también complejas) de Windows como por ejemplo RSAT (*Remote Server Administration Tools*) y obviamente a través del sistema GNU/Linux y la CLI de Samba4 se tendrá acceso a toda la configuración y administración del AD (sin necesidad de utilizar Windows en ningún caso).[30, 33]

En nuestra instalación habitual utilizaremos Debian Wheezy como distribución y si bien Samba4 está en los repositorios *backport*, (en los de Wheezy solo existe la versión 3.x pero estará disponible en la nueva edición de Debian Jessie) optamos, en la facilidad a la hora de resolver las dependencias, por bajar el código fuente y compilarlo.

1.5.1. Pasos preliminares

Una de las primeras verificaciones que debemos hacer es que NPT esté instalado y funcionando correctamente (`apt-get install ntp` y probarlo que sincronza correctamente con `ntpq -p`) ya que Kerberos es muy exigente en cuanto a los valores del reloj. Un segundo aspecto a cuidar es definir los servidores con IP estáticas (al menos durante la prueba de concepto de la instalación y configuración del AD y óbviamente en un servidor en producción). Para ello definir */etc/network/interfaces* con:

```
auto eth0
iface eth0 inet static
    address 192.168.1.33
    netmask 255.255.255.0
    gateway 192.168.1.1
```

Luego deberemos verificar el */etc/hosts* (si es que no disponemos de un DNS que podamos modificar los registros A y PTR para incluir nuestro servidor de AD) para agregar (en nuestro caso) una entrada como: `192.168.1.33 sysdw.nteum.org sysdw`. Como DNS se podría utilizar cualquiera de los que se ha visto anteriormente, pero en este caso nos pareció más adecuado, para hacer una prueba total de Samba4, utilizar el que trae interno el paquete y que lo utilizará el AD. Es por ello que deberemos estar seguros de desactivar cualquier otro servicio de DNS que tengamos en esta máquina para que no existan conflictos (mirar por ejemplo con `netstat -anotup`). El otro aspecto importante es el nombre de dominio, y sin entrar en polémicas con relación a cómo se trata en GNU/Linux y cómo se 'ven' en Windows, hemos escogido por simplificar el del DNS (si bien algunos expertos indican que puede haber conflictos cuando tratemos con un DNS externo). Es decir un *AD Domain* no significa realmente un *DNS domain* sino que es una 'unión' híbrida de DNS y que Kerberos denomina `realm`. Es por ello que configuraremos el */etc/resolv.conf* apuntando como DNS a la propia máquina de Samba4 y con el dominio DNS de la misma (y como secundario uno público) -recordar que si se tiene instalado y activo el paquete `resolvconf` las modificaciones de */etc/resolv.conf* son dinámicas y las inclusión de los DNS se deberán hacer sobre */etc/network/interfaces* con los tags `dns-nameservers` y `dns-search`-:

```
domain nteum.org
search nteum.org
nameserver 192.168.1.33
nameserver 8.8.8.8
```

Además verificar que `/etc/hostname` coincide con el nombre indicado en `/etc/hosts`. Por último como Samba4 actuará como servidor de archivos y como PDC/AD, es necesario asegurar que el sistema de archivos que estemos exportando soporte ACL (*access control list*) y atributos extendidos, para ello modificaremos */etc/fstab* para indicarle al *file system* correspondiente los atributos `user_xattr,acl` incluyendo una línea como:

```
UUID=.... / ext4 user_xattr,acl,errors=remount-ro 0 2
```

Aquí lo hemos hecho sobre el directorio root ya que tenemos una partición solamente, pero esto solo se debe aplicar a aquellas particiones que se desee servir a los clientes.

Finalmente reiniciar la máquina para que todas las configuraciones se actualicen correctamente.

1.5.2. Compilar Samba4

En primer lugar debemos tener una serie de paquetes instalados (si bien muchos de ellos ya se encuentran instalados) -esta lista puede variar según las fuentes que se consulten- pero los requerimientos de Samba4 indican:

```
apt-get install build-essential libacl1-dev libattr1-dev \
   libblkid-dev libgnutls-dev libreadline-dev python-dev libpam0g-dev \
   python-dnspython gdb pkg-config libpopt-dev libldap2-dev \
   dnsutils libbsd-dev attr krb5-user docbook-xsl libcups2-dev acl
```

Otros expertos en Samba4 y AD también incluyen (algunas de ellas ya estarán incluidas o se resolverán por dependencias):

```
apt-get install  pkg-config libacl1 libblkid1 attr libattr1 libgnutls-dev  \
libncurses-dev libdm0-dev libfam0 fam libfam-dev xsltproc libnss3-dev \
docbook-xsl-doc-html docbook-xsl-ns python-dnspython libcups2-dev krb5-config
```

En nuestro caso usaremos como servidor de Kerberos nuestro servidor AD por lo cual es importante su configuración (paquetes `krb5-user` y `krb5-config`) y deberemos introducir durante su instalación el dominio (realm) que servirá en letras MAYÚSCULAS, que en nuestro caso será NTEUM.ORG y el *password* deberá tener cierta complejidad (como por ejemplo PaSSw0d2014) para que el programa lo considere válido. Para obtener y compilar la última versión de Samba4 haremos:

`wget http://www.samba.org/samba/ftp/stable/samba-4.x.y.tar.gz` (Reemplazar x.y con la última versión)

```
tar -xzf samba-4.x.y.tar.gz
cd samba-4.x.y
./configure --prefix=/opt/samba4
make
make install
```

Nosotros hemos escogido como lugar de instalación */opt/samba4* por lo cual deberemos actualizar la variable PATH con (poner en `.profile`):

```
export PATH=/usr/local/samba/bin:/usr/local/samba/sbin:$PATH
```

y para que la configuración quede como es habitual se crea un enlace a */etc/samba* con `ln -s /usr/local/samba/etc /etc/samba`.[32, 33]

Lo siguiente es aprovisionar el AD Domain:

```
samba-tool domain provision --use-rfc2307 --interactive
```

Aquí deberemos indicar el Realm (NTEUM.ORG que es el mismo que configuramos en krb5) y el Dominio (NTEUM en nuestro caso). Es importante que el *password* coincida con el que hemos introducido en krb5 (y tenga cierta complejidad). Asegurarse que el *DNS forwarder* apunte a el DNS server real que resolverá la peticiones de DNS e introducir la línea `allow dns updates = nonsecure` en los parámetros globales. El archivo */etc/samba/smb.conf* quedará como:

```
[global]
workgroup = NTEUM
realm = nteum.org
netbios name = SYSDW
server role = active directory domain controller
dns forwarder = 158.109.0.9
allow dns updates = nonsecure

[netlogon]
path = /opt/samba4/var/locks/sysvol/nteum.org/scripts
read only = No

[sysvol]
path = /opt/samba4/var/locks/sysvol
read only = No
```

A continuación debemos ajustar la configuración de Kerberos haciendo:

```
mv /etc/krb5.conf /etc/krb5.conf.org
cp  /opt/samba4/private/krb5.conf .
```

Cuyo contenido deberá quedar como:

```
[libdefaults]
default_realm = NTEUM.ORG
dns_lookup_realm = false
dns_lookup_kdc = true
```

Una vez reiniciado el sistema podemos poner en marcha el servidor simplemente ejecutando `samba` y mirar el */var/log/syslog* para verificar que se ha arrancado correctamente y si hay errores de que tipo son para solucionarlos (si ejecutamos `ps -edaf | grep samba` veremos aproximadamente 14 procesos samba si todo ha ido bien). Podremos verificar la ejecución del servidor haciendo:

```
smbclient -L localhost -U%
    Domain=[NTEUM] OS=[Unix] Server=[Samba 4.1.9]
       Sharename       Type      Comment
   ---------       ----      -------
   netlogon        Disk
   sysvol          Disk
   IPC$            IPC       IPC Service
    Domain=[NTEUM] OS=[Unix] Server=[Samba 4.1.9]
   Server               Comment
   ---------            -------
   Workgroup            Master
       ---------            -------

smbclient //localhost/netlogon -UAdministrator -c 'ls'      Nos solicitará el passwd de administrator
    Enter Administrator's password:
    Domain=[NTEUM] OS=[Unix] Server=[Samba 4.1.9]
    .                                   D        0  Wed Jul 23 10:11:33 2014
    ..                                  D        0  Wed Jul 23 10:11:39 2014

44541 blocks of size 262144. 7486 blocks available
```

Para verificar el DNS:

```
host -t SRV _ldap._tcp.nteum.org
    _ldap._tcp.nteum.org has SRV record 0 100 389 sysdw.nteum.org.

host -t SRV _kerberos._udp.nteum.org.
    _kerberos._udp.nteum.org has SRV record 0 100 88 sysdw.nteum.org.
```

```
host -t A sysdw.nteum.org
sysdw.nteum.org has address 158.109.65.67
```

También verificar que tenemos conexión externa con un `ping google.com` por ejemplo. Por último para verificar el funcionamiento de KRB5 (el realm debe ir en Mayúsculas):

```
kinit administrator@NTEUM.ORG
    Password for administrator@NTEUM.ORG:
    Warning: Your password will expire in 41 days on Wed Sep  3 17:10:52 2014
```

Y podríamos quitar (si bien en entornos de producción no sería recomendable) la caducidad del passwd con:

```
samba-tool user setexpiry administrator -noexpiry
```

Finalmente si todo funciona deberemos repetir el procedimiento de recrear el dominio pero agregando `-use-ntvfs` como parámetro, para ello podemos hacer:

```
mv /etc/samba/smb.conf /etc/samba/smb.conf.org
samba-tool domain provision --realm nteum.org --domain NTEUM --adminpass PaSSw0d2014 \
   --server-role=dc --use-ntvfs
```

El nuevo archivo smb.conf quedará (agregando nuevamente la línea `allow dns updates`):

```
[global]
workgroup = NTEUM
realm = nteum.org
netbios name = SYSDW
server role = active directory domain controller
dns forwarder = 158.109.0.9
server services = rpc, nbt, wrepl, ldap, cldap, kdc, drepl, winbind, ntp
_signd, kcc, dnsupdate, dns, smb
dcerpc endpoint servers = epmapper, wkssvc, rpcecho, samr, netlogon, lsa
rpc, spoolss, drsuapi, dssetup, unixinfo, browser, eventlog6, backupkey, dnsserv
er, winreg, srvsvc
allow dns updates = nonsecure
[netlogon]
path = /opt/samba4/var/locks/sysvol/nteum.org/scripts
read only = No

[sysvol]
path = /opt/samba4/var/locks/sysvol
read only = No
```

Y que /etc/krb5.conf es:

```
[libdefaults]
default_realm = NTEUM.ORG
dns_lookup_realm = false
dns_lookup_kdc = true
```

Reiniciar nuevamente y ejecutar `samba` (recordar que todavía no lo tenemos como servicio y que lo debemos ejecutar manualmente. Si queremos que Samba4 sirva los directorios *homes* deberemos agregar en *smb.conf* una sección con:

```
[home]
        path = /home/
        read only = no
```

Para que el administrador tenga los privilegios correctos en la gestión de las cuentas desde entornos Windows (y que Samba4 cree los directorios *home* automáticamente) deberemos ejecutar:

```
net rpc rights grant `NTEUM\Domain Admins` SeDiskOperatorPrivilege
-Uadministrator
```

Ahora deberíamos probar que podemos configurar un cliente Windows y agregarlo al directorio. Para ello (en nuestro caso) utilizaremos una máquina Windows8 donde en primer lugar modificaremos la interface de red para que el DNS apunte a nuestro servidor de AD (vamos a *Control Panel* > *Network and Internet* > *Network and Sharing Center* seleccionamos el dispositivo de red activo y en propiedades le cambiamos el DNS, dejando solo la IP de nuestro AD). Luego vamos a *Control Panel* > *System and Security* > *System* y seleccionamos *Advanced System Settings* y dentro *Computer Name* > *Change* e introducimos administrator y PaSSw0d2014 para agregar la máquina al dominio (en caso que no nos seleccione el dominio por defecto podemos poner en Username: NTEUM\administrator y luego el *passwd*), que nos dará un mensaje de confirmación si todo ha ido bien y nos pedirá hacer un *reboot* de la máquina. Cuando se inicie nuevamente, podremos seleccionar otro usuario/dominio y conectarnos a NTEUM\administrator (que es la cuenta previamente definida). Con todo ello se generarán una serie de pasos en el cliente y el servidor para aprovisionar en esa nueva máquina la configuración y directorio raíz.[32, 33]

Para administrar el sitio AD se pueden utilizar las herramientas de Windows (RSAT), las cuales se pueden obtener (sin cargo) desde el sitio del fabricante y con la guía indicada en [31] y ejemplos de configuración en [32]. También es posible utilizar la herramienta Swat (https://wiki.samba.org/index.php/SWAT2, última versión Noviembre de 2012) pero su instalación puede presentar algunos inconvenientes (sobre todo si Samba4 se ha instalado desde los fuentes). Por último en https://wiki.samba.org/index.php/Samba4/InitScript podremos encontrar el *script* para iniciar y apagar el servidor AD en forma automática.

1.6. Servicios de intercambio de información a nivel de usuario

1.6.1. El *Mail Transport Agent* (MTA)

Un MTA (*Mail Transport Agent*) se encarga de enviar y recibir los correos desde un servidor de correo electrónico hacia y desde Internet, que implementa el protocolo SMTP (*Simple Mail Transfer Protocol*). Todas las distribuciones incorporan diferentes MTA y por ejemplo las de Debian se pueden consultar en su paquete virtual *mail-transport-agent**. Una de las que se utilizan habitualmente es `exim`, ya que es más fácil de configurar que otros paquetes MTA, como son `postfix` o `sendmail` (este último es uno de los precursores). `Exim` presenta características avanzadas tales como rechazar conexiones de sitios de *spam* conocidos, posee defensas contra correo basura (*junk mails*) o bombardeo de

***https://packages.debian.org /wheezy/mail-transport-agent**

correo (*mail bombing*) y es extremadamente eficiente en el procesamiento de grandes cantidades de correos.

Su instalación es a través de `apt-get install exim4-daemon-heavy` (en este caso se optará por la instalación de la versión *heavy* que es la más completa y soporta lista de acceso (ACL) y características avanzadas, en instalaciones más simple se puede optar por *exim4-daemon-light*). Su configuración se realiza a través de `dpkg-reconfigure exim4-config` donde una respuesta típica a las preguntas realizadas es:

- *General type of mail configuration*: internet site; mail es enviado y recibido utilizado SMTP.
- *System mail name*: remix.world (nuestro dominio)
- *IP-addresses to listen on for incomming SMTP connections*: (dejar en blanco)
- *Other destinations for which mail is accepted*: remix.world
- *Domains to relay mail for*: (dejar en blanco)
- *Machines to relay mail for*: (dejar en blanco)
- *Keep number of DNS-queries minimal (Dial-on-Demand)*?: No
- *Delivery method for local mail: Maildir format in home directory*
- *Split configuration into small files*?: No

El servidor ya estará configurado y puede probarse con `echo test message | mail -s "test" adminp@SySDW.nteum.org` (por supuesto cambiando la dirección) y verificando que el mail ha llegado al usuario adminp (los errores se pueden encontrar en */var/log/exim4/mainlog*). La configuración será almacenada en */etc/exim4/update-exim4.conf.conf*. Para configurar autentificación por TLS, ACL y Spam Scaning consultar https://wiki.debian.org/Exim.

1.6.2. External SMTP

Cuando instalamos un nuevo sistema como servidores o estaciones de trabajo un aspecto relevante es el servidor de correo y podemos instalar grandes paquetes como los ya mencionados Postfix, Exim o Zimbra (en su version Community http://www.zimbra.com/) haciendo que los correos hacia dominios externos utilicen los servicios externos de SMTP (por ejemplo los de Google). Para máquinas virtuales, estaciones de trabajo o portátiles es un poco más complicado ya que generalmente tienen IP privadas o en redes internas por lo cual es necesario tener un servidor que haga de receptor de los correos externo a mi dominio, es decir un servidor que haga las funciones de *smarthost* por ejemplo el Google Apps SMTP. Para detalles de su configuración se puede seguir la documentación de https://wiki.debian.org/GmailAndExim4. De acuerdo a la información de Google* y para cuentas gratuitas el número máximo de destinatarios permitido por dominio y día es de 100 mensajes y que Gmail reescribirá la dirección del remitente. Para su configuración ejecutaremos `dpkg-reconfigure exim4-config` y seleccionaremos:

***https://support.google.com/a/answer/2956491?hl=es**

- *mail sent by smarthost; received via SMTP or fetchmail.*
- *System mail name*: localhost
- *IP-addresses to listen on for incomming SMTP connections*: 127.0.0.1
- *Other destinations for which mail is accepted*: (dejar en blanco)
- *Machines to relay mail for*: (dejar en blanco)
- *IP address or host name of the outgoing smarthost*: smtp.gmail.com::587
- *Hide local mail name in outgoing mail*?: No
- *Keep number of DNS-queries minimal (Dial-on-Demand)*?: No
- *Delivery method for local mail: mbox format in /var/mail*
- *Split configuration into small files*?: Yes

Esta es la configuración más adecuada si no se tiene un IP visible externamente. El envío sobre el puerto 587 de Gmail utiliza STARTTLS para asegurar la protección del passwd y para indicar el usuario y passwd de acceso a Gmail (utilizar una cuenta solo para este objetivo, no la cuenta habitual de Gmail) se debe editar el fichero */etc/exim4/passwd.client* y agregar la siguiente línea.

```
*.google.com:SMTPAccountName@gmail.com:y0uRpaSsw0RD
```

Luego ejecutar (para evitar que otros usuarios de la máquina puedan leer su contenido:

```
chown root:Debian-exim /etc/exim4/passwd.client
chmod 640 /etc/exim4/passwd.client
```

Gmail reescribirá la dirección del remitente automáticamente pero si no lo hiciera o enviamos a un *smarthost* que no lo hace deberíamos configurar el archivo */etc/email-addresses* con todas las combinaciones de direcciones posibles a utilizar (una por línea) y las dirección que se reescribirá (por ejemplo, nteum@remix.world: SMTPAccountName@gmail.com). Luego se deberá ejecutar:

```
update-exim4.conf
invoke-rc.d exim4 restart
exim4 -qff
```

Con ello se actualiza y recarga la configuración y se fuerza a enviar todos los correos que están pendientes. Como mostramos anteriormente en */var/log/exim4/mainlog* tendremos los errores si los hay.Si existen errores de autentificación sobre gmail verifique con el comando `host smtp.gmail.com` cuales son los *hosts* que devuelve y si estos concuerdan con el patrón incluido en */etc/exim4/passwd.client*. Si es diferente cámbielo para que coincida.

1.7. *Internet Message Access Protocol* (IMAP)

Este servicio permite acceder a los correos alojados en un servidor a través de un cliente de correo como por ejemplo Thunderbird del proyecto Mozilla.org.

Este servicio soportado por el *daemon* `imapd` (los actuales soportan el protocolo IMAP4rev1) permite acceder a un archivo de correo electrónico (*mail file*) que se encuentra en una máquina remota. El servicio `imapd` se presta a través de los puertos 143 (imap2), 220 (imap3) o 993 (imaps) cuando soporta encriptación por SSL. Si se utiliza `inetd`, este servidor se pone en marcha a través de una línea en `/etc/inetd.conf` como:

```
imap2 stream tcp nowait root /usr/sbin/tcpd /usr/sbin/imapd
imap3 stream tcp nowait root /usr/sbin/tcpd /usr/sbin/imapd
```

En este ejemplo se llama al *wrapper* `tcpd` que funciona con `hosts.allow` y `hosts.deny` para incrementar la seguridad. Las aplicaciones más populares son `courier-imap`, `cyrus-imapd`, `dovecot-imapd` entre otros. Para probar que el servidor `imap` funciona, se podría utilizar un cliente, por ejemplo Thunderbird/Icedove (Debian), Evolution, Squirremail, o cualquier otro cliente que soporte IMAP, crear una cuenta para un usuario local, configurarlo adecuadamente para que se conecte sobre la máquina local y verificar el funcionamiento de `imap`.

Se puede instalar `apt-get install dovecot-imapd` como prueba de concepto, que con las opciones por defecto permite una conexión encriptada por SSL y sobre buzones mailbox (o si queremos sobre buzones maildir deberemos cambiar la configuración de `/etc/dovecot/conf.d/10-mail.conf`). Dovecot es un servidor muy potente por lo cual permite una gran cantidad de opciones y configuraciones (consultar http://wiki2.dovecot.org/ y un resumen aplicado de ellas en Debian Wheezy*). Las pruebas se pueden completar configurando Evolution para que se conecte a nuestro servidor/usuario y leer los correos del servidor previamente configurado. Es importante notar que algunos clientes de correo/servidores de Imap solo soportan el formato MailBox y no Maildir y es por ello que se debe tener en cuenta cuando se utilicen los clientes/servidores de Imap. En las actividades que hemos realizado hasta ahora tanto `exim4` como `dovecot-imapd` soportan ambos formatos y se deben configurar durante la instalación.

***https://workaround.org/ispmail/wheezy/setting-up-dovecot**

1.7.1. Aspectos complementarios

Supongamos que como usuarios tenemos cuatro cuentas de correo en servidores diferentes y queremos que todos los mensajes que llegan a estas cuentas se recojan en una única, a la que podamos acceder externamente, y que haya también un filtro de correo basura (*antispam*). Primero se deben instalar `exim4 + imapd` y comprobar que funcionan.

Para recoger los mensajes de diferentes cuentas se utilizará `Fetchmail`, (que se instala con `apt-get install fetchmail`). A continuación, se debe crear el fichero `.fetchmailrc` en nuestro `$HOME` (también se puede utilizar la herramienta `fetchmailconf`) que deberá tener ser algo así como:

```
set postmaster “adminp”
set bouncemail
set no spambounce
set flush
poll pop.domain.com proto pop3
   user 'nteum' there with password 'MyPaSSwOrD' is 'nteum' here
poll mail.domain2.com
   user 'adminp' there with password 'MyPaSSwOrD' is 'adminp' here
   user 'nteum' there with password 'MyPaSSwOrD' is 'nteum' here
```

La acción `set` indica a `Fetchmail` que esta línea contiene una opción global (envío de errores, eliminación de los mensajes de los servidores, etc.). A continuación, se especifican dos servidores de correo: uno para que compruebe si hay correo con el protocolo POP3 y otro para que pruebe a usar varios protocolos con el fin de encontrar uno que funcione. Se comprueba el correo de dos usuarios con la segunda opción de servidor, pero todo el correo que se encuentre se envía al *spool* de correo de `adminp`. Esto permite comprobar varios buzones de diversos servidores como si se tratara de un único buzón. La información específica de cada usuario comienza con la acción `user`. El `Fetchmail` se puede poner en el `cron`* para que se ejecute automáticamente o ejecutarlo en modo *daemon* (poned `set daemon 60` en `.fetchmailrc` y ejecutadlo una vez, por ejemplo, en `autostart` de Gnome/KDE o en el `.bashrc` –se ejecutará cada 60 segundos–).

***Por ejemplo, en** `/var/spool/cron/crontabs/fetchmail` **agregando** `1 * * * * /usr/bin/fetchmail -s`

Para quitar el correo basura se utilizará SpamAssassin y en esta configuración se ejecutará a traves de Procmail, que es una herramienta muy potente en la gestión del correo (permite repartir el correo, filtrarlo, reenviarlo automáticamente, etc.). Una vez instalado (`apt-get install procmail`), se debe crear un fichero llamado `.procmailrc` en el `home` de cada usuario, que llamará al SpamAssassin:

Podéis instalar SpamAssassin mediante `apt-get install spamassassin`.

```
# Poned yes para mensajes de funcionamiento o depuración
VERBOSE=no
# Consideramos que los mensajes están en “~/.Maildir”), cambiar si es otro
PATH=/usr/bin:/bin:/usr/local/bin:.
MAILDIR=$HOME/Maildir
DEFAULT=$MAILDIR/

# Directorio para almacenar los ficheros
PMDIR=$HOME/.procmail
# Comentar si no queremos log de Procmail
LOGFILE=$PMDIR/log
# filtro de Smap
INCLUDERC=$PMDIR/spam.rc
```

El archivo `~/.procmail/spam.rc` contiene:

```
# si el SpamAssassin no está en el PATH, agregar a la variable PATH el directorio
:0fw: spamassassin.lock
| spamassassin -a

# La tres líneas siguientes moverán el correo Spam a un directorio llamado
# “spam-folder”. Si se quiere guardar el correo en la bandeja de entrada,
# para luego filtrarlo con el cliente, comentad las tres líneas.


:0:
```

```
* ^X-Spam-Status: Yes
spam-folder
```

El archivo `~/.spamassassin/user_prefs` contiene algunas configuraciones útiles para SpamAssassin (consultad la bibliografía).

```
# user preferences file. Ved man Mail::SpamAssassin::Conf

# Umbral para reconocer un Spam.
# Default 5, pero con 4 funciona un poco mejor
required_hits 4

# Sitios de los que consideraremos que nunca llegará Spam
whitelist_from root@debian.org
whitelist_from *@uoc.edu

# Sitios de los que siempre llega SPAM (separados por comas)
blacklist_from viagra@dominio.com

# las direcciones en Whitelist y Blacklist son patrones globales como:
# "amigo@lugar.com", "*@isp.net", o "*.domain.com".

# Insertad la palabra SPAM en el subject (facilita hacer filtros).
# Si no se desea comentar la línea.
subject_tag [SPAM]
```

Esto generará un *tag* `X-Spam-Status: Yes` en la cabecera del mensaje si se cree que el mensaje es *spam*. Luego se deberá filtrar y poner en otra carpeta o borrarlo directamente. Se puede usar el `procmail` para filtrar mensajes de dominios, usuarios, etc. Por último, se puede instalar un cliente de correo y configurar los filtros para que seleccionen todos los correos con `X-Spam-Status: Yes` y los borre o los envíe a un directorio. Después verificaremos los falsos positivos (correos identificados como basura pero que no lo son). Un aspecto complementario de esta instalación es que si se desea tener un servidor de correo a través de correo web (*webmail*, es decir poder consultar los correos del servidor a través de un navegador sin tener que instalar un cliente ni configurarlo, igual que consultar una cuenta de Gmail o Hotmail) es posible instalar Squirrelmail (`apt-get install squirrelmail`) para dar este servicio.

Enlace de interés

Para más información sobre `procmail` y el filtrado de mensajes, consultad: http://www.debian-administration.org/articles/242.

Enlace de interés

Sobre Squirrelmail en Debian, consultad: http://www.debian-administration.org/articles/200.

Enlace de interés

Hay otras posibilidades como instalar MailDrop en lugar de Procmail, Postfix en lugar de Exim, o incluir Clamav/Amavisd como antivirus (Amavisd permite vincular Postfix con SpamAssassin y Clamav). Para saber más sobre este tema podéis visitar la siguiente página web: http://www.debian-administration.org/articles/364.

1.8. Grupos de discusión

Las *news* o grupos de discusión son soportados a través del protocolo NNTP. Instalar un servidor de grupos de discusión es necesario cuando se desea leer *news* fuera de línea, cuando se quiere tener un repetidor de los servidores centrales o se quiere un propio servidor maestro de *news*. Los servidores más comunes son INN o CNEWS, pero son paquetes complejos y destinados a grandes servidores. Leafnode es un paquete USENET que implementa el servidor TNP, especialmente indicado para sitios con grupos reducidos de usuarios, pe-

ro donde se desea acceder a gran cantidad de grupos de noticias. Este servidor se instala en la configuración básica de Debian y se pueden reconfigurar con `dpkg-reconfigure leafnode` todos parámetros como los servidores centrales, el tipo de conexión, etc. Este *daemon* se pone en marcha desde `inetd` de forma similar al `imap` (o con `xinetd`). Leafnode soporta filtros a través de expresiones regulares indicadas (del tipo ^Newsgroups:. * [,] alt.flame$) en `/etc/news/leafnode/filters`, donde para cada mensaje se compara la cabecera con la expresión regular y, si existe coincidencia, el mensaje se rechaza.

La configuración de este servidor es simple y todos los archivos deben ser propiedad de un usuario de *news* con permiso de escritura (se debe verificar que dicho propietario existe en `/etc/passwd`). Todos los archivos de control, *news* y la configuración se encuentran en `/var/spool/news`, excepto la configuración del propio servidor que está en el fichero `/etc/news/leafnode/config`. En la configuración existen algunos parámetros obligatorios que se deben configurar (por ejemplo, para que el servidor pueda conectarse con los servidores maestros), como son `server` (servidor de *news* desde donde se obtendrán y enviarán las *news*) y `expire` (número de días a los que un hilo o sesión se borrará tras haber sido leído). Tenemos, asimismo, un conjunto de parámetros opcionales de ámbito general o específico del servidor que podrían configurarse. Para más información, consultad la documentación (`man leafnode` o `/usr/doc/leafnode/README.Debian`). Para verificar el funcionamiento del servidor, se puede hacer `telnet localhost nntp` y, si todo funciona correctamente, saldrá la identificación del servidor y se quedará esperando un comando. Como prueba, se puede introducir `help` (para abortar, haced `Ctrl+` y luego `Quit`).

1.9. *World Wide Web* (httpd)

Apache es uno de los servidores más populares y con mayores prestaciones de HTTP (*HyperText Transfer Protocol*). Apache tiene un diseño modular y soporta extensiones dinámicas de módulos durante su ejecución. Es muy configurable en cuanto al número de servidores y de módulos disponibles y soporta diversos mecanismos de autentificación, control de acceso, *metafiles*, *proxy caching*, servidores virtuales, etc. Con módulos (incluidos en Debian) es posible tener PHP3, Perl, Java Servlets, SSL y otras extensiones*.

*Podéis consultar la documentación en http://www.apache.org.

Apache está diseñado para ejecutarse como un proceso *daemon standalone*. En esta forma, crea un conjunto de procesos hijos que gestionarán las peticiones de entrada. También puede ejecutarse como *Internet daemon* a través de `inetd`, por lo que se pondrá en marcha cada vez que se reciba una petición pero no es recomendado. La configuración del servidor puede ser extremadamente compleja según las necesidades (consultad la documentación); sin embargo, aquí veremos una configuración mínima aceptable. Su instalación es simple, por ejemplo en Debian, `apt-get install apache2 apache2-doc apache2-utils`. La configuración del servidor estará en */etc/apache2* y por defecto el RootDirectory en */var/www*. Después de su instalación se pondrá

en marcha y llamando a través de un navegador veremos que funciona (nos mostrará el famoso **It works!**). Existen 5 comandos que deberán están en mente de todo administrador: `a2enmod|a2dismod` para habilitar/deshabilitar módulos, `a2ensite|a2dissite` para habilitar|deshabilitar sitios (virtuales) y `apachectl` para gestionar la configuración del servidor (start|stop|restart|graceful|graceful-stop|configtest|status|fullstatus|help).

La configuración de Apach2 en Debian es un poco diferente a la distribución general ya que intenta facilitar al máximo la configuración del servidor en cuanto a módulos, hosts virtuales y directivas de configuración (no obstante rápidamente se puede encontrar las equivalencias con otras distribuciones). Los principales archivos que se encuentran en el directorio */etc/apache2/* son *apache2.conf, ports.conf* y cinco directorios *mods-available|mods-enabled, sites-available|sites-enabled* y *conf.d*. Para información adicional leer */usr/share/doc/apache2.2** y en particular */usr/share/doc/apache2.2-common/README.Debian*.

1) *apache2.conf* es el archivo principal de configuración donde se define a nivel funcional las prestaciones del servidor y se llama a los archivos de configguración correspondientes (*ports, conf.d, sites-enabled*). Se recomienda poner como sufijo *.load* para los módulos que deban ser cargados y *.conf* para las configuraciones pero hay reglas más extensas en cuanto a los sufijos/nombres que pueden ampliarse en la documentación (p. ej., se ignoran todos los archivos que no comienzan por letra o número).

2) *ports.conf* (se incluye en el archivo de configuración global) define los puertos donde se atenderán la conexiones entrantes, y cuales de estos son utilizados en los *host* virtuales.

3) Los archivos de configuración en *mods-enabled/* y *sites-enabled/* son para los sitios activos y los módulos que desean ser cargado en el servidor. Estas configuraciones son activadas creando un link simbólico desde los directorios respectivos ***-available/* utilizando los comandos `a2enmod/a2dismod`, `a2ensite/a2dissite`.

4) Los archivos de *conf.d* son para configuraciones de otros paquetes o agregados pro el administrador y se recomienda que acaben con *.conf*.

5) Para que sea efectiva la configuración por defecto en estos directorios apache2 tendrá que ser gestionado a través de */etc/init.d/apache2* o `service` o `apache2ctl`.

6) El archivo *envvars* es el que contendrá la definición de las variables de entorno y es necesario modificar básicamente dos USER/GROUP que serán con las cuales se ejecutará y obtendrá los permisos. Por defecto se crea el usuario www-data y el grupo www-data (se pueden cambiar). Por lo cual deberá utilizarse `APACHE_RUN_USER=www-data` y `APACHE_RUN_GROUP=www-data`.

Apache también puede necesitar integrar diversos módulos en función de la tecnología que soporte y por lo cual se deberán agregar las librerías/paquetes correspondientes, por ejemplo:

1) Perl: `apt-get install libapache2-mod-perl2`

2) Rugby: `apt-get install libapache2-mod-ruby`

3) Python: `apt-get install libapache2-mod-python`

4) MySQL in Python: `apt-get install python-mysqldb`

5) PHP: `apt-get install libapache2-mod-php5 php5 php-pear php5-xcache`

6) PHP with MySQL: `apt-get install php5-mysql`

1.9.1. Servidores virtuales

Por servidores virtuales se entiende sitios aliados que serán servidos cada uno independiente del otro con sus propios archivos y configuración. En primer lugar deshabilitaremos el sitio por defecto con `a2dissite default`. Los sitios que crearemos serán remix.world y lucix.world que dispondrán de dos archivos de configuración en */etc/apache2/sites-available/* llamados como el dominio.

Contenido del archivo /etc/apache2/sites-available/remix.world.conf

```
<VirtualHost *:80>
    ServerAdmin adminpSySDW.nteum.org
    ServerName remix.world
    ServerAlias www.remix.world
    DocumentRoot /var/www/remix/
    ErrorLog /var/log/apache2/remix-error.log
    CustomLog /var/log/apache2/remix-access.log combined
    Options ExecCGI # habilitar Script en Perl
    AddHandler cgi-script .pl
</VirtualHost>
```

Contenido del archivo /etc/apache2/sites-available/lucix.world.conf

```
<VirtualHost *:80>
    ServerAdmin adminpSySDW.nteum.org
    ServerName lucix.world
    ServerAlias www.lucix.world
    DocumentRoot /var/www/lucix/
    ErrorLog /var/log/apache2/lucix-error.log
    CustomLog /var/log/apache2/lucix-access.log combined
    Options ExecCGI # habilitar Script en Perl
    AddHandler cgi-script .pl
</VirtualHost>
```

Esta configuración es básica y el estudiante deberá consultar la información detallada en [14]. Como se puede observar los directorios raíz para cada dominio estarán en */var/www/remix|lucix* y los archivos de log en /errores/accesos en */var/log/apache2/mmmm-error.log* y *var/log/apache2/nnnn-access.log/*. Para crear los directorios `mkdir -p /var/www/remix`; `mkdir -p /var/www/lucix` y en los cuales se podría poner un *index.html* con alguna identificación que mostrara que dominio se está cargando. por ejemplo para remix.world:

```
<html><body><h1>REMIX: It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been added, yet.</p>
</body></html>
```

Y lo mismo para lucix.world pero cambiando la línea en <h1></h1>. Para los log no debemos hacer nada ya que el directorio */var/log/apache2* ya existe y los archivos los creará el servidor. Finalmente debemos activar los sitios (para ello debemos crear el enlace desde *sites-available* a *sites-enabled*) con `a2ensite remix.world.conf`; `a2ensite lucix.world.conf` y reiniciar apache2 con `service apache2 reload`. Como no disponemos de los dominios en un DNS primario podemos editar */etc/hosts* y agregar para la IP de nuestro servidor (p. ej., 192.168.1.37) dos líneas:

```
192.168.1.37 remix.world
192.168.1.37 lucix.world
```

Luego desde un navegador podremos introducir la URL remix.world y el resultado será la visualización del index.html que nos dirá: **REMIX: It works!**

Una de las ventajas de apache es que puede agregar funcionalidad a través de módulos especializados y que se encontrarán en */etc/apache2/mods-available/*. Para obtener la lista de módulos en disponible para apache podemos hacer por ejemplo `apt-cache search libapache2*`, y para instalarlo `apt-get install [module-name]` los cuales estarán disponibles para su uso (recordad que puede ser necesario alguna configuración adicional en los archivos del sitio). Con `ls -al /etc/apache2/mods-available/` podemos mirar los disponibles e instalarlo con `a2enmod [module-name]`. Para listar los módulos cargados podemos hacer `/usr/sbin/apache2 -M` que nos listará con *shared* los cargados dinámicamente y con *static* los que se encuentran compilados con el servidor (estos se puede obtener también con `apache2 -l`). Los módulos en el directorio *mods-available* tienen extensiones *.load* (indica la librería a cargar) y *.conf* (configuración adicional del módulo) pero cuando utilizamos el comando `a2enmod` solo se debe indicar el nombre del módulo sin extensión. Para deshabilitar un módulo `a2dismod [module-name]`.

Como muestra de estas propiedades configuraremos un sitio seguro (https) bajo el dominio remix.world pero que redirigiremos al directorio */var/www/remix.ssl*. En primer lugar crearemos un certificado (autofirmado) para nuestro sitio con la orden `make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/ssl/private/remix.crt` indicándole el dominio que queremos validar (remix.world en nuestro caso) -solo introducir el dominio y dejar los alias en blanco- y si no podemos ejecutar `make-ssl-crt` asegurarnos que tenemos el paquete ssl-cert. Luego activamos el módulo SSL con `a2enmod ssl`, creamos el directorio */var/www/remix.ssl* y modificamos el index.html como hicimos con los anteriores. A continuación modificamos creamos la configuración del sitio (podemos utilizar la que viene por defecto modificándola):

```
cd /etc/apache2/sites-available; cp default-ssl remix.world.ssl.conf
```

Editamos el archivo remix.world.ssl.conf (solo mostramos las líneas principales/modificadas):

```
<IfModule mod_ssl.c>
<VirtualHost _default_:443>
    ServerAdmin adminpSySDW.nteum.org
    DocumentRoot /var/www/remix.ssl
    <Directory />
        Options FollowSymLinks
        AllowOverride None
    </Directory>
    <Directory /var/www/remix.ssl>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride None
        Order allow,deny
        allow from all
    </Directory>
# líneas igual que el archivo original...
    ErrorLog $APACHE_LOG_DIR/remix.world.ssl_error.log
    CustomLog $APACHE_LOG_DIR/remix.world.ssl_access.log combined

    SSLEngine on
    SSLCertificateFile /etc/ssl/private/remix.crt
    #SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
# líneas igual que el archivo original...
</VirtualHost>
</IfModule>
```

Finalmente nos queda activar el sitio (`a2ensite remix.world.ssl.conf`), reiniciar apache2 (`service apache2 reload`) y desde el navegador hacer *https://remix.world* que como el certificado es autofirmado nos hará una advertencia y aceptaremos el certificado y deberemos obtener **SSL - REMIX: It works!**.

Un aspecto interesante es la función del archivo *.htaccess** en los directorios de nuestro dominio. Este archivo se puede utilizar para control de acceso al sitio (p. ej., habilitar/restringir IP), control de acceso a carpetas, listados, redirecciones (p. ej., a otra página/site, a otra carpeta, a otro dominio, a https, ...), evitar el *hotlinking* (para evitar que nos hagan enlaces a ficheros -generalmente vídeos- y consuman ancho de banda de nuestro servidor), cambiar la página por defecto, crear URL amigables, favorecer el cache de nuestro sitio, etc. Como muestra de ello para evitar por ejemplo que una carpeta sea inaccesible solo basta poner un archivo *.htaccess* en ella con el contenido `deny from all`. Para permitir que una carpeta de nuestro sitio (p. ej., del dominio remix.com/valid-user) tenga acceso con un usuario y passwd deberemos crear dentro de ella un archivo *.htaccess* con el siguiente contenido (también podemos crear un index.html modificado dentro de esta carpeta para verificar el funcionamiento):

***http://httpd.apache.org/docs/2.2/howto/htaccess.htm**

```
AuthName “Restricted Area”
AuthType Basic
AuthUserFile /etc/apache2/htpasswd
AuthGroupFile /dev/null
require valid-user
```

Para crear el usuario ejecutamos `htpasswd -c /etc/apache2/htpasswd adminp` que nos pedirá el passwd para este usuario y los almacenará en el archivo indicado. Luego ponemos como URL *http://remix.world/valid-user/* nos

pedirá el usuario (adminp) y el passwd que almacenamos y veremos: **REMIX->Valid-User: It works!**. En caso contrario nos continuará pidiendo el usuario/passwd y si hacemos Cancel no indicará un mensaje de Authorization Required impidiendo el acceso.

1.9.2. Apache + PHP + Mysql + PhpMyAdmin

Una cuestión importante para los servidores web dinámicos es aprovechar las ventajas de Apache PHP (un lenguaje de programación usado generalmente para la creación de contenido para sitios web) y una base de datos como MySQL incluyendo un programa adminsitrador de MySQL como PHPMyAdmin, todo ello funcionando conjuntamente. Las distribuciones han evolucionado mucho y en Debian es sumamente fácil poner en marcha este conjunto (pero tampoco representa ninguna dificultad bajarse el software fuente, compilarlo e instalarlo si se desea, por ejemplo, tener las últimas versiones de los paquetes por algún motivo pero recordad que implicará más trabajo y dedicación).

En primer lugar instalamos PHP con `apt-get install php5` que nos indicará que cambiará el modo de trabajo de apache2 instalando otro. Esta cuestión es porque la configuración por defecto de apache trabaja con una herramienta llamada MPM-worker. Este es un módulo de multi-procesamiento que puede manejar múltiples peticiones rápidamente utilizando múltiples *threads* por proceso cliente. Sin embargo este no es compatible con alguna extensiones PHP y por ello el MPM-worker es reemplazado por MPM-prefork, que permite manejar todas las peticiones PHP (en modo compatibilidad) y evitar que si una petición falla pueda afectar a otras peticiones. Existe otro módulo llamado mpm-itk (http://mpm-itk.sesse.net/) que es similar prefork pero tiene mejores prestaciones y gestión de permisos (consultar la bibliografía en apache.org). Para verificar que PHP funciona creamos un fichero por ejemplo dentro de RootDirectory de remix.world llamado *test.php* con el siguiente contenido: *<?php phpinfo() ?>* y si en la URL introducimos *http://remix.world/test.php* deberemos ver una tabla con la versión e información sobre el paquete PHP instalado.

Para instalar los paquetes MySQL y PHPMyAdmin haremos `apt-get install mysql-server` (es importante recordar la contraseña de acceso que introduzcamos pero siempre podemos hacer `dpkg-reconfigure mysql-server` que perderemos todo lo que haya en la BD pero también hay otros métodos -menos agresivos- para recuperar la contraseña del root). Luego para instalar PHPMyAdmin haremos `apt-get install phpmyadmin` y prestar atención que nos pedirá la clave de acceso para entrar en la base de datos y crear una clave de acceso para entrar en la aplicación vía un navegador. Luego podremos poner en la URL de nuestro navegador *http://localhost/phpmyadmin*, nos solicitará el usuario (root generalmente) y el passwd que hemos introducido y ya podremos gestionar el servidor de bases de datos MySQL.

1.9.3. Otros servidores httpd

Lighttpd es un servidor web (con licencia BSD) diseñado para ser rápido, seguro, flexible, que implementa la mayoria de los estándares y está optimizado para entornos donde la velocidad es muy importante (consume menos CPU/-RAM que otros servidores) y es muy apropiado para cualquier servidor que tenga que dar soporte a grandes cargas. Entre sus principales características están la de Virtual hosting, redirecciones http y reescrituras de URL, dar soporte a CGI, SCGI y FastCGI, PHP, Ruby, Python entre otros y además con consumo de memoria constante.

Su instalación en Debian es `apt-get install lighttpd`, y si tenemos apache sobre el puerto 80 nos dará un error. Para ello debemos editar el archivo `/etc/lighttpd/lighttpd.conf` y cambiar la línea `server.port = 8080` y reiniciar `service lighttpd start`. Desde el navegador se puede hacer *http://localhost:8080index.lighttpd.html* y veremos la página inicial de lighttpd. Por defecto Lighttpd tiene su directorio raíz en */var/www* (en Debian) y el archivo de configuración en */etc/lighttpd/lighttpd.conf*. Configuraciones adicionales están en */etc/lighttpd/conf-available* y puede ser habilitadas con el comando `lighttpd-enable-mod` el cual crea enlaces entre *conf-enabled* y *conf-available*, las cuales se pueden deshabilitar con `lighttpd-disable-mod`.

Para habilitar el servidor de FastCGI con la intención de ejecutar PHP deberemos instalar PHP-FPM con `apt-get install php5-fpm php5` y sobre el archivo */etc/php5/fpm/php.ini* quitar el comentario a `cgi.fix_pathinfo=1`. Luego deberemos activar el servidor PHP-FPM, por lo cual haremos una copia del archivo original y lo modificaremos:

```
cd /etc/lighttpd/conf-available/
cp 15-fastcgi-php.conf 15-fastcgi-php-spawnfcgi.conf
```

Modificar *15-fastcgi-php.conf* con:

```
# -*- depends: fastcgi -*-

# Start an FastCGI server for php
fastcgi.server += ( ".php" =>
        ((
                "socket" => "/var/run/php5-fpm.sock",
                "broken-scriptfilename" => "enable"
        ))
)
```

Para habilitar fastcgi deberemos cargar los módulos `lighttpd-enable-mod fastcgi` y `lighttpd-enable-mod fastcgi-php` lo cual crea los enlaces correspondientes que podemos ver con `ls -l /etc/lighttpd/conf-enabled`. Luego podemos reiniciar `service lighttpd force-reload`. Para visualizar si el servidor y FastCGI funciona creamos un archivo */var/www/info.php* con el siguiente contenido `<?php phpinfo(); ?>` y podremos visualizar (http://localhost:8080/info.php) la página de configuración de PHP donde indica como Server API = FPM/FastCGI.

Otro servidor muy utilizado actualmente es **Nginx** (http://nginx.org/) programado en C y licencia BSD. Sus funciones principales son como servidor web/proxy inverso de muy alto rendimiento (puede soportar más de 10.000 conexiones simultáneas) y también puede funcionar como proxy para protocolos de correo electrónico (IMAP/POP3). Es un servidor utilizado por grandes instalaciones (WordPress, Netflix, Hulu, GitHub, y partes de Facebook entre otros) y entre sus principales características están (además de servidor de archivos estáticos, índices y autoindexado y proxy inverso con opciones de caché) el balanceo de carga, tolerancia a fallos, SSL, FastCGI, servidores virtuales, streaming de archivos (FLV y MP4.8), soporte para autenticación, compatible con IPv6 y SPDY. Su instalación básica es simple y para su configuración se puede consultar la wiki de nginx a http://wiki.nginx.org/Configuration.

1.10. Servidor de WebDav

El nombre webDAV son las siglas de *Web Based Distributed Authoring and Versioning* (también se refiere al grupo de trabajo de Internet Engineering Task Force) y es un protocolo que permite que la web se transforme en un medio legible y editable y proporciona funcionalidades para crear, cambiar y mover documentos en un servidor remoto (típicamente un servidor web). Esto se utiliza sobre todo para permitir la edición de los documentos que envía un servidor web, pero puede también aplicarse a sistemas de almacenamiento generales basados en la web y a los que se puede acceder desde cualquier lugar. En este subapartado instalaremos un servidor WebDav sobre Apache. El proceso es el siguiente:

Enlace de interés

Sobre la integración de WebDav con Apache podéis consultar el artículo "WebDAV on Apache2" disponible en: http://www.debian-administration.org/articles/285

1) Verificar que tenemos instalado apache2 y si no realizar su instalación como hemos visto anteriormente y verificar que funciona (`apt-get install apache2`).

2) Habilitar los módulos de Apache que son necesarios para WebDav: `a2enmod dav_fs` y `a2enmod dav`.

3) Crear el directorio para el directorio virtual (podemos hacer por ejemplo `mkdir -p /var/www/webdav`) y permitir que Apache sea el propietario del directorio `chown www-data /var/www/webdav/`.

4) Crear el archivo `/etc/apache2/sites-available/webdav.conf` para definir la funcionalidad del servidor (en esta configuración estamos configurando todo el servidor como WebDav pero podría ser un servidor virtual y en modo SSL para mayor seguridad):

```
<VirtualHost *:80>
    ServerAdmin adminpSySDW.nteum.org
    DocumentRoot /var/www/webdav/
    ErrorLog /var/log/apache2/webdav-error.log
    CustomLog /var/log/apache2/webdav-access.log combined
   <Directory /var/www/webdav>
        Options Indexes MultiViews
        AllowOverride None
        Order allow,deny
```

```
        allow from all
        DAV On
        AuthName “Restricted WeDav Area”
        AuthType Basic
        AuthUserFile /etc/apache2/htpasswd
        AuthGroupFile /dev/null
        require valid-user
    </Directory>
</VirtualHost>
```

Se puede comprobar que la configuración es correcta con la orden `apache2ctl configtest`.

5) Como se ha hecho anteriormente, se crean los usuarios (`htpasswd [-c] /etc/apache2/htpasswd` usuario) indicando el -c si es el primer usuario a crear.

6) Se reinicia Apache para que lea la configuración `/etc/init.d/apache2 reload` y ya nos podemos conectar a `http://localhost`, previa autentificación.

7) Desde un GNU/Linux podemos probar la funcionalidad del servidor abriendo el nautilus (gestor de ficheros) y desde el menu *File->Connect to Server* podemos seleccionar Servidor WebDav introduciendo los datos (IP, directorio, usuario, passwd) y tendremos acceso como si de una carpeta local se tratara.

8) Desde MacOS podemos utilizar el mismo procedimiento que el anterior desde el gestor de archivos o instalar un cliente específico (igualmente para Windows). El más recomendado para ello es CyberDuck (http://cyberduck.io/) que tiene licencia GPL y es una excelente aplicación (soporta múltiples protocolos) y muy fácil de configurar.

9) Otra forma de probarlo es con un cliente WebDav (en modo texto), por ejemplo Cadaver (http://www.webdav.org/cadaver), con `apt-get install cadaver`. Después, nos conectamos al servidor con `cadaver IP-nombre del servidor` y después de autentificarnos, podemos crear un directorio (`mkdir`), editar un archivo, listar un directorio (`ls`), cambiar de directorio (`cd`), cambiar los permisos de ejecución (`chexec`), borrarlo (`rm`), etc.

En muchas ocasiones, y dado que estaremos haciendo transferencias de archivos, es importante preservar la privacidad por lo cual sería adecuado trabajar con WebDav pero sobre SSL. Su configuración no implica mayores complicaciones y veremos una forma diferente de generar los certificados (serán autofirmados -hasta que podamos tener nuestra propia entidad certificadora-) para un dominio en particular, en nuestro caso webdav.world. Para ello hacemos:

1) Nos cambiamos al directorio donde almacenaremos los certificados: `cd /etc/ssl/private`. Hacemos la petición del certificado:

```
openssl req -config /etc/ssl/openssl.cnf -new -out webdav.csr
```

Este comando nos pedirá un *passwd* y una serie de información que quedará en el certificado pero la más importante es Common Name (CN) que será donde validará el certificado (en nuestro caso webdav.world). Podemos verificar la petición con:

```
openssl req -in /etc/ssl/private/webdav.csr -noout -text
```

2) Creamos la llave (*key*):

```
openssl rsa -in privkey.pem -out webdav.key
```

3) Firmamos:

```
openssl x509 -in webdav.csr -out webdav.crt -req -signkey webdav.key
-days 3650
```

Podemos verificarlo con:

```
openssl x509 -noout -in /etc/ssl/private/webdav.crt -text
```

4) Generamos en certificado en formato DER:

```
openssl x509 -in webdav.crt -out webdav.der.crt -outform DER
```

Se puede verificar con:

```
openssl x509 -in /etc/ssl/private/webdav.der.crt -inform der
-noout -text
```

5) Ahora generamos el archivo de configuración de apache a partir del anterior:

```
cd /etc/apache2/sites-available; cp webdav.conf webdav-ssl.conf
```

Modificamos para incluir las siguiente cuatro líneas al inicio y modificar el VirtualHost:

<VirtualHost *:443>
ServerName webdav.world
SSLEngine on
SSLCertificateFile /etc/ssl/private/webdav.crt
SSLCertificateKeyFile /etc/ssl/private/webdav.key
...

Solo nos resta activar el sitio (`a2ensite webdav-ssl.conf`), reiniciar apache (`service apache2 restart`) y verificar que funciona en la dirección https://webdav.world/ previa aceptación del certificado.

1.11. Servicio de *proxy*: Squid

Un servidor *proxy* (*proxy server*, PS) se utiliza para ahorrar ancho de banda de la conexión de red, mejorar la seguridad e incrementar la velocidad para obtener páginas de la red (*web-surfing*).

Squid es un *proxy caching server* para Web y da soporte a los protocolos HTTP, HTTPS, FTP, entre otros. Éste reduce el ancho de banda y mejora los tiempo de respuesta almacenado en caché y reutilizando las páginas más frecuentes. Squid tiene un extenso conjunto de reglas de control que permiten optimizar el flujo de datos ente el cliente y el servidor aportando seguridad y control y encaminando las peticiones correctamente, permitiendo su control y mejorando la utilización del ancho de banda de la red. Squid tiene diferentes modos de funcionamiento pero como los más importantes podemos mencionar: *Forward Proxy* (es el modo básico sobre el cual se configuran todo lo demás),

Transparent o *Interception Proxy* (permite incluir un proxy en una red sin que los clientes tengan que configurar nada) y *Reverse Proxy* -o *Accelerator-mode*- (permite ejecutar squid para mejorar la respuesta de una granja de servidores web).

Para instalar Squid como *proxy-cache* (http://www.squid-cache.org/) en Debian hacemos `apt-get install squid3` y editaremos el archivo de configuración */etc/squid3/squid.conf* para realizar una configuración básica. Se debe tener en cuenta que squid es muy potente pero esto se traduce en una configuración que puede ser compleja; como idea simplemente considerar que el archivo de configuración, que está muy bien explicado, tiene aproximadamente 5700 líneas (no obstante si ejecutamos

```
grep -v "^#" /etc/squid3/squid.conf | awk '$1 != "" {print $0}'
```

podremos ver que la configuración básica son unas 40 líneas mientras que el resto son comentarios y opciones comentadas).[16][17]

Definir la ACL (access control list) para habilitar la red/ip que deseamos hacer de proxy, en la línea 718 agregar: `acl lan src 192.168.1.0/24`

Permitir el acceso, en la línea 842 (aprox.) agregar: `http_access allow lan`
Cambiar el puerto de acceso (linea 1136), por defecto está `http_port 3128` y se puede dejar que es el estándar para squid o poner el que se prefiera (por ejemplo 8080).

Agregar las reglas de control, en la línea 3449 (aprox.):
request_header_access Referer deny all
request_header_access X-Forwarded-For deny all
request_header_access Via deny all
request_header_access Cache-Control deny all

Definimos el nombre visible, línea 3748 (aprox.): `visible_hostname remix.world`

Modificamos visibilidad de la IP, línea 5541 (aprox.): `forwarded_for off`

Finalmente reiniciamos el servidor: `service squid3 restart`
Nos dará un mensaje similar a:

[ok] Restarting Squid HTTP Proxy 3.x: squid3[....] Waiting.....done.
(tardará unos segundos...)

Con el servidor en marcha podemos configurar los navegadores para que utilicen el proxy, esto es por ejemplo en IceWeasel/Firefox en el apartado de *Preferences*/*Options->Advanced->Network->Proxy* e introducir los datos de nuestro servidor. Sobre Chrome por ejemplo en Windows, se debe ir a las *Settings->Advanced->Change proxy setting->Connections->Lan Settings* e introducir los datos de nuestro servidor.

Otra de las opciones que permite Squid es actuar como *Reverse Proxy* que es un tipo de *Proxy* donde los datos se recuperan de un/unos servidor/es y de devuelven a los clientes como si se originaran en el *proxy* quedando los servidores ocultos a los clientes como se muestra en la Wikipedia*. Esto permite hacer políticas de balanceo de carga y que las peticiones estén en un único dominio a pesar que internamente pueden estar distribuidas en diversos servidores. Para su configuración debemos hacer:

***http://upload.wikimedia.org /wikipedia/commons/6/67 /Reverse_proxy_h2g2bob.svg**

Especificar la dirección del servidor interno, en la línea 1136 modificar:
`http_port 80 defaultsite=192.168.1.33`
Agregar cache_peer, en la línea 1937 (aprox.) agregar:
`cache_peer 192.168.1.33 parent 80 0 no-query originserver`
Cambiar la acl para permitir cualquier conexión, en la línea 842 (aprox.) modificar `http_access allow all`
Finalmente reiniciamos el servidor: `service squid3 restart`

Cuando nos conectemos a la IP/dominio del servidor *Proxy* en realidad veremos las páginas enviadas por el servidor 192.168.1.33. Como prueba de concepto (si queremos hacer la prueba con una única máquina) podemos poner que en lugar del servidor 192.168.1.33 poner un servidor externo (p. ej., debian.org o su IP) y cuando pongamos como URL la de nuestro dominio visualizaremos la página de debian.org.

Otras de las configuraciones ampliamente utilizadas es como *interception proxy* (o *transparent proxy*) que intercepta la comunicación normal a la capa de red sin necesidad de configuraciones específicas en el cliente y por lo cual no sabrán que están detrás de un *proxy*. Generalmente un *transparent proxy* esta normalmente localizado entre el cliente e Internet con el *proxy* haciendo las funciones de *router* o *gateway*. Es habitual en las instituciones que desean filtrar algún tráfico de sus usuarios, ISP para hacer caché y ahorrar ancho de banda o países que controlan el acceso a determinados sitios por parte de sus ciudadanos. Para implementarlo sobre una institución lo habitual es disponer de una máquina con dos interfaces de red, una conectada a la red interna y otra hacia la red externa. Los pasos para su configuración serán básicamente los descritos en http://wiki.squid-cache.org/ConfigExamples/Intercept/LinuxDnat:

Sobre squid.conf:

Modificar la `acl/http_access` sobre las IP, IP/Mask permitidas como en el primer caso.

Configurar el puerto/modo como `http_port 3129 transparent` o en Squid 3.1+ se deberá utilizar http_port 3129 intercept para interceptar paquetes DNAT.

Sobre /etc/sysctl.conf modificar:

Permitir el packet forwarding: net.ipv4.ip_forward = 1

Controlar el source route verification: net.ipv4.conf.default.rp_filter = 0

: No aceptar source routing: net.ipv4.conf.default.accept_source_route = 0

Considerando que la IP de Proxy está en la variable SQUIDIP y el puerto esta en SQUIDPORT incluir las siguientes reglas de DNAT:

```
iptables -t nat -A PREROUTING -s $SQUIDIP -p tcp --dport 80 -j ACCEPT
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination
    $SQUIDIP:$SQUIDPORT
iptables -t nat -A POSTROUTING -j MASQUERADE
iptables -t mangle -A PREROUTING -p tcp --dport $SQUIDPORT -j DROP
```

1.11.1. Proxy SOCKS

SOCKS (abreviación de SOCKetS) es un protocolo de Internet (en el modelo OSI estaría en una capa intermedia entre la de aplicación y la de transporte) que permite a las aplicaciones en modo cliente-servidor atravesar en forma transparente un *firewall* de red. Los clientes que hay detrás de un *firewall*, los cuales necesitan acceder a los servidores del exterior, pueden conectarse en su lugar a un servidor *proxy* SOCKS. Tal servidor *proxy* controla qué cliente puede acceder al servidor externo y pasa la petición al servidor. SOCKS puede ser usado también de la forma contraria, permitiendo a los clientes de fuera del *firewall* (clientes externos) conectarse a los servidores de dentro del *firewall* (servidores internos).[26][27]

Se debe tener en cuenta que SOCKS solo sirve en modo cliente/servidor por o cual un usuario debe tener instalado un cliente SOCKS, ya sea en la aplicación (como Firefox, Chrome) o dentro de la pila TCP/IP desde donde el software del cliente redirige los paquetes en un túnel SOCKS. El procedimiento habitual comienza cuando el cliente SOCKS (p. ej., interno en una red privada) inicia una conexión a un servidor SOCKS (el protocolo SOCKS permite la autentificación y el registro de las solicitudes de conexión) y este (servidor SOCKS) actuará como cliente IP para la solicitud de conexión (del cliente interno en su nombre) lo cual significa que el servidor externo solo será consciente de las peticiones del servidor SOCKS (por lo cual actuará como *proxy forwarding*).

La pregunta habitual es si SOCKS es diferente a resolver el problema de acceso externo mediante NAT. La respuesta es: sí, es diferente, ya que los paquetes en NAT solo modifican las direcciones (p. ej., cambian las IP privadas por las pública del *router* como ocurre en un *router* ADSL) y el servidor recibe/contesta las peticiones. La sesión IP se establece directamente desde el cliente al servidor y el router/FW solo modifica/filtra el paquete pero no hay autenticación ni inspección del paquete/aplicación/datos (se podría hacer pero es complejo). La ventajas de SOCKS radican en que provee autentificación para protocolos que no lo permiten, puede traspasar el *routing* por defecto de una red interna, si bien HTTP y Telnet soportan autentificación por firewall (p. ej., utilizando *Authenticated Proxy* on a Cisco *firewall*) los protocolos encriptados nunca pueden ser autentificados por un FW en cambio SOCKS si puede hacerlo. No obstante existen desventajas ya que el cliente debe tener interface a SOCKS, el SO cliente debe tener interface a SOCKS (para interceptar el tráfico y reenviarlo al SOCKS *proxy*) y necesitaremos un servidor SOCKS específico para este fin.

Un caso de uso habitual es si consideramos que estamos en un punto de conexión inalámbrica abierta (p. ej., Wifi) y no se desea enviar datos de navegación sobre texto plano o se quiere acceder a una página web filtrada por router/FW perimetral. Una solución muy simple es utilizando SSH (que incluye un servidor SOCKS) que puede cifrar todo el tráfico de navegación por la web y redireccionarlo a través de un equipo de confianza cuando se está en algún otro punto de la red. Para ello deberemos disponer de un servidor SSH para que actúe como representante en un ordenador remoto (que le permita conectarse a él a través de SSH) y un cliente de SSH en el equipo que está utilizando. Lo que se hará con un proxy es la creación de un *middle-person* entre el usuario e Internet. El navegador hará las solicitudes de páginas web al servidor *proxy*, que controla la solicitud y obtiene la página desde internet y las devuelve al cliente. El sitio web en realidad piensa que la solicitud proviene del servidor *proxy*, no del equipo que la ha originado ocultando las dirección IP de origen. Además, la conexión entre el ordenador y el *proxy* que pasa a través de SSH es cifrada y esto evita que alguien pueda obtener los paquetes desde la Wifi (*sniffers* de wifi) en el sitio de la conexión.

Para su configuración desde donde nos conectamos debemos tenemos acceso a un servidor SSH y sobre el que crearemos un túnel que pasará el tráfico web entre nuestra máquina local y el proxy SOCKS sobre SSH. Para ello ejecutamos sobre nuestra máquina `ssh -ND 9999 login@remote-server.org` donde deberemos reemplazar login@remote-server.org con el usuario y nombre o IP del servidor remoto. Lo que está haciendo este comando es un *port forwarding* a través del puerto 9999 (puede ser cualquier otro pero conviene que sea superior a 1024 para evitar que solo lo pueda hacer root) y la conexión se reenvía a través de una canal seguro donde el protocolo de aplicación se utiliza para determinar dónde conectar desde la máquina remota. Actualmente OpenSSH soporta los protocolos SOCKS4-5 y por ello actuará como un servidor SOCKS. A continuación se solicitará el passwd y una vez autenticado no pasará NADA (el -N indica que no abra un prompt interactivo pero continuará funcionando). Si por el *firewall* solo podemos salir por e puerto 443 por ejemplo deberíamos configurar el ssh server para escuchar por el puerto 443 y en su lugar ejecutar `ssh -ND 9999 login@remote-server.org -p 443`. Ahora es necesario configurar el cliente para conectarse al proxy, Por ejemplo Firefox: *Options ->Advanced ->Network ->Connection* y seleccionar SOCKS, como nombre del servidor *localhost* (o su nombre real si tiene) y el puerto (9999) y guardar los ajustes y verificar que podemos navegar sin problemas. Se puede utilizar el plugin FoxyProxy (https://addons.mozilla.org/es/firefox/addon/foxyproxy-standard/) para Firefox que permite cambiar entre el proxy y la conexión directa en función del sitio o de un control. Como medida adicional (de anonimato) se puede configurar el servidor *proxy* para resolver peticiones DNS en lugar del método habitual haciendo en Mozilla Firefox poniendo como URL: `about:config` y modificando `network.proxy.socks_remote_dns=true`. También para conexiones lentas se puede utilizar la opción -C de ssh para utilizar la compresión de SSH por gzip. En Thunderbird u otros clientes la configuración es similar.

Si el túnel deja de funcionar (suele ocurrir en redes muy ocupadas) se puede utilizar el paquete `autossh` en lugar del `ssh` para establecer la conexión que se encargará de mantener el túnel abierto reiniciando automáticamente la conexión. Otro paquete interesante es `tsocks` (http://tsocks.sourceforge.net/) que se puede utilizar cuando el cliente que deseamos utilizar no soporta el protocolo SOCKS. `tsocks` monitoriza la llamada de inicio de sesión de una aplicación (*connect*) y redirecciona la comunicación hacia el *server* SOCKS sin que la aplicación tenga ninguna información. Para ello se debe instalar tsocks y configurar el proxy SOCKS que deberá utilizar en el fichero */etc/tsocks.conf* indicándole los valores (p. ej., server = 127.0.0.1, server_type = 5, server_port = 9999). Luego bastará con llamar a la aplicación con `tsocks aplicación` o simplemente el comando que abrirá una nueva shell redirigida al proxy y por lo cual todo lo que se ejecute allí será enviado al proxy SOCKS.

1.12. OpenLdap (Ldap)

LDAP significa *Lightweight Directory Access Protocol* y es un protocolo para acceder a datos basados en un servicio X.500. Este se ejecuta sobre TCP/IP y el directorio es similar a una base de datos que contiene información basada en atributos. El sistema permite organizar esta información de manera segura y utiliza réplicas para mantener su disponibilidad, lo que asegura la coherencia y la verificación de los datos accedidos-modificados.

El servicio se basa en el modelo cliente-servidor, donde existen uno o más servidores que contienen los datos; cuando un cliente se conecta y solicita información, el servidor responde con los datos o con un puntero a otro servidor de donde podrá extraer más información; sin embargo, el cliente solo verá un directorio de información global [28, 19]. Para importar y exportar información entre servidores `ldap` o para describir una serie de cambios que se aplicarán al directorio, el formato utilizado se llama LDIF (LDAP *Data Interchange Format*). LDIF almacena la información en jerarquías orientadas a objetos que luego serán transformadas al formato interno de la base de datos. Un archivo LDIF tiene un formato similar a:

```
dn: o = UOC, c = SP o: UOC
objectclass: organization
dn: cn = Remix Nteum, o = UOC, c = SP
cn: Remix Nteum
sn: Nteum
mail: nteumuoc.edu
objectclass: person
```

Cada entrada se identifica con un nombre indicado como `DN` (*Distinguished Name*). El DN consiste en el nombre de la entrada más una serie de nombres que lo relacionan con la jerarquía del directorio y donde existe una clase de objetos (`objectclass`) que define los atributos que pueden utilizarse en esta entrada. LDAP provee un conjunto básico de clases de objetos: **grupos** (inclu-

ye listas desordenadas de objetos individuales o grupos de objetos), **localizaciones** (tales como países y su descripción), **organizaciones y personas**. Una entrada puede, además, pertenecer a más de una clase de objeto, por ejemplo, un individuo es definido por la clase persona, pero también puede ser definido por atributos de las clases `inetOrgPerson`, `groupOfNames` y `organization`.

La estructura de objetos del servidor (llamado *schema*) determina cuáles son los atributos permitidos para un objeto de una clase (que se definen en el archivo `/etc/ldap/schema` como `inetorgperson.schema, nis.schema, opeldap.schema, corba.schema`, etc.). Todos los datos se representan como un par atributo = valor, donde el atributo es descriptivo de la información que contiene; por ejemplo, el atributo utilizado para almacenar el nombre de una persona es `commonName`, o cn, es decir, una persona llamada Remix Nteum se representará por cn: Remix Nteum y llevará asociado otros atributos de la clase persona como `givenname: Remix surname: Nteum mail: nteum@uoc.edu`. En las clases existen atributos obligatorios y optativos y cada atributo tiene una sintaxis asociada que indica qué tipo de información contiene el atributo, por ejemplo, `bin` (*binary*), `ces` (*case exact string*, debe buscarse igual), `cis` (*case ignore string*, pueden ignorarse mayúsculas y minúsculas durante la búsqueda), `tel` (*telephone number string*, se ignoran espacios y '-') y `dn` (*distinguished name*). Un ejemplo de un archivo en formato LDIF podría ser:

```
dn: dc = UOC, dc = com
objectclass: top
objectclass: organizationalUnit


dn: ou = groups, dc = UOC, dc = com
objectclass: top
objectclass: organizationalUnit
ou: groups


dn: ou = people, dc = UOC, dc = com
objectclass: top
objectclass: organizationalUnit
ou: people


dn: cn = Remix Nteum, ou = people, dc = UOC, dc = com
cn: Remix Nteum
sn: Nteum
objectclass: top
objectclass: person
objectclass: posixAccount
objectclass: shadowAccount
   uid:remix userpassword:{crypt}p1pss2ii(0pgbs*do& = )eksd uidnumber:104
   gidnumber:100
   gecos:Remix Nteum
   loginShell:/bin/bash
   homeDirectory: /home/remix
   shadowLastChange:12898
   shadowMin: 0
   shadowMax: 999999
   shadowWarning: 7
   shadowInactive: -1
   shadowExpire: -1
   shadowFlag: 0
```

```
dn:
cn = unixgroup, ou = groups, dc = UOC, dc = com
objectclass: top
objectclass: posixGroup
cn: unixgroup
   gidnumber: 200
   memberuid: remix
   memberuid: otro-usuario
```

Las líneas largas pueden continuarse debajo comenzando por un espacio o un tabulador (formato LDIF). En este caso, se ha definido la base DN para la institución `dc = UOC, dc = com`, la cual contiene dos subunidades: `people` y `groups`. A continuación, se ha descrito un usuario que pertenece a `people` y a `group`. Una vez preparado el archivo con los datos, este debe ser importado al servidor para que esté disponible para los clientes LDAP. Existen herramientas para transferir datos de diferentes bases de datos a formato LDIF [19]. Sobre Debian, se debe instalar el paquete `slapd` y `ldap-utils`, que es el servidor de OpenLdap y un conjunto de utilidades para acceder a servidores locales y remotos. Durante la instalación, solicitará un passwd para gestionar la administración del servicio, se generará una configuración inicial y se pondrá en marcha el servicio que se puede verificar con el comando `slapcat` que tomando la información disponible (FQDN de */etc/hosts*) generará algo como:

```
   dn: dc=nteum,dc=org
   objectClass: top
   objectClass: dcObject
   objectClass: organization
   o: nteum.org
   dc: nteum
...

   dn: cn=admin,dc=nteum,dc=org
   objectClass: simpleSecurityObject
   objectClass: organizationalRole
   cn: admin
   description: LDAP administrator     userPassword:: e1NTSEF9TU9jVE1qWllPVFBmd2FiZWJtSjcrY0pYd2wvaTk5aUc=
...
```

Con los comandos `ldapadd` y `ldapdelete` se pueden agregar/borrar registros de la tabla del servicio utilizando normalmente archivos de texto (sobre todo para el `add`) donde estén definidos los nuevos registros. Si bien no es un procedimiento complejo para gestionar el servidor puede ser más adecuado, en los primeros pasos, utilizar algunas de las aplicaciones que permiten gestionar el servidor de un forma más amigable como por ejemplo `phpldapadmin` (administración por medio apache+php), `jxplorer` (aplicación en java), `gosa` o `lat` (todos ellos en la mayoría de las distribuciones). En nuestro caso instalaremos `phpldapadmin` que combina simplicidad y permite gestionar la mayoría de las opciones del servidor Ldap.

Para instalarlo ejecutamos `apt-get install phpldapadmin` el cual ya reiniciará el servidor apache pero sino lo hace podemos reiniciarlo con la orden `/etc/init.d/apache2 restart`. Para conectarnos en un navegador introducimos `http://localhost/phpldapadmin/`. Saldrá la página de bienvenida y en la banda izquierda podremos hacer el *login* (si el *ServerName* de

apache no coincide con el LDAP no pedirá como usuario DN por lo cual le deberemos indicar el correcto cn=admin,dc=nteum,dc=org y la contraseña del servidor LDAP). Una vez conectado al servidor, seleccionamos la raíz (dc=nteum,dc=org) y seleccionamos entre los *templates* que aparecen una *Organizational Unit (ou)* a la que llamaremos `users`, repetimos los pasos (opción `create new entry` desde la raíz) y creamos otra *ou* llamada `groups`. Ahora solamente nos queda crear los usuarios y los grupos y asignar los usuarios a sus grupos. Dentro de la unidad organizativa `groups` crearemos los grupos `ventas` (gid=1001) y `compras` (gid=1002) y en la unidad organizativa `users` crearemos los usuarios `juan pirulo` (uid=1001, ventas, id=jpirulo) y `ana pirulo` (uid=1002, compras, id=apirulo). Para los grupos seleccionamos la OU `groups`, hacemos `create new child` y seleccionamos `Posix Group`. Creamos los dos grupos indicados pero deberemos modificar el `gid` ya que los asigna por defecto a partir de 1.000 y nosotros los queremos diferentes. Repetimos la operación en `users` seleccionando `User Account`. Aquí nos pedirá los datos de los usuarios (nombre, apellido, grupo, contraseñas, *home directory*, *shell*, etc.) y luego deberemos cambiar el `uid`, ya que los asigna por defecto. Luego podemos observar nuevamente el contenido ejecutando `slapcat` los nuevos datos generados. Phpldapadmin se puede configurar con el archivo */usr/share/phpldapadmin/config/config.php* pero en la mayoría de los casos la configuración por defecto ya es operativa (en algunos casos puede ser necesario adaptar la línea $servers->setValue('server','base',array('dc=nteum,dc=org')); para adecuar a los datos de nuestro servidor).

Para poder configurar un cliente tenemos que instalar los paquetes `slapd` y `ldap-utils` con `apt-get install slapd ldap-utils`, que nos solicitará una serie de información: URI del servidor (ldap://192.168.1.33), el DC (dc=nteum,dc=org), la versión (3), el usuario administrativo (cn=admin, dc=nteum,dc=org) y el `passwd`, nos informará que hagamos el cambio manual de nsswitch (ok), permitir a PAM cambiar los `password` locales (Yes), `enforces login` (No), `admin acount suffix` (cd=admin,dc=nteum, dc=org), y finalmente el `passwd` nuevamente.

A continuación deberemos modificar el archivo */etc/nsswitch.conf* con:

```
passwd: compat ldap
group: compat ldap
shadow: compat ldap
...
netgroup: ldap
```

Por último deberemos modificar el archivo */etc/pam.d/common-password* (quitando de la línea el 'use_authtok')

```
password [success=1 user_unknown=ignore default=die] pam_ldap.so
try_first_pass
```

y en */etc/pam.d/common-session* agregar al final (para crear el home directory automáticamente):

`session optional pam_mkhomedir.so skel=/etc/skel umask=077`.

Luego deberemos reiniciar la máquina (`shutdown -r now`) y ya nos podremos conectar con los usuarios que hemos creado (jpirulo o apirulo) desde la interfaz gráfica.

1.13. Servicios de archivos (NFS, *Network File System*)

El sistema NFS permite a un servidor exportar un sistema de archivo para que pueda ser utilizado de forma interactiva desde un cliente. El servicio se compone básicamente de un servidor (básicamente representado por `nfsd*` y un cliente (representado por `rpc.mountd`) que permiten compartir un sistema de archivo (o parte de él) a través de la red. En la última versión de NFSv4 se incluyen una serie de *daemons* más como *idmapd*, *statd*, además de uan serie de módulos para las nuevas funcionalidades de esta versión. En Debian, instalad `apt-get install nfs-common` (será necesario el paquete *rpcbin* que como se comentó anteriormente es el nuevo *portmap* y generalmente ya está instalado) para el cliente, mientras que el servidor necesita `apt-get install nfs-kernel-server`. El servidor (en Debian) se gestiona a través del *script* `/etc/init.d/nfs-kernel-server` o simplemente con `service nfs-kernel-server start|stop`. El servidor utiliza un archivo (`/etc/exports`) para gestionar el acceso remoto a los sistemas de archivo y el control de los mismos. Sobre el cliente (u otro usuario a través de `sudo`), el root puede montar el sistema remoto a través del comando:

```
mount -t nfs Ipserver:directorio-remoto directorio_local
```

y a partir de este momento, el directorio-remoto se verá dentro de directorio local (este debe existir antes de ejecutar el `mount`). Esta tarea en el cliente se puede automatizar utilizando el archivo de *mount* automático (`/etc/fstab`) incluyendo una línea; por ejemplo:

`remix.world:/home /home nfs defaults 0 0`.

Esta sentencia indica que se montará el directorio `/home` del *host* `remix.world` en el directorio local `/home`. Además, este sistema de archivo se montará con los parámetros por defecto (ver `man mount` apartado `mount options for ntfs` y man nfs para opciones específicas para NFSv4). Los últimos dos ceros indican que el sistema de archivos no debe ser `dumped` y que no se activará el `fsck` sobre él. El archivo `/etc/exports` sirve de ACL (lista de control de acceso) de los sistemas de archivo que pueden ser exportados a los clientes. Cada línea contiene un sistema de ficheros (*filesystem*) por exportar seguido de los clientes que lo pueden montar, separados por espacios en blanco. A cada

cliente se le puede asociar un conjunto de opciones para modificar el comportamiento (consultad *man exports* para ver un lista detallada de las opciones). Un ejemplo de esto podría ser:

```
# Ejemplo de /etc/exports
/           master(rw) trusty(rw,no_root_squash)
/projects   proj*.local.domain(rw)
/usr        .local.domain(ro) @trusted(rw)
/pub        (ro,insecure,all_squash)
/home       195.12.32.2(rw,no_root_squash) www.first.com(ro)
/user       195.12.32.2/24(ro,insecure)
/home       192.168.1.0/24(rw,sync,fsid=0,no_root_squash,no_subtree_check)
```

La primera línea exporta el sistema de archivos entero (`/`) a `master` y `trusty` en modo lectura/escritura. Además, para `trusty` no hay *uid squashing* (el root del cliente accederá como root a los archivos root del servidor, es decir, los dos root son equivalentes a pesar de ser de máquinas diferentes y se utiliza para máquinas sin disco). La segunda y tercera líneas muestran ejemplos de '*' y de *netgroups* (indicados por `@`). La cuarta línea exporta el directorio `/pub` a cualquier máquina del mundo, solo de lectura, permite el acceso de clientes NFS que no utilizan un puerto reservado para el NFS (opción `insecure`) y todo se ejecuta bajo el usuario *nobody* (opción `all squash`). La quinta línea especifica un cliente por su IP y en la sexta, se especifica lo mismo pero con máscara de red (/24) y con opciones entre paréntesis sin espacio de separación. Solo puede haber espacios entre los clientes habilitados. La última línea exporta el directorio */home* a todas las máquinas de la red 192.168.0.* en modo syncronizado, de lectura/escritura, con acceso del root remoto, fsid es la identificación de sistema de archivo y no_subtree_check indica que no se hará la verificación de la ruta/archivo en una petición sobre el servidor.

Dos comandos útiles para trabajar con el nfs son el `exportfs` (muestra y nos permite actualizar las modificaciones que se hayan realizado sobre */etc/exports*) y `nfsiostat/nfsstat` que nos permitirá obtener estadísticas de funcionamiento sobre NFS y observar su funcionamiento.

1.14. Servidor de wiki

Un (o una) **wiki** (del hawaiano *wiki wiki*, "rápido") es un sitio web colaborativo que puede ser editado por varios usuarios que pueden crear, editar, borrar o modificar el contenido de una página web, de forma interactiva, fácil y rápida; dichas facilidades hacen de una wiki una herramienta eficaz para la escritura colaborativa. La tecnología wiki permite que páginas web alojadas en un servidor público (las páginas wiki) sean escritas de forma colaborativa a través de un navegador, utilizando una notación sencilla para dar formato, crear enlaces, etc. y conservando un historial de cambios que permite recuperar de manera sencilla cualquier estado anterior de la página. Cuando alguien edita

una página wiki, sus cambios aparecen inmediatamente en la web, sin pasar por ningún tipo de revisión previa. Wiki también se puede referir a una colección de páginas hipertexto, que cualquier persona puede visitar y editar (definición de Wikipedia). Debian tiene su wiki en http://wiki.debian.org/ o también Apache en http://wiki.apache.org/general/ y ambas están basadas en **MoinMoin**. MoinMoin es una *Python WikiClone* que permite inicializar rápidamente su propia wiki y solo se necesitan un servidor de web y el lenguaje Python instalado. En la web de MoinMoin se encuentran las instrucciones detalladas para instalar MoinMoin, pero hay dos formas principales de hacerlo: instalación rápida e instalación de servidor.

Enlace de interés

Para saber más sobre MoinMoin podéis visitar su página web en: http://moinmo.in.
En concreto encontraréis las instrucciones detalladas para instalar MoinMoin en: http://master19.moinmo.in/InstallDocs.

1.14.1. Instalación rápida

1) Descargar el paquete desde http://moinmo.in/MoinMoinDownload que será, por ejemplo, para la versión 1.9 `moin-1.9.7.tar.gz`. Si se quiere verificar la integridad del paquete, se puede hacer `md5sum moin-x.x.x.tar.gz` y verificar que coincidan el *hash* generado con el que existe en la página de descarga.

2) Desempaquetar MoinMoin `tar xvzf moin-x.x.x.tar.gz`. Esto creará un directorio `moin-x.x.x` en el directorio actual con los archivos en su interior.

3) Dado que MoinMoin está escrita en Python, es necesario utilizar el intérprete de Python:

```
cd moin-x.x.x; python wikiserver.py
```

Esta orden mostrará por pantalla los mensajes de ejecución del servidor. Entre esta información se mostrará la dirección IP sobre la cual está corriendo el servidor, que podrá ser algo como `http://127.0.0.1:8080`. Esta opción usa un servidor web interno, será accesible desde `http://localhost:8080/` y funcionará hasta que se presione `Ctrl-C` en el terminal.

1.14.2. Instalación de servidor

MoinMoin es una aplicación WSGI (*Web Server Gateway Interface*), por lo tanto, el mejor entorno para ejecutar Moin Moin es uno que permita WSGI como, por ejemplo, Apache con mod_wsgi. En Debian podemos instalar el módulo instalando `apt-get install libapache2-mod-wsgi`.

Enlace de interés

Las instrucciones para instalar WSGI para Apache y configurar MoinMoin en este caso se pueden encontrar en la siguiente dirección: http://moinmo.in/HowTo/ApacheWithModWSGI.

Instalación de MoinMoin

Para instalar MoinMoin descargar la última versión y descompactar el archivo (p. ej., `tar xvzf moin-1.9.7.tar.gz`) y luego hacer una `cd moin-1.9.7/` y a continuación ejecutar:

```
python setup.py install --force –record=install.log --prefix='/usr/local'
```

Para hacer un test simple:

```
cd /usr/local/share/moin/server
python test.wsgi
```

En el navegador introducir como URL localhost:8000 y veremos la página de test de WSGI.

Copiar la configuración: `cd /usr/local/share/moin`

```
cp server/moin.wsgi .
cp config/wikiconfig.py .
```

Agregar un archivo en */etc/apache2/conf.d/moin.conf* con el siguiente contenido:

MoinMoin WSGI configuration
you will invoke your moin wiki at the root url, like http://servername/FrontPage:
WSGIScriptAlias / /usr/local/share/moin/moin.wsgi
create some wsgi daemons - use these parameters for a simple setup
WSGIDaemonProcess moin user=www-data group=www-data processes=5 threads=10
maximum-requests=1000 umask=0007
use the daemons we defined above to process requests!
WSGIProcessGroup moin

Modificar el archivo */usr/local/share/moin/moin.wsgi* agregando al final del párrafo a2: sys.path.insert(0, '/usr/local/share/moin')

Modificar los permisos de los directorios/páginas: `cd /usr/local/share; chown –R www-data:www-data moin; chmod –R ug+rwX moin; chmod –R o-rwx moin`

Verificar que tengamos un site por defecto en apache (sino hacer `a2ensite default`) y reiniciar apache (`service apache2 restart`).

Si nos conectamos a la URL localhost tendremos la página inicial de MoinMoin. Para configurar el nombre de la wiki y el usuario administrador podemos editar el archivo */usr/local/share/moin/wikiconfig.py*, quitamos el comentario de `page_front_page = u"FrontPage"` e indicamos el nombre del administrador, por ejemplo, `superuser = [u"WikiAdmin", ]` reiniciando apache nuevamente. Para configurar el lenguaje, debemos entrar como administrador (WikiAdmin) (si no tenemos un usuario seleccionamos *login*, seleccionamos 'you can create one now' y lo creamos -debe coincidir con el que introducimos como superuser-). Luego podremos configurar el idioma desde `http://localhost/LanguageSetup?action=language_setup` y a partir de esta acción ya podremos comenzar a crear nuestra primera wiki (podéis acceder a información adicional en http://moinmo.in/HowTo y particularmente en http://moinmo.in/HowTo/UbuntuQuick).

Para configurar múltiples wikis, primero debéis copiar `config/wikifarm/*` de la distribución en el directorio `moin/config/`. Luego se deben seguir las

instrucciones anteriores para cada una de las wikis de la colección (*farm*) teniendo en cuenta que:

1) es necesario tener `data_dir` y `data_underlay_dir` separados para cada wiki,

2) si buscáis que compartan alguna configuración, entonces esta debe estar en `farmconfig.py` y las específicas deben estar en `mywiki.py`.

1.15. Gestión de copias de respaldo (*backups*)

Las copias de seguridad o copia de respaldo (*backup*) se refiere a una copia de los datos originales que se realiza con el fin de disponer de un medio de recuperarlos en caso de pérdida total o parcial debido a fallos en los dispositivos físicos, borrados por accidente, ataques informáticos, infectados por virus u otras causas que hacen que la información no existe o no es la deseada. El proceso de copia de seguridad se complementa con un proceso de restauración de los datos (*restore*) que puede ser total o parcial/selectivo, que permite devolver el sistema informático al punto en en cual fueron los datos almacenados. Esto puede significar la pérdida de información entre el momento que se realiza la copia de seguridad y el momento que se detecta que los datos no existen o están corrompidos por lo cual la política de planificación de copias de seguridad debe ser una de las actividades importantes en todo administrador de sistemas.

Se debe tener en cuenta que la pérdida de datos es habitual (y no por ello sin consecuencia y en algunos casos fatales) ya que de acuerdo a estadísticas recientes más del 60 % de los usuarios de Internet declaran haber sufrido una seria pérdida de datos en alguna ocasión y según un estudio de la Universidad de Texas, solo el 6 % de empresas con pérdida catastrófica de datos sobrevivirá, frente a un 43 % que nunca reabrirá su negocio y un 51 % que tendrá que cerrar en un plazo de 2 años. Para reafirmar más aún la necesidad de copias de seguridad, aquellos sistemas que contengan datos de carácter personal y estén sujetos a la legislación del país (p. ej., en España la LOPD -Ley Orgánica de Protección de Datos-) entre una de las obligaciones que deben cumplir las empresas/instituciones/individuos, es tener copias de seguridad para preservar los datos que tienen almacenados y que están sujetos a esta normativa.

1.15.1. Programas habituales de copias de respaldo

Existen diversas opciones para hacer copias de respaldo con diferentes objetivos, prestaciones e interfaces en todas las distribuciones GNU/Linux (p. ej., backintime, bacula, backup2l, backuppc, bup, chiark, dejadup, dirvish, flexbackup, lucky, rdiff, vbackup entre otras). Una de las más potentes es **Bacula***

*http://www.bacula.org

(http://blog.bacula.org/), que es una colección de herramientas para realizar copias de seguridad en una red. Bacula se basa en una arquitectura cliente/servidor que resulta muy eficaz y fácil de manejar, ya que presenta un conjunto muy amplio de características y es eficiente tanto para un conjunto de ordenadores personales como para grandes instalaciones. El paquete está formado por diferentes componentes, entre los más importantes se pueden encontrar:

- **Bacula-director**, *daemon* que gestiona la lógica de los procesos de *backup*;
- **Bacula-storage**, *daemon* encargado de manejar los dispositivos de almacenamiento;
- **Bacula-file**, *daemon* por medio del cual Bacula obtiene los ficheros que necesita para hacer el respaldo y que se deberán instalar en las máquinas fuente de los ficheros a respaldar, y
- **Bacula-console**, que permite interactuar con el servicio de *backup*.

Bacula soporta discos duros, cintas, DVD, USB y también diferentes bases de datos (MySQL, PostgreSQL y SQLite) pero como contrapartida, es necesario disponer de todos los paquetes instalados y su instalación y puesta a punto pueden resultar complejas.

Otro paquete interesante es **BackupPC***, que permite hacer copias de seguridad de disco a disco con una interfaz basada en la web. El servidor se ejecuta en cualquier sistema Gnu/Linux y admite diferentes protocolos para que los clientes puedan escoger la forma de conectarse al servidor. Este programa no es adecuado como sistema de copia de seguridad de imágenes de disco o particiones ya que no soporta copias de seguridad a nivel de bloque de disco; sin embargo, es muy simple de configurar y la posible intrusión sobre la red de ordenadores en la cual se desea hacer el respaldo es mínima. Este servidor incorpora un cliente *Server Message Block* (SMB) que se puede utilizar para hacer copia de seguridad de recursos compartidos de red de equipos que ejecutan Windows.

***http://backuppc.sourceforge.net/info.html**

Su instalación es sumamente sencilla haciendo, en Debian, `apt-get install backuppc`. Nos indicará que seleccionemos el servidor web (apache2), y nos indicará el usuario y *passwd* que ha creado, no obstante este *passwd*/usuario se pueden cambiar con `htpasswd /etc/backuppc/htpasswd backuppc`. Luego en nuestro navegador ponemos como URL http://localhost/backuppc y con el usuario/*passwd* accederemos a la página principal de la aplicación donde nos dará el estado del servidor y las opciones.

Hay diversas formas de configurar los clientes para hacer las copias de respaldo y dependerá del método para hacerlo y del sistema operativo. Para ello consultar el apartado de documentación como se configurará cada una de las transferencias (http://backuppc.sourceforge.net/faq/BackupPC.html).

En el caso de un sistema (remoto) Gnu/Linux, desde la interfaz de administración editar la configuración del mismo (*host* y *xfer*) y confirmar que se ha

definido como método `rsync` y el directorio directorio a realizar la copia. Como los respaldos se realizan a través de `rsync` en combinación con `ssh` es necesario que el usuario *backuppc* del servidor pueda acceder como root sin clave a la máquina remota. Para ello adoptar la entidad del usuario *backuppc* (`su - backuppc`) y generar las llaves (sin *passwd*) `ssh-keygen -t dsa` y luego utilizar el comando `ssh-copy-id root@cliente` para copiar la llave. Verificar que se puede acceder al usuario root del cliente a través de ssh y sin *passwd*. A partir de este punto, solo bastará seleccionar al equipo remoto a realizar el *backup* desde la interfaz de administración e iniciar una primera copia seleccionando el botón Comenzar copia de seguridad completa.

En sistemas Windows, la forma más simple de realizar *backups* es a través del protocolo SMB por lo cual se deberá sobre el sistema Windows ingresar como administrador y configurar el sistema para compartir carpetas que se deseen hacer la copia de seguridad o bien el disco duro completo (por ejemplo C:) definiendo un usuario/*passwd* para compartir este recurso. Desde la interfaz de administración editar la configuración del host remoto con Windows a respaldar (por su IP por ejemplo), el usuario y el método smb (no olvidar de hacer Save después de haber modificado estos datos). Defina en la pestaña *Xfer* el nombre del recurso compartido a respaldar en el equipo remoto y el nombre del usuario y clave de acceso de recurso compartido del equipo Windows remoto. A partir de este punto, solo bastará seleccionar al equipo remoto a respaldar desde la interfaz de administración e iniciar un primer respaldo seleccionando el botón Comenzar copia de seguridad completa.

Con Backuppc se puede definir la frecuencia de los respaldos totales y respaldos incrementales. De modo predeterminado el valor para los respaldos totales es cada 7 días y para los incrementales es cada día. Se recomienda utilizar un valor ligeramente inferior a los días. Por ejemplo, 6.97 en lugar de 7 días y 0.97 en lugar de 1 día para mejorar la granularidad del respaldo (recordar siempre de hacer *Save* después de modificar cada opción).

1.15.2. rdiff-backup y rdiff-backups-fs

Para administradores o usuarios avanzados existe la opción de `rediff-backup` que es un comando para la copia de seguridad de un directorio a otro y que también puede ser a través de una red. El directorio de destino poseerá una copia del directorio fuente pero también información adicional (diffs) para gestionar mejor las copias incrementales (aún de archivos antiguos). La aplicación también preserva subdirectorios, enlaces no simbólicos (hard links), archivos dev, permisos, propiedad (uid/gid), fechas de modificación, atributos extendidos y ACL y puede operar de forma eficiente a través de un *pipe* a *rsync* por ejemplo o utilizar rdiff-backup + ssh para realizar backups incrementales en lugar remoto transmitiendo solo las diferencias. También es habitual (y muy simple) tener un proveedor de servicios (Gdrive, Dropbox, etc) con un directorio sobre el ordenador local que será sincronizado sobre el *cloud* del

proveedor y hacer la copia rdiff-backup sobre este directorio para que luego se transmita al *cloud* del proveedor. La forma habitual de trabajo (después de instalado el comando) es muy simple `rdiff-backup fuente destino` y en el caso remoto /algun/dir-local a /algun/dir-remoto sobre la máquina hostname.org será

```
rdiff-backup /algun/dir-local hostname.org::/algun/dir-remoto
```

pero puede ser a la inversa

```
rdiff-backup user@hostname.org::/remote-dir local-dir
```

y también podrían ser sobre dos máquinas remotas

```
rdiff-backup -v5 -print-statistics user1@host1::/source-dir user2@host2::/dest-dir
```

Para recuperar un directorio local es simplemente a través de la copia y si es remoto (más ejemplos en http://www.nongnu.org/rdiff-backup/examples.html)

```
rdiff-backup -r now hostname.org::/remote-dir/file local-dir/file
```

Uno de los problemas de recuperar la información previamente guardada es que acceder a los archivos de copia de seguridad más reciente es fácil (solo se debe introducir el directorio de copia de seguridad) pero es complicado si deseamos acceder y navegar a través de las versiones anteriores. `rdiff-backup` permite acceder a ellos por un conjunto de instrucciones, que requieren un conocimiento preciso de los nombres de archivo y los tiempos de copia de seguridad y que puede ser complicado de recordar o seleccionar. `rdiff-backup-fs` (https://code.google.com/p/rdiff-backup-fs/) permite crear un sistema de archivo en espacio de usuario basado en la información de rdiff. Para ello se utiliza FUSE (*Filesystem in userspac*e - http://fuse.sourceforge.net/) que permite que cualquier usuario pueda montar el sistema de archivo guardado por rdiff y navegar por cada incremento y copia realizada.

Una alternativa para las copias de resguardo de un sistema remoto por ssh es utilizar `sshfs` (http://fuse.sourceforge.net/sshfs.html) que permite acceso en el espacio de usuario a un directorio remoto mostrándolo localmente por lo cual luego aplicando rdiff se puede hacer copias incrementales de este recurso remoto.

1.16. *Public Key Infrastructure* (PKI)

Por PKI (*Public Key Infrastructure*) se entiende un conjunto de hardware y software, políticas y procedimientos de seguridad que permiten la ejecución con

garantías de operaciones como el cifrado, la firma digital o el no repudio de transacciones electrónicas. El término PKI se utiliza para referirse tanto a la autoridad de certificación y al resto de componentes si bien a veces, de forma errónea, se utiliza para referirse al uso de algoritmos de clave pública. En una operación que se utilice PKI intervienen, además de quien inicia la acción y su destinatario un conjunto de servicios que dan validez a la operación y garantizan que los certificados implicados son válidos. Entre ellas podemos contar con la autoridad de certificación, la autoridad de registro y el sistema de sellado de tiempo. La infraestructura PKI utiliza procedimientos basados en operaciones criptográficas de clave pública, con algoritmos de cifrado bien conocidos, accesibles y seguros y es por ello que la seguridad está fuertemente ligada a la privacidad de la clave privada y la políticas de seguridad aplicadas para protegerla. Los principales usos de la PKI son entre otros: autenticación de usuarios y sistemas (*login*), identificación, cifrado, firma digital, comunicaciones seguras, garantía de no repudio.[21][22][24]

Como se vio anteriormente (para apache+SSL) la generación de certificados digitales es extremadamente necesario dentro de las necesidades habituales de los administradores y usuarios de los sistemas de información, por ejemplo, para acceder a una página SSL o para firmar un correo electrónico. Estos certificados se pueden obtener de las entidades certificadoras privadas como StartComm (https://www.startssl.com/) o Verisign (http://www.verisign.es/) que tienen algunos productos gratuitos (por ejemplo para firmar correos) pero, en su mayoría, los productos tiene un costo elevado. También podemos recurrir a utilizar GNUPG (https://www.gnupg.org/) que es *Open-Source* y para determinados fines es correcto pero no para otros o bien considerar entidades de certificación institucionales, por ejemplo en Cataluña IdCat (http://idcat.cat/) que nos permitirán obtener certificados de ciudadano (gratuitos) para firmado, encriptación, autentificación, no repudio pero no para servidores (Idcat si que puede expedir otro tipo de certificados pero solo es para la administración pública y universidades del país). En el presente apartado instalaremos y configuraremos una entidad certificadora raíz (y sub-entidades CA para los diferentes dominios de control de esta CA) basada en la aplicación TinyCA (si bien existen otros paquetes como OpenCA -https://pki.openca.org/- que son más potentes y escalables pero son más complejos en su configuración) que se adapta muy bien para pequeñas y medianas instituciones/empresas. Una entidad certificadora es la base de la infraestructura PKI y emite certificados para dar garantías de autenticidad de una comunicación, un sitio o una información. Cuando instalamos Apache hemos auto-firmado el certificado digital (es decir nosotros hemos hecho todos los pasos: petición, generación y firma) que codifican la comunicación pero ello no da garantía si no se verifica la firma del certificado auto-firmado. Para solucionar este problema, y sin recurrir a un proveedor público/privado, crearemos nuestra propia estructura de CA y distribuiremos por canales seguros a nuestros usuarios/clientes el certificado personal/de servidores y el certificado raíz de la CA para que los instalen en sus navegadores (como ya están los de StartComm, Verisign, Catcert/Idcat u otras entidades de certificación). De esta forma que cuando un sitio web, por

ejemplo, le presente al navegador un certificado digital para codificar la comunicación SSL, el navegador reconozca el sitio por el certificado raíz que tiene instalado y confíe en él (y no salga la típica ventana de advertencia de sitio inseguro) y manteniendo la privacidad de las comunicaciones. La utilización de estos certificados creado por esta CA puede ser varios: para firmar/encriptar un mail, para validar nuestros servidores SSL o para configurar la VPN entre otros.

La práctica habitual es tener una CA y crear Sub-CA (una por cada dominio) para que el sistema sea escalable y por lo cual la CA firmará la creación de nuevas Sub-CA y luego estas (su responsable) tendrán capacidad para firmar sus certificados y todas tendrán el mismo certificado raíz de la RootCA. (guía muy detallada en [25]) Una vez instalada (`apt-get install tinyca`), ejecutamos `tinyca2` y nos presentará la pantalla principal y indicaciones para crear una nueva CA que será la rootCA. Completamos la pantalla con los datos, por ejemplo, *Name=ROOTCA-nteum.org, Data for CA=ROOTCA-nteum.org, SP, passwd, BCN, BCN, NTEUM, NTEUM, adminp@sysdw.nteum.org, 7300, 4096, SHA-1* para todos los campos respectivamente. Cuando le damos a OK, aparecerá una nueva pantalla para configurar la rootCA. Seleccionar *"Certificate Signing/CRL Signing"*, "*critical*" y en Netscape *Certificate Type="SSL CA, S/MIME CA, Object Signing CA*.", si se desea poner una URL para revocar los certificados (lo cual puede ser una buena idea) se pueden rellenar los campos correspondientes, luego finalmente OK. Con ello se crearán los certificados y aparecerán la pantalla principal de tinyCA con 4 tabuladores donde indica *CA* (información sobre la CA activa), *Certificates* (muestra los certificados creados por la CA) *Keys* (muestra las llaves de los certificados) y *Requests* (peticiones de certificados que esperan ser firmados por la CA). Por encima aparecen una serie de iconos para acceder a funciones directas pero Tinyca2 tiene un error y no muestra los nombres de los iconos.

El siguiente paso es crear una nueva sub-CA y para hacerlo verificar que estamos en el tab de la CA y haciendo click en el tercer icono desde la derecha que nos mostrará una ventana que como subtítulo tiene "*Create a new Sub CA*". Deberemos introducir el *passwd* que pusimos en la rootCA, darle un nombre (subca-sysdw.nteum.org), *Data-for-CA* (sysdw.nteum.org) y el resto de datos como hicimos anteriormente (el *passwd* no debe ser necesariamente el mismo de la rootCA) y cuando hacemos OK nos saldrá la ventana principal pero con la Sub-CA seleccionada, si queremos volver a la CA deberemos ir al menu *File->Open* y abrir la rootCA. Recordad que la Sub-CA será quien gestionará las peticiones y firmará los certificados para ese dominio por lo cual debemos seleccionar la adecuada en cada momento. La root-CA solo la utilizaremos para crear/revocar nuevas sub-CA y para exportar el certificado raíz de la rootCA (necesario para enviarles a nuestros clientes para que se lo instalen en sus navegadores).

Para crear un certificado que permita certificar nuestro web-server cuando utiliza SSL (https://sysdw.nteum.org), con nuestra Sub-CA seleccionada vamos al

tab de *Request* y con el botón derecho seleccionamos "*New request*" en la cual se abrirá una ventana donde deberemos introducir la URL de nuestro servidor (sysdw.nteum.org) como *CommonName* y rellenar el resto de los datos. Una vez creado, lo seleccionamos y con el botón derecho indicamos "*Sign request*" y seleccionamos "*Server request*" que nos mostrará una ventana con el *password* de la CA y la validez. Este procedimiento es el que genera confianza ya que estamos validando la información aportada en la petición y firmando el certificado (los cuales ya podremos ver en los tabs correspondientes.

Ahora deberemos exportar los certificados (del web y la rootCA), la *key* y configurar Apache para que los incorpore. Lo primero será exportar el rootCA e introducirlo en Firefox/Iceweasel. Para ello en la ventana principal *File->OpenCA* seleccionamos nuestra RootCA y seleccionando el segundo icono de la derecha que corresponde a "*Export CA Certificate*" indicamos un nombre de archivo y el formato (PEM es el adecuado) y salvamos el certificado en nuestro sistema de archivo (p. ej., /tmp/ROOTCA-nteum.org.pem). Es importante tener en cuenta de hacer un `chmod 644 /tmp/ROOTCA-nteum.org.pem` ya que se salvará como 600 y por lo cual otros usuarios no lo podrán importar. Desde Firefox/Iceweasel seleccionamos *Preferences/Setting->Advanced->Certificates->View Certificates->Authorities->Import* y seleccionamos el archivo antes creado marcando todas las "*trust settings*" que nos presenta en la ventana siguiente (3). Luego podremos ver con el nombre de que le dimos a "*Organization*" el certificado correspondiente.

A continuación en TinyCA2 abrimos nuestra sub-CA, seleccionamos el tab *Certificates* y sobre el certificado seleccionamos el botón derecho e indicamos "*Export certificate*" dando un nombre de archivo, p. ej, sysdw.nteum.org-cert.pem, luego repetimos el proceso con la *key* en el *Key* tab y haciendo "*Export key*" salvándolo como p. ej., sysdw.nteum.org-key.pem. Es importante decidir si le ponemos *passwd* o no ya que si le ponemos cada vez que arranque el servidor nos solicitará el *passwd*. Sobre el tab CA deberemos exportar ahora el "*certificate chain*" que es el primer icono desde la derecha y salvarlo como sysdw.nteum.org-chain.pem. Moveremos estos tres archivos al directorio */etc/ssl/private/* y pasaremos a configurar Apache modificando el archivo que configure nuestro sitio SSL, por ejemplo nosotros hemos utilizado */etc/apache2/sites-available/default-ssl* en el cual hemos modificado (solo se muestra las líneas principales):

```
<IfModule mod_ssl.c>
<VirtualHost _default_:443>
    ServerName sysdw.nteum.org
...
    SSLEngine on
    SSLCertificateFile /etc/ssl/private/sysdw.nteum.org-cert.pem
    SSLCertificateKeyFile /etc/ssl/private/sysdw.nteum.org-key.pem
    SSLCertificateChainFile /etc/ssl/private/sysdw.nteum.org-chain.pem
...
</VirtualHost>
</IfModule>
```

Solo nos resta habilitar el sitio (*a2ensite default-ssl*), e reiniciar Apache (con la orden `service apache2 restart`) -nos pedirá el *passwd* si hemos lo hemos puesto en la *key*- y probar (en el navegador que tenemos instalado el certificado raíz de la rootCA) la URL https://sysdw.nteum.org que si todo está correcto cargará la página sin la típica ventana que el sitio no es seguro.[25]

Para crear certificados para una dirección de correo y distribuirlos a nuestros usuarios junto con el certificado de la rootCA podemos hacer en el tab *Certificates* de nuestra sub-CA, seleccionar "*New - Create Key and Certificate (Client)*" y entrar el nombre y la dirección de correo para la cual queremos validar así como el *passwd* para protegerlo hasta que llegue a su nuevo destinatario (y que luego el podrá/deberá cambiar). A continuación debemos exportarlo teniendo en cuenta de utilizar el formato PKCD#12 que incluye el certificado y la llave. Después de enviarle al usuario el archivo con el certificado más el de la root-CA, el podrá agregarlo a su gestor de correo de forma similar a root-CA pero como *personal certificates*. Luego podrá enviar correo firmados digitalmente y el destinatario (que deberá tener instalado el certificado de la rootCA) podrá verificar la firma del correo.

Como apunte final con relación a la PKI, todos nuestros usuarios/clientes/visitantes de nuestros servicios deberán tener instalado el certificado de la root-CA en sus navegadores/clientes para así validar los sitios/servicios que están bajo nuestro dominio/sub-dominios ya que los navegadores/clientes de correo no incorporan estos certificados raíz por defecto. Si nuestro dominio/servicios lo requiere podemos gestionar con Mozilla la inclusión de nuestro certificado en http://www.mozilla.org/en-US/about/governance/policies/security-group/certs/ pero no es un trámite fácil y es necesario cumplir con una serie de requisitos ya que las garantías del sistema radica en la inclusión de estos certificados.

Actividades

1. Configurad un servidor DNS como caché y con un dominio propio.

2. Configurad un servidor/cliente NIS con dos máquinas exportando los directorios de usuario del servidor por NFS.

3. Configurad un servidor SSH para acceder desde otra máquina sin contraseña.

4. Configurad un servidor Apache+ SSL+ PHP+ MySQL+ PHPAdmin para visualizar las hojas personales de los usuarios.

5. Configurad un servidor Apache + un Reverse Proxy para acceder al servidor web desde una máquina externa a través del Proxy.

6. Cread y configurad un sistema de correo electrónico a través de Exim, Fetchmail, SpamAssassin y un servidor IMAP para recibir correos desde el exterior y poder leerlos desde una máquina remota con el cliente Mozilla (Thunderbird).

7. Instalad la Wiki MoinMoin y cread un conjunto de páginas para verificar su funcionamiento.

8. Instalad el servidor de *backups* BackupPC y generad una copia de respaldo desde una máquina Windows y otra desde una máquina Linux. Escoged el método de comunicación con los clientes y justificar la respuesta.

9. Configurad una CA con tiny CA y generar/probar los certificados para una página web con SSL y para enviar correos firmados y verificarlos desde otra cuenta.

Bibliografía

Todas las URLs han sido visitadas por última vez en Junio de 2014.

[1] **Debian.org**. *Debian Home*.
<http://www.debian.org>

[2] *Server-World*.
<hhttp://www.server-world.info/en/note?os=Debian_7.0>

[3] **Langfeldt, N.** *DNS HOWTO*.
<http://tldp.org/HOWTO/DNS-HOWTO.html>

[4] **IETF**. Repositorio de Request For Comment desarrollados por Internet Engineering Task Force (IETF) en el Network Information Center (NIC). <http://www.ietf.org/rfc.html>

[5] **Instituto de Tecnologías Educativas**. *Redes de área local: Aplicaciones y Servicios Linux*.
<http://www.ite.educacion.es/formacion/materiales/85/cd/linux/indice.htm>

[6] **Trenholme, S.** *MaraDNS - A small open-source DNS server*.
<http://maradns.samiam.org/>

[7] **DNSMasq** *DNS forwarder and DHCP server*.
<https://wiki.debian.org/HowTo/dnsmasq>

[8] *Comparison of FTP client software*
<http://en.wikipedia.org/wiki/Comparison_of_FTP_client_software>

[9] *Servicios de red: Postfix, Apache, NFS, Samba, Squid, LDAP*
<http://debian-handbook.info/browse/es-ES/stable/network-services.html>

[10] *NIS HOWTO*.
<http://www.linux-nis.org/>

[11] **Kukuk, T.** *The Linux NIS(YP)/NYS/NIS+ HOWTO -obs:EOL-*.
<http://tldp.org/HOWTO/NIS-HOWTO/verification.html>

[12] *Exim*.
<http://www.exim.org/docs.html>

[13] *ProcMail*.
<http://www.debian-administration.org/articles/242>

[14] **Apache HTTP** *Server Version 2.2*.
<http://httpd.apache.org/docs/2.2/>

[15] *Apache2 + WebDav*.
<http://www.debian-administration.org/articles/285>

[16] **Squid** *Proxy Server*.
<http://www.squid-cache.org/ >

[17] **Squid** *Configuration Examples*.
<http://wiki.squid-cache.org/ConfigExamples >

[18] **Kiracofe, D.** *Transparent Proxy with Linux and Squid mini-HOWTO -obs:EOL pero interesante en conceptos-*.
<http://tldp.org/HOWTO/TransparentProxy.html#toc1>

[19] **Pinheiro Malère, L. E.** *Ldap. The Linux Documentation Project*.
<http://tldp.org/HOWTO/LDAP-HOWTO/>

[20] *Proftpd*.
<http://www.debian-administration.org/articles/228>

[21] **PKI** *Public-key cryptography*.
<http://en.wikipedia.org/wiki/Public_key_cryptography >

[22] *SSL/TLS Strong Encryption: An Introduction*.
<http://httpd.apache.org/docs/2.2/ssl/ssl_intro.html#cryptographictech>

[23] *Transparent Multi-hop SSH*.
<http://sshmenu.sourceforge.net/articles/transparent-mulithop.html>

[24] **Christof Paar, Jan Pelzl** *Introduction to Public-Key Cryptography*.
<http://wiki.crypto.rub.de/Buch/movies.php >

[25] **Magnus Runesson** (2007). *Create your own CA with TinyCA2*.
<http://theworldofapenguin.blogspot.com.es/2007/06/create-your-own-ca-with-tinyca2-part-1.html>

[26] **Greg Ferro** *Fast Introduction to SOCKS Proxy*.
<http://etherealmind.com/fast-introduction-to-socks-proxy/ >

[27] *Proxy SOCKS*.
<http://en.flossmanuals.net/bypassing-es/proxis-socks/ >

[28] **Mourani, G.** (2001). *Securing and Optimizing Linux: The Ultimate Solution*. Open Network Architecture, Inc.
<http://www.tldp.org/LDP/solrhe/Securing-Optimizing-Linux-The-Ultimate-Solution-v2.0.pdf>

[29] *Samba*.
<http://www.samba.org/ >

[30] *Wiki - Samba*.
<https://wiki.samba.org >

[31] *RSAT. Remote Server Administration Tools on a Windows workstation*.
<https://wiki.samba.org/index.php/Installing_RSAT_on_Windows_for_AD_Management>

[32] **M. López, C. Alonso***Samba 4: Controlador Active Directory*.
<http://waytoit.wordpress.com/2013/05/12/samba-4-controlador-active-directory-parte-1-de-3/ >

[33] **M. Rushing***Compiling Samba 4 on Debian Wheezy - Active Directory Domain Controllers*..
<http://mark.orbum.net/2014/02/22/compiling-samba-4-on-debian-wheezy-active-directory-domain-controllers-ho/ >

[34] *Guía Samba4 como Controlador de Dominio y Directorio Activo*.
<http://fraterneo.wordpress.com/2013/08/19/guia-samba4-como-controlador-de-dominio-y-directorio-activo-actualizacion/ >