Configuración de Servicios de Red en GNU/Linux

Margarita Manterola y Maximiliano Curia

Versión Beta - Octubre 2003

Índice

1. X

Para empezar, veremos un servicio que utilizamos todos los días dentro del entorno GNU/Linux: el servidor X. El servidor X es un servicio muy poderoso, que nos permite utilizar terminales gráficas remotamente, ejecutar aplicaciones en una máquina y que se muestren en otra y varias cosas más.

Muchas veces, cuando se instala una distibución, el servicio de X está configurado para que no se puedan utilizar conexiones TCP (es decir que solamente están permitidas las conexiones de tipo **UNIX**).

Las conexiones de tipo UNIX se utilizan entre los diversos procesos que se están ejecutando en un mismo servidor. Al ejecutar `netstat -nl`, las conexiones de tipo UNIX son las últimas en aparecer; en la última columna se listan los archivos utilizados por la conexión. Estos archivos no son archivos comunes, sino que son de tipo *socket*. A través de esos archivos se comunican los procesos cuando utilizan estas conexiones de tipo UNIX.

1.1. Ejecutando aplicaciones en otras computadoras

Para permitir que otras computadoras puedan utilizar el servicio X de un servidor es necesario verificar que el servicio **no** se esté iniciando con la opción `-nolisten tcp`.

Si se está utilizando el *kdm*, *gdm* o *xdm*, será necesario modificar el archivo de configuración correspondiente para indicar que el servicio de X se inicie sin esa opción.

Una vez que el X se ha iniciado correctamente, desde una consola gráfica se puede utilizar el comando `xhost` para agregar o quitar permisos a determinada maquina para que pueda o no utilizar nuestra pantalla. Esto se hace con la siguiente sintaxis.

```
xhost +host
xhost -host
```

Por otro lado, en la máquina remota donde se quiera ejecutar la aplicación, será necesario configurar la variable de entorno `$DISPLAY`.

```
export DISPLAY=157.92.49.211:0
xmessage Hola!!
```

Luego de ejecutar estas líneas, en la computadora quet tenga la dirección IP `157.92.49.211` se verá un mensaje provisto por la aplicación *xmessage* en la pantalla 0.

Se pueden hacer pruebas con otras aplicaciones, como *xclock*, que muestra un reloj en modo gráfico, *xpenguin* que muestra un pingüino, o *xpenguins* que muestra una familia de pingüinos que caminan por la pantalla.

Es necesario indicar que es la pantalla 0, porque puede haber más de una pantalla en una misma máquina, como se explica a continuación.

1.2. Utilizando más entornos, con Xdmcp

De una manera similar, en lugar de ejecutar solamente las aplicaciones en forma remotamente, es posible obtener una pantalla de inicio de sesión.

Es decir, al servidor X que se está ejecutando en una máquina, se le puede indicar que reciba todas las aplicaciones a partir de otro servidor. De maneras que el procesamiento se realiza en el servidor y la otra máquina pasa a ser una terminal que simplemente muestra la pantalla y recibe la entrada de mouse y teclado.

Sobre estos principios se basa el desarrollo de *Thin Clients*, es decir, *Clientes Delgados*, que son computadoras sin disco rígido, que utilizan un servidor X para interactuar con el usuario y todo el procesamiento se hace en el servidor.

Para todo esto se utiliza el protocolo **xdmcp**: *X Display Manager Control Protocol*.

En el servidor que va a realizar el procesamiento, tiene que estar habilitado el servicio de xdmcp. Para esto, al igual que antes, es necesario modificar la configuración del programa de inicio de sesión que se esté utilizando. Ya sea el *xdm*, el *gdm* o el *kdm*.

Para poder iniciar el X, desde la máquina en la que solamente se va a mostrar la pantalla, es necesario utilizar el comando **X** con algunos parámetros especiales, para que realice consultas xdmcp a la red, como `-broadcast` que pregunta a toda la red quiénes responden el servicio xdmcp o `-query host` que trata de conectarse al xdmcp que este en ese ip.

Por ejemplo, si el servidor que va a atender las consultas tiene la dirección IP `157.92.49.102`, el siguiente comando abriría una nueva sesión de X, pero que se estaría ejecutando en ése servidor.

```
X :1 -query 157.92.49.102
```

2. Servidor de SSH

El servicio de ssh es prestado por el sshd, es decir el *daemon* de ssh.

El archivo de configuración, normalmente es `/etc/ssh/sshd_config`. En general, la configuración predeterminada cumple con los requerimientos que se pueden llegar a tener de este servicio.

Se puede seleccionar qué usuarios pueden ingresar y qué usuarios no pueden ingresar al sistema mediante este servicio, utilizando las siguientes opciones.

```
AllowUsers usuario1 usuario2
PermitRootLogin no
```

La opción `AllowUsers lista de usuarios` especifica qué usuarios tienen permiso para ingresar al sistema a través de ssh, cualquier otro usuario del sistema **no** va a poder acceder utilizando este servicio. Esto puede ser útil si se tienen varios usuarios con contraseñas nulas, o muy tontas y que no les interesa utilizar ssh o cuando el administrador quiere poder acceder de manera remota a la máquina, pero no quiere que los demás puedan ingresar.

También existen: `DenyUsers`, `AllowGroups`, `DenyGroups`, `AllowHosts`, `DenyHosts`.

La opción `PermitRootLogin no` es fundamental en cualquier ambiente, para que el usuario *root* no pueda ingresar directamente en forma remota. No es una buena idea ingresar directamente como administrador, ya que como política de seguridad siempre hay que tratar de utilizar el admiinistrador lo menos posible. Por otro lado, como el usuario *root* existe en todos los sistemas UNIX, a un posible atacante le alcanza con encontrar la contraseña del administrador para poder entrar a la computadora, si esta opción no está activada; en cambio de esta manera, debe además conocer un usuario del sistema y su contraseña para entrar y conseguir de alguna manera la contraseña del administrador.

Otra opción útil, es la que permite utilizar el protocolo X a través de ssh:

```
X11Forwarding yes
```

Cuando esta opción está habilitada, un cliente que se conecte utilizando la opción `-x` del cliente de ssh, generará un *DISPLAY* virtual en el servidor, creando un *túnel* entre ambas máquinas. Es decir que se pueden ejecutar aplicaciones gráficas en el servidor, que se muestran en el cliente, pero la información se transmite en forma segura, a través del ssh.

Por ejemplo, si el servidor *serrucho* se ha configurado para que acepte conexiones gráficas, se podría realizar la siguiente prueba.

```
user@boxitracio:~$ ssh -X serrucho
(...)
user@amadeus:~$ echo $DISPLAY
localhost:200.0
user@amadeus:~$ xclock
```

En este caso, el reloj gráfico se vería en la máquina original (*boxitracio*), aunque se estaría ejecutando en *serrucho* y los datos se estarían transmitiendo en forma segura a través del protocolo ssh.

Otra opción, utilizada para habilitar el servicio de sftp.

```
Subsystem sftp /usr/lib/sftp-server
```

2.1. Cliente de SSH - Utilización de claves

Cuando uno ingresa a un sistema en forma remota, muchas veces tiene que escribir su contraseña, y esto no es muy seguro. Uno puede equivocarse y escribirla en otra ventana o alguien puede estar mirando mientras la está escribiendo. Además es difícil utilizar una clave realmente complicada, ya que de alguna manera hay que recordarla, y no es una buena idea escribirla en un papel.

El cliente de ssh posee una característica especial que permite utilizar un método de autenticación alternativo. Este método consiste en utilizar una clave pública y una clave privada. Estas claves suelen tener un tamaño de 4096 bits, que lleva varios años de procesamiento para poder descifrar.

En primer lugar es necesario generar las claves correspondientes a un usuario de una máquina.

```
ssh-keygen -t dsa
```

Esto nos generará dos archivos `id_dsa` (clave privada) y `id_dsa.pub` (clave pública), generalmente almacenados en el directorio `.ssh` del home del usuario. La clave privada no se debe distribuir. La clave pública es la que se distribuye a los demás, pero para poder descifrar lo que esta cifrado con la clave pública se necesita la privada.

Una vez que están generados estos archivos, se puede agregar la clave pública al archivo `.ssh/authorized_keys` a toda computadora a la que se quiera ingresar sin necesidad de que pida la contraseña.

3. inetd

inetd es el super servidor de los sistemas unix. Se trata de un *demonio* que escucha en múltiples puertos, esperando conexiones de distintos clientes y, una vez establecida la conexión, inicia un programa para que se encargue de prestar el servicio correspondiente.

Se utiliza para disminuir la cantidad de *demonios* sin uso ejecutándose en el servidor, además provee un sencillo soporte de redes a programas que normalmente no lo tendrían.

Otra característica es que, internamente, contiene varios servicios triviales.

En sistemas unix, la configuración normalmente se encuentra en `/etc/inetd.conf` y su sintaxis es:

servicio tipo protocolo wait/nowait usuario server parámetros

Donde:

servicio es uno de los nombres que en `/etc/services` están asociados a un puerto y protocolo.

tipo puede ser: *stream*, *dgram*, *raw*, *rdm* o *seqpaquet*. Para las conexiones **tcp** normalmente se utiliza el stream y para las conexiones udp el dgram.

protocolo se refiere a un nombre asociado en `/etc/protocols` con un número que lo identifica. Se trata del protocolo que se va a utilizar en esa conexión para el intercambio de datos.

wait/nowait (sólo tiene efecto con el tipo dgram) especifica si cuando se inicia el servicio el inetd tiene que esperar a que el servicio termine para volver a escuchar en ese puerto o no. Opcionalmente se puede pasar una segunda opción agregándole *.max*, siendo max el número máximo de veces que se puede iniciar un server en un minuto (por omisión es 40).

usuario se refiere al usuario con que se ejecuta el servicio y opcionalmente el grupo (agregándole *.grupo*).

server el programa a iniciar una vez establecida la conexión.

parámetros los parámetros para el programa especificado en *server*.

3.1. Ejemplo: agregando servicios con inetd

Se dijo antes que inetd provee manejo de redes a programas que normalmente no acceden a la red. Para demostrarlo, se utilizan a continuación dos programas bien simples, date y cat.

Será necesario agregar las siguientes líneas a **inetd.conf**.

```
prospero          stream  tcp    nowait  root    /bin/date date
irc               stream  tcp    nowait  root    /bin/cat cat
```

Es decir, se están utilizando los puertos de conexión de **prospero** y de **irc**, que en el archivo **/etc/services** están asociados al puerto 191 y 194 respectivamente. El último parámetro es el nombre del programa, ya que el inetd puede invocar un mismo comando con distintos nombres.

Para que estos cambios tengan efecto, será necesario reiniciar el inetd. Luego, con el comando telnet es posible conectarse los puertos 191 y 194 para usar los nuevos servicios.

En el caso del comando *cat* se hace evidente que la entrada y salida estándar de los servicios están asociadas a la conexión de red. Para estos comando es imprescindible utilizar una conexión TCP, con otro tipo de conexión no funcionaría.

3.2. tcp-wrappers

Algunos servicios utilizan como nombre de servicio el comando `/usr/bin/tcpd` y como primer parámetro la ruta completa de un servicio de red. El comando tcpd provee un control de acceso a los servicios de red, conocido como *tcp-wrappers*.

Para configurarlo se utilizan los archivos de configuración `/etc/hosts.allow` y `/etc/hosts.deny`, y a partir de ellos se define si se dará o no servicio a una determinada máquina. Es decir que a través de *tcp-wrappers* se puede hacer que un servicio sencillo se pueda limitar sólo a algunas máquinas.

El orden es:

- Si en hosts.allow se autoriza se ofrece servicio
- Si en hosts.deny se deniega no se ofrece el servicio
- Si no esta especificado se ofrece el servicio

La sintaxis de `hosts.allow` y `hosts.deny` sigue el siguiente esquema.

```
servicio: cliente [: shell_script]
ALL: ALL EXCEPT cliente
```

En primer lugar se utiliza el nombre del programa que se quiere configurar.

A continuación se ingresan los clientes, que pueden ser: el nombre de una máquina (si empieza con . es para todo el dominio), una dirección IP, un rango de direcciones IP (con una máscara de la forma 255.255.254.0) o una ruta a un archivo que contenga la lista de máquinas.

Se aceptan algunos comodines especiales como *ALL*, *LOCAL* o *EXCEPT* tanto en los campos de servicios como en los de clientes.

El script de shell que se incluye al final es opcional, y permite ejecutar un comando cuando se selecciona una determinada regla.

Para más información sobre estos archivos de configuración: `man 5 hosts_access`

3.3. Ejemplo: agregando control de acceso

En el ejemplo anterior, se agregaron dos servicios de red, utilizando la configuración de inetd. Para agregar control de acceso al servicio de **date**, será necesario modificar la línea de la siguiente manera.

```
prospero      stream  tcp   nowait  root  /usr/sbin/tcpd  /bin/date
```

Y, por otro lado, modificar el archivo **/etc/hosts.deny**:

```
date: localhost
```

De esta manera, se está negando el acceso al servicio *date* al servidor local. Si se ejecuta **telnet localhost 191**, habrá una pequeña pausa y luego se cerrará la conexión, sin haber mostrado la fecha.

3.4. Servicios Adicionales

En inetd.conf se suelen encontrar los servicios que no son de mayor importancia para el sistema, y que solamente se utilizan esporádicamente. A continuación una explicación somera de algunos de estos servicios.

3.4.1. talk

Se trata de un servicio que permite conversar con otros usuarios, ya sea dentro de una misma máquina (un usuario local con uno remoto, o dos usuarios remotos) o en distintas máquinas.

Por ejemplo, para hablar con el usuario pepe que se encuentra utilizando la máquina *anteojito* su puede utilizar `talk pepe@anteojito`. El usuario pepe verá en su pantalla una invitación a entablar una conversación con el usuario que lo está llamando, a la cual puede contestar escribiendo `talk juan@antifaz` (si es que el usuario juan desde la máquina antifaz lo está invitando).

Una vez establecida la conversación, pueden hablar mediante una cómoda ventana de chat.

3.4.2. finger

El servicio de *finger* permite ver la información personal de un usuario (nombre real, teléfono, email, etc). Puede tratarse de un usuario local de la máquina, o un usuario de otra computadora.

Utilizando `finger user@oaky` se podrá ver la información del usuario *user* en el servidor *oaky*.

Si existe el archivo `.nofinger` dentro del directorio personal de un usuario, el servicio de finger negará la existencia del usuario para cualquier pedido externo.

3.4.3. telnetd

Este servicio es uno de los más sencillos de configurar. Normalmente utiliza el super demonio *inetd* para iniciarse, opcionalmente puede iniciarse a mano con la opción `-debug`. Al ingresar nos muestra el contenido del archivo `/etc/issue.net`, que tiene la misma sintaxis que `/etc/issue`.

Actualmente existen servidores y clientes telnet que utilizan, transparentemente, un método de autenticación cifrado, pero el resto de la conexión, no. Es el caso del demonio *telnetd-ssl*.

3.4.4. tftp

tftp significa *Trivial FTP*. Se trata de un ftp muy sencillo, que no tiene autenticación. Se utiliza principalmente para máquinas que tienen que iniciar a través de la red, ya que es tan simple que el programa cliente puede entrar junto con otros pequeños programas dentro de la memoria de una placa de red (unos 16Kb).

3.4.5. ident

El ident es un servicio que provee información sobre una determinada conexión TCP/IP. Por omisión, informa qué usuario es el dueño de esa conexión.

4. Apache

Apache es un servidor web para GNU/Linux, que también se ha portado para otros sistemas operativos. Es un servidor web confiable, eficiente y muy flexible en cuanto a la configuración.

Fue desarrollado por un conjunto de empresas que necesitaban un servidor web y se unieron para hacer entre ellas lo que luego se convirtió en el proyecto Apache.

Hoy en día, dentro del proyecto Apache se incluyen una gran cantidad de proyectos de mayor o menor complejidad, siempre relacionados de algún modo con servicios web.

4.1. Configuración básica

Una vez que se ha instalado el servidor en un sistema GNU/Linux, según la distribución, el directorio raíz para las páginas web, puede encontrarse en distintos lugares del sistema. En Debian, el directorio que se utiliza es `/var/www`, en otras distribuciones puede ser `/home/httpd`, `/var/www/html`, etc.

El directorio raíz del sitio es el punto de partida para el contenido de `http://dominio`, donde `dominio` es el nombre de la máquina que se está configurando como servidor web (puede ser un dominio real, o simplemente el nombre local de la máquina).

Si se colocan archivos en formato HTML dentro del directorio raíz, el servidor los proveerá a quién los solicite. Por omisión, el archivo que el servidor entrega, si no se pide ninguno en particular (es decir, si se accede directamente a `http://dominio/`, es `index.html`.

También es posible proveer archivos en otros formatos, como PHP o Perl, si se le agregan a Apache los módulos para que soporte estos lenguajes.

El archivo principal de configuración de Apache es `/etc/apache/httpd.conf`, en ese mismo directorio (`/etc/apache`) se encuentran otros archivos adicionales. Según la distribución puede llamarse `/etc/httpd` en lugar de `/etc/apache`.

Usualmente el archivo de configuración que se instala por omisión contiene mucha documentación acerca de las opciones posibles, que puede ser útil a la hora de configurar el servidor.

4.2. Directorios public_html

La instalación típica de Apache, le asigna a cada usuario su propio espacio web. Todo lo que se encuentre dentro del directorio `public_html` dentro del directorio *home* del usuario, será publicado por el servidor web con el nombre de `http://dominio/~usuario/`.

En estos directorios se pueden poner archivos HTML, y también PHP o Perl, siempre y cuando el servidor esté configurado para entender estos lenguajes.

4.3. Algunas configuraciones generales

En el archivo de configuración, las siguientes líneas pueden ser cambiadas y tienen los siguientes significados:

ServerAdmin el email del administrador del servidor

DocumentRoot el directorio raíz del servidor (según se explicó anteriormente).

Port el puerto donde el servidor está escuchando. Por omisión es el puerto 80, pero se puede cambiar.

Además de las configuraciones generales, se configuran distintas opciones para los diversos directorios.

```
<Directory />
  Options SymLinksIfOwnerMatch
  AllowOverride None
</Directory>
```

Esta configuración se refiere al acceso a cualquier archivo al disco rígido. Es una configuración muy restrictiva. Con configuraciones específicas para determinado directorio podemos elegir opciones menos restrictivas.

```
<Directory /var/www/>
  Options Indexes Includes FollowSymLinks MultiViews
  AllowOverride None
  Order allow,deny
  Allow from all
</Directory>
```

Esta es la configuración del directorio raíz del sitio. Las opciones indicadas en la línea **Options** significan:

Indexes Al acceder a un directorio, si no existe alguno de los archivos que se muestran por omisión (definidos por la directiva **DirectoryIndex**), se hace un listado de los archivos que hay en el directorio.

Includes Para el uso de ServerSideIncludes, esto es, una página que incluye a otra utilizando un código especial que el apache interpreta para agregar esa página antes de mostrarla.

FollowSymLinks Le indica al servidor que siga los symlinks (esto se aplica únicamente dentro del árbol del directorio establecido).

SymLinksIfOwnerMatch Le indica al servidor que solamente siga los symlinks si el dueño del symlink coincide con el dueño del archivo destino (esto se utiliza dentro de la /, para proteger los archivos del sistema).

MultiViews Permite enviar diferente contenido al cliente según la configuración de su navegador (preferencias de lenguaje, formatos predeterminados, etc).

Las líneas de **Order** indican el orden en que se tienen que asignar o no los permisos para acceder a las páginas. En este caso el orden es **allow,deny**. Y la línea de **Allow** indica que se le dé permisos a todo el mundo para acceder.

```
<IfModule mod_dir.c>
    DirectoryIndex index.html index.htm index.shtml index.cgi index.php
</IfModule>
```

Indica cuáles son los archivos que el servidor busca para mostrar, cuando no se especificó ningún archivo. Se buscan en el mismo orden en que aparecen. Es decir, en este caso, si existe un **index.html** y un **index.php**, se va a mostrar el `.html` por omisión.

```
<IfModule mod_userdir.c>
    UserDir public_html
</IfModule>
```

Indica cuál es el directorio a utilizar dentro del directorio de cada usuario.

4.4. Dominios virtuales

Un servidor web con una sola dirección IP puede proveer páginas web de más de un dominio. A esta capacidad se la denomina *Dominios Virtuales*.

En primer lugar, será necesario editar el archivo de configuración y colocar una línea que contenga: **NameVirtualHost direccion-IP**.

Donde la dirección IP puede ser:

- La dirección IP real de la máquina, si se trata de una máquina con IP pública.
- La dirección IP dentro de una red interna. Por ejemplo: 192.168.5.1).
- La dirección local de la máquina, 127.0.0.1, con lo cual el servicio web será solamente local
- Un nombre de dominio que resuelva a una determinada dirección IP (no es recomendable).
- Un *, que utiliza la o las direcciones IP que estén configuradas en el servidor.

Además, será necesario agregar una configuración para cada uno de los dominios virtuales que se quieran configurar.

```
<VirtualHost *>
ServerName dominio-virtual.com
ServerAlias www.dominio-virtual.com
ServerAdmin webmaster@host.dominio-virtual.com
DocumentRoot /var/www/host.dominio-virtual.com
ErrorLog /var/log/apache/host.dominio-virtual.com-error.log
TransferLog /var/log/apache/host.dominio-virtual.com-access.log
</VirtualHost>
```

Lo que va al lado de **VirtualHost** es una dirección IP, con el mismo criterio explicado anteriormente (una dirección IP, un nombre de dominio -opción no recomendada-, o un *). Pueden incluirse varias entradas con la misma dirección (o *).

La opción **ServerName** es la que indica el dominio virtual que se está agregando a la configuración.

La opción **ServerAlias** provee uno o más nombres alternativos para el mismo dominio virtual.

La opción **ServerAdmin** es el email correspondiente al administrador del dominio virtual.

La opción **DocumentRoot** indica donde se encuentra el directorio raíz de ese dominio virtual.

Las opciones **ErrorLog** y **TransferLog**, indican donde se almacenan los registros de Log del dominio virtual.

Es posible agregar una gran variedad de opciones adicionales para cada dominio virtual.

Otras configuraciones posibles, por VirtualHost o generales.

```
Alias /icons/ /usr/share/apache/icons/
Alias /doc/ /usr/share/doc/
ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
```

La directiva **Alias** indica que cuando se introduzca un determinado directorio en una dirección web, representa en realidad a otro directorio dentro del disco rígido.

La directiva **ScriptAlias**, hace lo mismo, definiendo a la vez que en ese directorio se encuentran archivos ejecutables cuya salida estará en formato HTML.

4.5. Tipos mime

Los tipos *mime* son los tipos que identifican a los archivos por su contenido. Muchas veces es responsabilidad de los distintos navegadores el encargarse de mostrar un contenido que es de determinado tipo mime de una determinada manera. En algunos casos es el servidor web el que tiene que presentar la información de distinta manera según de qué tipo sea el contenido.

Para estos casos existe un archivo, generalmente llamado `mime.types`, que incluye un largo listado de los diversos tipos existentes y la extensión asociada. De manera que cuando el servidor va a enviar un archivo al cliente, antes de enviarlo envía una línea de la forma `image/png`, que le indica de qué tipo es.

Es necesario tener habilitado el módulo **mod_mime** para poder acceder a las siguientes directivas, con una línea como la siguiente al comienzo de la configuración del servidor.

```
LoadModule mime_module /usr/lib/apache/1.3/mod_mime.so
```

Con la directiva `TypesConfig /etc/mime.types`, se indica en qué archivo se encuentra el listado de tipos mime.

Con la directiva `DefaultType text/plain`, se indica cuál es el tipo que se debe asignar si no es posible determinar correctamente el tipo del archivo. El valor que se asigne con esta directiva puede variar según los directorios. Por ejemplo, si se tiene un directorio que contiene mayormente archivos de tipo JPEG, se puede incluir la directiva `DefaultType image/jpeg`, para el caso en que no sea posible determinar el tipo de alguno de los archivos.

4.6. Restricción de acceso

El servidor web Apache nos permite una restricción muy selectiva del acceso que se quiere dar al contenido. Por ejemplo, a continuación se reproduce la configuración para los directorios de los usuarios (**public_html**), que permite que se utilicen los métodos GET y POST para la transmisión de información, pero deniegan todos los métodos peligrosos.

```
  <Directory /home/*/public_html>
    AllowOverride FileInfo AuthConfig Limit
    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
    <Limit GET POST OPTIONS PROPFIND>
        Order allow,deny
        Allow from all
    </Limit>
    <Limit PUT DELETE PATCH PROPPATCH MKCOL COPY MOVE LOCK UNLOCK>
        Order deny,allow
        Deny from all
    </Limit>
</Directory>
```

Es posible, además, tener configuraciones particulares por archivos, tanto dentro de un directorio en particular, como sin importar en qué directorio se encuentren.

```
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>
```

En este caso, se impide el acceso a cualquier archivo que comience con '.ht', esto es para que no se pueda acceder a los archivos .htaccess, y .htpasswd, que contienen la información de restricción específica dentro de los directorios.

4.7. Cambios dentro de los directorios

La directiva `AccessFileName .htaccess` dentro de la configuración general indica cuál es el archivo que se utiliza para la configuración específica dentro de cada directorio.

Se trata de un archivo oculto (ya que comienza con .), que permite cambiar el comportamiento de determinados factores dentro del directorio, desde cuál va a ser el archivo a mostrar de forma predeterminada hasta requerir un usuario y una clave para poder acceder al contenido.

La utilidad principal consiste en que no es necesario modificar la configuración de todo el servidor ni reiniciar el apache si se quieren hacer pequeños cambios personalizados.

Para poder configurar estos valores particulares es necesario que el directorio en el que se quiera colocar el `.htaccess` tenga la directiva `AllowOverride` acompañada de alguno del valor que corresponda al área que se quiera reconfigurar.

Por ejemplo, `AllowOverride AuthConfig` permite cambiar la configuración relacionada con los permisos para acceder al directorio, mientras que `AllowOverride FileInfo Indexes` permite modificar la información relacionada con el manejo de archivos y la configuración del archivo índice del directorio.

Por otro lado, se puede incluir `AllowOverride All` o `AllowOverride None`, que permiten sobreescribir todas o ninguna de las directivas respectivamente

Por ejemplo, para que el contenido de directorio esté protegido con un usuario y clave se utiliza una configuración como la siguiente.

```
AuthType Basic
AuthName "Nombre de Dominio"
AuthUserFile  .htpasswd
AuthGroupFile .htgroups
Require  {valid-user|{user|group} cadena}
```

La directiva **Require** indica de qué manera se va a efectuar la validación. Si se utiliza `Require user usuarios`, solamente los usuarios listados pueden acceder al directorio. Por otro lado, si se utiliza `Require group grupos`, sólo los usuarios que pertenezcan a los grupos listados pueden acceder al directorio. Finalmente, se se utiliza `Require valid-user`, todos los usuarios válidos pueden acceder.

Cabe aclarar que con este tipo de validación no se utilizan usuarios *reales* del sistema, sino los usuarios que se den de alta como se explica a continuación.

Con la directiva **AuthUserFile** se indica la ubicación del archivo que contiene los nombres de usuarios y sus respectivas contraseñas encriptadas. El nombre usual para este archivo es **.htpasswd**, aunque puede ser cualquier otro.

Para generar este archivo se utiliza el comando `htpasswd`, con la siguiente sintaxis.

```
htpasswd -c .htpasswd usuario1
(...)
htpasswd .htpasswd usuario2
```

En el primer caso se utiliza la opción `-c` para que cree el archivo. Una vez que ya está creado pueden seguir agregándose usuarios indefinidamente.

Cuando se utiliza seguridad por grupos, es necesario crear un archivo que contenga qué usuarios pertenecen a qué grupos. En el ejemplo anterior este archivo era el `.htgroups`. La sintaxis de este archivo es muy sencilla.

```
grupo: usuario vos yo
```

En general, siempre que sea posible es una buena idea que los archivos `.htpasswd` y `.htgroups` no estén ubicados en el mismo directorio que la información que se quiere proteger.

5. DHCPD

El servicio de DHCP (*Dynamic Host Configuration Protocol* es el que permite asignar determinados parámetros de configuración a una computadora sin necesidad de que estén colocados en forma estática dentro de esa máquina.

Entre los parámetros que se pueden configurar están: la dirección IP, la máscara de subred, los servidores de nombres (DNS), el gateway de la red, etc.

En una red física no debe haber más de un servidor DHCP. Ya que el cliente para recibir la configuración que le corresponde hace un pedido general a la red, y el servidor DHCP le contesta, enviando los parámetros de configuración correspondientes.

Por otro lado, un mismo servidor DHCP puede atender las consultas de diferentes redes físicas, asignando distintos valores a las máquinas según la red en la que se encuentren.

El protocolo de DHCP es un protocolo estándar, de manera que una máquina con GNU/Linux puede proveer este servicio a computadoras con otros sistemas operativos, y también ser cliente de este servicio aún si está provisto por otro sistema operativo.

A continuación un ejemplo de la configuración de un servidor de DHCP. Que se encuentra en el **/etc/dhcpd.conf**.

```
option domain-name "l11.fi.uba.ar";
option domain-name-servers dns.fi.uba.ar;

option subnet-mask 255.255.255.0;
default-lease-time 600;
max-lease-time 7200;

subnet 192.168.0.0 netmask 255.255.255.0 {
  range 192.168.0.51 192.168.0.100;
  option broadcast-address 192.168.0.255;
  option routers gateway.fi.uba.ar;
}

subnet 192.168.1.0 netmask 255.255.255.0 {
  range 192.168.1.11 192.168.1.200;
  option broadcast-address 192.168.1.255;
  option routers lug.fi.uba.ar;
}
```

En primer lugar se listan las opciones generales para todas las redes.

Donde **domain-name** es el nombre de dominio que se va a ingresar como la primera línea del archivo **/etc/resolv.conf**. Y **domain-name-servers** son los servidores de nombres (DNS) que se van a asignar, que también se almacenan en **/etc/resolv.conf**.

La opción **subnet-mask** es la máscara de subred que se va a asignar a cada uno de los clientes.

La opción **default-lease-time** indica cada cuánto tiene que renovar la información el cliente. Esto permite corroborar si todavía hay una computadora ocupando esa dirección IP, a la vez que permite cambiar determinada información en forma dinámica.

Por otro lado, la opción **max-lease-time** es el tiempo máximo permitido para la renovación antes de que la IP se considere liberada. Es decir, normalmente, aunque una máquina se reinicie, o se reinicie el cliente de DHCP, volverá a tomar la misma dirección que ya tenía asignada anteriormente.

A continuación se configuran los parámetros específicos para cada una de las subredes. Puede haber una o más, según sea necesario. La dirección y la máscara se utilizan para definir a qué red hacen referencia estos valores.

Dentro de la subred, el rango indica cuáles son las IPs que están disponibles para asignar. El resto de las direcciones IP no son asignadas mediante el servicio de DHCP, están reservadas para uso estático.

La opción **broadcast-address** indica cuál es la dirección de *broadcast* a utilizar dentro de esa red. Se trata de una dirección tal que cuando se le envían paquetes, todos esos paquetes se reenvían a toda la red.

La opción **routers** indica cuál es el servidor a utilizar como *gateway* de salida al resto de la red.

```
host flaco {
  hardware ethernet 00:D0:09:A5:95:4C;
  fixed-address 192.168.1.13;
  server-name "gordo";
  option root-path "/var/lib/diskless/default/root";
  filename "kernel-delgado";
}
```

Dentro o fuera de una subred se puede especificar una configuración para una máquina en particular. Para estos casos específicos se utiliza la opción **hardware address** que indica la identificación *única* (aunque no es verdaderamente única) de una placa de red, también conocida como *MAC address*.

A continuación se especifica una dirección en particular, de manera que siempre que se inicie esa máquina va a tener esa IP. Además, se incluye una configuración adicional que se utiliza para el arranque de clientes delgados; indica cuál es el servidor del cuál se tiene que bajar la imagen de arranque (**server-name**), cuál es la ruta para obtener el archivo (**option root-path**) y cuál es el archivo que debe bajar (**filename**).

6. BIND

La aplicación más utilizada como servidor de nombres (DNS) es el **bind**. La explicación que sigue es para el servicio de Bind 8.

En Debian GNU/Linux la configuración se encuentra en el directorio **/etc/bind**. Esta configuración está dividida en un archivo de configuración principal (**named.conf**), que define las zonas, y varios archivos adicionales, que configuran los registros para resolución de nombres y resolución inversa (de IPs a nombres) de cada una de las zonas.

En el archivo **/etc/bind/named.conf**, La sección **options** es la sección donde se configuran muchas de las funcionalidades adicionales del **bind**.

Para que un servidor de nombres resuelva direcciones IP (es decir, que traduzca nombres a direcciones), es necesario configurar alguna manera de llegar a los root servers, esto se define en **zone .** .

```
zone "." {
    type hint;
    file "/etc/bind/db.root";
};
```

Cuando el DNS está configurado de esta manera se suele llamar *cache DNS*, ya que va recordando las direcciones que resolvió recientemente.

Para la resolución de nombres a IPs se define una zona con el nombre del domino que vamos a configurar. Por ejemplo:

```
zone "gnuservers.com.ar" {
        notify no;
        type master;
        file "/etc/bind/db.gnuservers.com.ar";
};
```

Para poder configurar un domino público (accesible desde internet), primero es necesario tener un DNS registrado en nic (es decir, la organización que se encarga de regular y administrar los dominios del mundo), o en nic.ar (la de Argentina).

El **notify no** le dice al DNS server que no le diga a ningún otro DNS que él tiene este dominio configurado. El **type master** indica que los nombres dentro de ese dominio los va a resolver de forma autoritativa. Y por último **file** es el archivo donde se encuentra la configuración de ese dominio.

La configuración del dominio define un TTL (Tiempo de vida) un SOA (Comienzo de Autoridad) y la información de la máquina. Por ejemplo, una configuración posible podría ser:

```
$TTL    604800
@   IN  SOA gnuservers.com.ar. root.gnuservers.com.ar. (
                    1     ; Serial
               604800     ; Refresh
                86400     ; Retry
              2419200     ; Expire
               604800 )   ; Negative Cache TTL
;
@   IN  NS  ns1.gnuservers.com.ar.
@   IN  A   24.232.108.92
                        MX          mail
```

```
ns1                     CNAME   @
mail                    CNAME   @

www      IN          A       24.232.108.93
         MX          mail
```

En este archivo, cuando utilizamos la @ hacemos referencia a la zona, luego del SOA se especifica qué dominio se hace cargo de estos datos y a quién se puede contactar en caso de algún problema.

El **serial** es el primer número dentro del SOA. Y sirve para poder diferenciar distintas versiones de archivos. Se recomienda usar YYYYMMDDXX donde YYYY es el año, MM el numero del mes, DD el numero del día y XX el numero de modificaciones que le fuimos haciendo ese día.

Luego decimos la zona que NS (name server) usa, luego en que ip esta y quien procesa el mail.

En este caso ns1 y mail es un nombres canónico de la zona. De esta manera definimos ns1.zona, también podemos escribirlo como ns1.zona. (debemos agregar el ultimo punto para diferenciarlo de ns1.zona.zona .

Y por último le decimos que www.zona está en otro IP, también le especificamos un Mail Exchanger (**MX**).

Además de servirnos para traducir nombres a direcciones de ip, el DNS también puede ayudarnos a hacer lo contrario. Esto es útil para mostrarle información al usuario. En este caso la zona que debemos configurar tiene que ver con el rango de ips que usemos, pero con los octetos invertidos. Es decir, para la dirección 192.168.200.0/24 usamos 200.168.192.in-addr.arpa, el in-addr.arpa es una terminación especial que señala que es una zona de resolución inversa.

```
zone "200.168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/db.200.168.192";
};
```

Y en el archivo **db.200.168.192**.

```
$TTL    604800
@   IN  SOA freak.amadeus   root.amadeus. (
                  6     ; Serial
             604800     ; Refresh
              86400     ; Retry
            2419200     ; Expire
             604800 )   ; Negative Cache TTL
```

```
;
@   IN  NS  freak.amadeus.

1     IN  PTR erwin.amadeus.
10    IN  PTR amadeus.amadeus.
52    IN  PTR bicicleta.amadeus.
13    IN  PTR freak.amadeus.
```

7. NFS

/etc/exports Limitación del demonio nfs a nivel kernel de un solo export por filesystem(o de la misma jerarquía)

```
directorio      que-maquinas(opciones,opciones) maquinas(opciones)
```

```
mount host:directorio punto-de-montaje
```

Configuración en el fstab. En la nueva versión es obligatorio poner sync o async en el exports

8. NIS

NIS, significa Network Information Services(Servicios de información de Redes), y nos sirve para distribuir configuraciónes (como usuarios, grupos, passwords, hosts, etc) de un sistema a otro.

Es de gran utilidad, sobre todo en conjunto con NFS. Pero, nuevamente, no se puede considerar como un servicio seguro.

8.1. Configuraciones generales

En el archivo `/etc/defaultdomain` definimos el dominio nis que vamos a usar, no hace falta que sea nuestro dominio real, ni nada, solo que todas las maquinas tengan el mismo valor. Las mayúsculas y minúsculas son distintas.

En el archivo `/etc/nsswitch.conf` se encuentra la configuración de como va a obtener el sistema varias de sus configuraciones. Lo que especifica es donde las tiene que buscar, puede ser, en los archivos locales(files), en el nis, en el dns, etc.

```
passwd:          compat
group:           compat
shadow:          compat
```

```
hosts:          files dns
networks:       files

protocols:      db files
services:       db files
ethers:         db files
rpc:            db files

netgroup:       nis
```

Esta configuración utliza una busqueda especial, `compat` que nos permite tener una configuración local que modifique(localmente) configuraciones del nis, mediante el uso de lineas que comienzan con `+` o `-` en los archivos de configuración (/etc/passwd, /etc/shadow y /etc/group).

Agregando `+::::::` en `/etc/passwd` estamos incluyendo todos los usuarios que esten configurados en el servidor nis y no en nuestra maquina, si queremos cambiar alguno de los campos tan solo basta con escribirlo, por ejemplo `+maxy:::::/tmp:`, hace que /tmp sea el home del usuario maxy.

Tener en cuenta que en `/etc/group` es `+:::` y en `/etc/shados` es `+::::::::`.

(en Debian)En el servidor, el archivo `/etc/default/nis` se define en que modo iniciamos el servicio, con la variable `NISSERVER`, si la seteamos en `master` creamos un nuevo servidor nis.

Para generar las bases del servidor nis es necesario ejecutar `/usr/lib/yp/ypinit -m` en el servidor master.

Al iniciar el nis, normalmente lo que hace es buscar en las maquina locales para encontrar el servidor. En el caso que no se pueda acceder al servidor de esta manera se puede configurar en el archivo `/etc/yp.conf`. Bajo la directiva ypserver.

Una buena medida de seguridad es poner en el archivo `/etc/ypserv.securenets`.

9. SAMBA

El protocolo SMB (*Service Message Block*) fue creado en 1984 por IBM, y muy pronto adoptado por Microsoft, con el objetivo de compartir archivos, impresoras, puertos serie y *named pipes*. Con el paso del tiempo el protocolo siguió siendo desarrollado, tanto por Microsoft como por otras empresas.

Es el protocolo que utilizan las computadoras con Microsoft Windows para compartir archivos con otras computadoras en red. Utilizando GNU/Linux es posible comunicarse con otras computadoras utilizando este protocolo, el programa que implementa este protocolo se llama **Samba**.

Actualmente, hay una iniciativa de renombrar el protocolo SMB a CIFS (*Common Internet FileSystem*).

9.1. Instalación

Samba compone un número importante de aplicaciones distintas que pueden ser necesarias o no, según el modo en que se vaya a utilizar el sistema GNU/Linux.

En principio, existen dos *daemons*: **nmbd** y **smbd**, que son los que se utilizan para que otras computadoras tengan acceso a la nuestra, utilizando SMB. Un cliente **smbclient** que permite conectarse a otras computadoras de una manera similar a un ftp, una herramienta **smbfs** que permite montar y desmontar sistemas de archivos a través del protocolo SMB. Y algunas otras herramientas que son utilizadas tanto por los usuarios como por los programas.

En la página de Samba, **http://www.samba.org**, se puede bajar un archivo, **samba-latest.tar.gz**, que tiene todas estas herramientas. Por otro lado, en cada una de las distintas distribuciones de GNU/Linux, las herramientas para poder utilizar Samba pueden estar distribuidas en uno o más paquetes.

9.2. Conceptos Generales

En una red en la que se quiere utilizar el protocolo SMB, cada computadora debe tener un nombre que la identifique. Este nombre puede ser una dirección IP, un nombre de dominio, o un nombre del tipo NetBIOS (el tipo de nombre que le asigna Microsoft Windows a cada máquina).

Desde una línea de comandos de GNU/Linux, es posible ver los nombres de las máquinas que estén compartiendo archivos a través del protocolo SMB, utilizando el comando `nmblookup '*'` para una búsqueda común, o también `nmblookup -T '*'` para una búsqueda que consulte con un DNS.

Dentro del sistema GNU/Linux, es el demonio **nmbd** el que se utiliza para comunicarle el nombre de la computadora a los demás miembros de la red.

Además, cada máquina debe tener al menos un recurso compartido para que se la pueda utilizar. El demonio **smbd** es el que permite compartir estos recursos.

9.3. Montar un sistema de archivos con smbfs

Normalmente, con las herramientas para clientes de samba, cualquier usuario puede montar un recurso compartido de otra computadora dentro de la propia. Esto se realiza a través del sistema de archivos smbfs, utilizando los comandos **smbmount** y **smbumount**.

Las siguientes dos líneas son equivalentes:

```
mount -t smbfs -o guest //maquina/recurso punto-de-montaje
```

```
smbmount //maquina/recurso punto-de-montaje -o guest
```

9.3.1. Configuración en el fstab

Es posible configurar
//familia/c /mnt/familia/c smbfs gid=fat32,guest,dmask=775,fmask=664

9.4. Configuración del servidor

El archivo de configuración de un servidor samba se encuentra usualmente en el directorio **/etc/samba/smb.conf**. En este archivo será necesario configurar el nombre de la máquina y del *grupo de trabajo* en el que se encuentra, la forma de autenticación de los usuarios, los recursos compartidos, etc.

Está dividido en distintas secciones para cada parte de la configuración. La sección `[global]` es la que contiene toda la información del servidor, mientras que las otras secciones contienen la información de cada uno de los recursos compartidos, con el nombre que llevará el recurso como etiqueta de la sección.

9.4.1. Configuración básica

Dentro de la sección `[global]`, encontramos:

```
workgroup = NOMBRE_GRUPO
server string = Descripcion
```

Donde **workgroup** es el nombre del grupo de trabajo al que pertenece la computadora, y **server string** es la descripción de la computadora, pero no el nombre.

9.4.2. Seguridad

Una de las opciones que hay que considerar en el momento de configurar un servidor de samba es **security**. Se trata de la opción que define de qué forma se van a autenticar los usuarios con el servidor. Las opciones son cuatro: share, user, server y domain (que empezó a estar a partir de la versión 2.0).

En todos los casos hay un diálogo entre el cliente, que está queriendo acceder a los recursos compartidos y el servidor que tiene que decidir si le da acceso o no.

security = user En este tipo de seguridad, cada usuario debe autenticarse con un nombre de usuario y contraseña que sean válidos en el servidor GNU/Linux al que se está conectando. La autenticación se realiza antes de visualizar el listado de recursos compartidos.

security = share En este tipo de seguridad, las políticas de autenticación se definen en cada uno de los recursos (*share*) que se definan. Es posible listar los recursos sin haberse autenticado previamente.

security = server En este tipo de seguridad, la autenticación la maneja otro servidor, que se configura en la opción **password server**. Desde el punto de vista del cliente, es equivalente a la seguridad tipo **user**, ya que la autenticación se realiza antes de ver el listado de recursos compartidos, pero en lugar de validar los usuarios con los mismos usuarios del sistema, los puede validar con algún servidor de Microsoft Windows NT, o 2000.

Por otro lado, es importante tener en cuenta las siguientes líneas dentro de la configuración global.

```
guest account = nobody
invalid users = root
```

El usuario que se indica como **guest account**, es el usuario que el servicio de samba asumirá cuando tenga que acceder a archivos (tanto para crear como para leer). La opción de **invalid users** impide que algún usuario malintencionado pueda conectarse como un usuario determinado (en este caso **root**) para acceder archivos a los que no tiene acceso normalmente.

9.4.3. Recursos

En un sistema GNU/Linux es posible compartir directorios o impresoras. Cada sección que se agregue al archivo de configuración será el nombre de un nuevo recurso. A continuación, algunos ejemplos de recursos compartido.

```
[public]
   comment = Directorio Publico
   browseable = yes
   writable = yes
   public = yes
   path = /public
   create mask = 0700
   directory mask = 0700
```

Esta sección crea el recurso 'public', donde todos los usuarios pueden navegar y escribir (siempre que los permisos de los directorios se lo permitan) y que apunta al directorio `/public` dentro del disco rígido. Además, se indica la máscara de creación de archivos y directorios.

```
[personal]
   comment = Directorio personal
   browseable = yes
   read only = yes
   path = /home/user
   guest ok = yes
```

En este caso, se trata del directorio personal de un usuario, que puede ser navegado por los otros usuarios, pero no se les permite escribir (para esta configuración es lo mismo poner **read only = yes** o **writeable = no**. La última línea, indica que aún los usuarios que no se hayan autenticado pueden navegar por estos archivos, esta línea es equivalente a **public = yes**.

```
 [homes]
   comment = Directorios de los usuarios
   browseable = yes
   read only = no
   create mask = 0664
   directory mask = 0775
```

La sección '[homes]' es una sección especial que viene pre-definida por el servicio de samba. Cuando esta sección está presente dentro del archivo de configuración, todos los directorios personales de los usuarios son exportados dentro de este recurso, y es posible implementar un sistema que monte a cada usuario su directorio personal, de modo que puedan tener los mismos archivos personales sin importar en qué computadora se estén conectando.

```
[cdrom]
   comment = Samba server's CD-ROM
   writable = no
   locking = no
   path = /cdrom
   public = yes
   preexec = /bin/mount /cdrom
   postexec = /bin/umount /cdrom
```

Cuando el recurso compartido es un CD-ROM, es necesario montarlo y desmontarlo, por eso se utilizan los parámetros de **preexec** y **postexec** que se encargan de montar y desmontar el CD-ROM para que la utilización sea transparente a los usuarios. La opción de **locking = no** hace que no se realice un bloqueo de archivos, aún cuando el cliente lo solicite, ya que no tiene sentido bloquear los archivos de un CD-ROM.

9.4.4. Compartir Impresoras

Para poder compartir impresoras, será necesario configurar algunas líneas dentro de la sección '[global]'. A continuación, un ejemplo de un sistema utilizando **CUPS** como sistema de impresión.

```
printing = cups
printcap name = /etc/printcap.cups
load printers = yes
```

Un recurso compartido de impresión será de la forma:

```
[printers]
comment = Todas las impresoras
path = /usr/spool/samba
browseable = no
guest ok = no
writable = no
printable = yes
create mode = 0700
```

La sección '[printers]', al igual que '[homes]', es una sección especial del servicio de samba. Si se cuenta con una configuración de impresoras del tipo BSD, no es necesario explicitar cada una de las impresoras compartidas, sino que samba las detecta automáticamente.

El parámetro clave en esta configuración es **printable = yes**, ya que a los usuarios no se les permite escribir en este recurso, pero sí se les permite imprimir.

9.5. Herramientas para examinar la red

Es posible examinar las máquinas que se encuentran en la red, visualizándolas de forma similar a como se lo hace en Microsoft Windows con herramientas como **gnomba**, **komba**, etc.

Estas herramientas no son más que una interfaz gráfica a las herramientas explicadas anteriormente desde línea de comandos.

10. SMTP

Existen muchos servidores de SMTP que son muy utilizados en el mundo de UNIX y de GNU/Linux: *sendmail* es el más antiguo y probablemente el más complicado; *qmail* es uno de los más poderosos, aunque no es software totalmente libre; *postfix* y exim son alternativas un poco más sencillas de configurar y libres.

En este caso se explica la configuración de exim, aunque los conceptos generales son muy similares para todos los casos.

(**ESTO ESTA TODO SIN TERMINAR!!!**)

aliases usuario: destino,destino destino: usuario@otrodominio.com.ar *: usuario

Exim especifico lsearch y lsearch*

qualified domain /etc/email-addreses

Tener muchos dominios. Un alias por dominio.

un archivo de texto con los dominios uno por linea agregarlo a la lista de dominios con la ruta absoluta.

Agregar un director

```
virtual:
  driver = aliasfile
  domains = /etc/domainlist
  search_type = lsearch*
  file = /etc/aliases.d/${domain}
```

Y los aliases que haga falta. (uno por dominio).

Tener muchos dominios, con usuarios distintos.

Agregar un transporte y dos directors

```
virtual_localdelivery:
  driver = appendfile
  create_directory = true
  directory_mode = 700
```

```
  file = /var/spool/virtual/${domain}/${local_part}
  user = mail
  group = mail
  mode = 660

virtual_alias:
   driver = aliasfile
   file_transport = address_file
   pipe_transport = address_pipe
   domains = /etc/virtual/domains
   file = /etc/virtual/${domain}/aliases
   search_type = lsearch*
   user = mail
   qualify_preserve_domain

virtual_localuser:
   driver = aliasfile
   file_transport = address_file
   pipe_transport = address_pipe
   transport = virtual_localdelivery
   domains = /etc/virtual/domains
   file = /etc/virtual/$domain/passwd
   search_type = lsearch
   no_more
```

Autenticación:
El exim contra un proveedor de internet
Agregar

```
plain:
  driver = plaintext
  public_name = PLAIN
  client_send = "^usuario^pass"
cram-md5:
  driver = cram_md5
  public_name = CRAM-MD5
  client_name = usuario
  client_secret = pass
```

Es recomendable usar el segundo.
Además, se debe agregar modificar el remote_smtp (transport):

```
remote_smtp:
```

```
  driver = smtp
  authenticate_hosts = mail.dominios.infovia.com.ar
```

Los usuarios contra exim

```
plain:
  driver = plaintext
  public_name = PLAIN
  server_condition = "${if \
    crypteq{$2}{\
      ${extract{1}{:}{${lookup{$1}lsearch{/etc/exim/passwd}{$value}{*:*}}}}\
    }{1}{0}}"
  server_set_id = $1

login:
  driver = plaintext
  public_name = LOGIN
  server_prompts = "Username:: : Password::"
  server_condition = "${if \
    crypteq{$2}{\
      ${extract{1}{:}{${lookup{$1}lsearch{/etc/exim/passwd}{$value}{*:*}}}}\
    }{1}{0}}"
  server_set_id = $1
```

Previamente creando el /etc/exim/passwd con htpasswd.
Usuarios virtuales contra el exim.

```
login:
   driver = plaintext
   public_name = LOGIN
   server_prompts = "Username:: : Password::"
   server_condition = ${if and {\
                   {exists:/etc/virtual/${domain:$1}/passwd}\
                   {crypteq {$2}{\
                     ${lookup {${local_part:$1}} lsearch \
                       {/etc/virtual/${domain:$1}/passwd}{$value}fail}\
                   }}\
                   }{1}{0}}
   server_set_id = $1
```

Creando cada uno con los respectivos htpasswd.

11. Seguridad

La seguridad es un tema muy amplio y que necesita constante actualización en el tema. Primero veamos a que llamamos seguro y a que no. Una computadora esta segura cubierta de concreto, en el fondo del océano sin conexión a ninguna red. Una computadora es muy insegura si la dejamos en la calle, tiene múltiples servicios corriendo, no tiene comprobación de contraseñas o son muy simples, tiene un gran ancho de banda, esta conectada a múltiples redes, etc.

En cualquier caso intermedio siempre existe un riesgo. Por eso siempre que estemos considerando la seguridad que debemos tener debemos hacer una evaluación de riesgos. Para hacer ese análisis se tienen que evaluar distintos puntos. El acceso físico, acceso a datos, perdida de datos, modificación de datos, etc.

Y cada uno de estos puntos ser considerado en nivel del sistema.

Y normalmente lo que vamos a hacer es acotar caminos para que esto suceda mayormente cuando nosotros queremos. El acceso físico es la posibilidad de que alguien agarre la maquina y salga corriendo, o saque el disco rígido lo ponga en otro CPU y extraiga los datos, etc. Con acceso físico se puede hacer casi cualquier cosa y es importante tenerlo en mente. Y mas allá de las trabas que podamos poner en el equipo, para detener el acceso físico podemos considerar tener la maquina en un cuarto cerrado o personal de vigilancia verificando su correcto uso.

En cuanto al acceso a través de la red o de servicios. Podemos dejar accesibles solo los servicios que usamos, eso lo podemos conseguir con un firewall. Podemos "asegurarnos" que intermediarios no van a poder leer la información haciendo que viaje encriptada, pero si password del usuario pepe es pepe o nuestro sistema no pide contraseña, todo los esfuerzos serán vanos.

Por otro lado dejamos todo andando, firewall, sistema, vigilancia, cuatro alarmas y ya esta pusimos la foto de nuestro tamagochi en internet. No, no evaluamos bien.

Por otro lado tenemos que tener un buen sistema de backup funcionando.

11.1. Firewall

Vamos a ver un poco de como crear nuestro firewall basado en iptables.

(**Atención: a esto le faltan todas las explicaciones!**)

```
-N nueva cadena
-F elimina cadenas
-X borra cadenas vacías
```

```
-P policy
-L lista

-A agregar
-I insertar
-R reemplazar
-D borrar

Filtros
-s direccion fuente
-d direccion destino
direccion/mascara, 0/0
! direccion/mascara   NOT

-p protocolo
TCP UDP ICMP   /etc/protocols

-i interfaz fuente INPUT, FORWARD
-o interfaz destino FORWARD, OUTPUT
ppp+ cualquier ppp

-f fragmentos

-p y -m, extensiones

-p tcp --sport --dport --syn --tcp-flags
-p udp --sport --dport
-p icmp --icmp-type

-m mac
--mac-source

-m limit
--limit n/second n/minute n/hour n/day
--limit-burst cuantos

-m owner
--uid-owner userid
--gid-owner groupid
--pid-owner processid
--sid-owner sessionid

-m estado
```

--state
NEW
ESTABLISHED
RELATED
INVALID

Target
ACCEPT
DROP

cadenas (-N)

LOG
--log-level
--log-prefix

REJECT rechaza

RETURN vuelve

QUEUE se lo pasa a un programa (user level)

Todo lo que no esta explícitamente permitido esta prohibido.
(policys)

Cerrar todos los servicios no usados(incluso si esta
bloqueado por el firewall)

Route packet verification
echo 1 > /proc/sys/net/ipv4/conf/ppp0/rp_filter

-t nat
PREROUTING DNAT --to-destination
-o no existe

caso especial de DNAT REDIRECT --to-port

POSTROUTING SNAT --to-source
caso especial de SNAT MASQUERADE