Resumen

Postfix es un agente de transporte de correo electrónico (MTA) bastante reciente que es una alternativa más al legendario Sendmail. En su diseño han sido muy importantes factores como la seguridad, el rendimiento y la facilidad de administración y configuración. Migrar del sistema actual a uno nuevo es delicado por eso este manual ayudará a los administradores de sistemas a conseguirlo con el tiempo mínimo de parada del servicio.

Introducción

Postfix es un MTA que puede sustituir (piensan muchos) con ventaja a Sendmail. Es un producto de software excelente, y en este artículo daré algunos argumentos que espero lo demuestren. Wietse Venema es un programador respetado como pocos. Autor de programas tan conocidos como los TCP Wrappers, SATAN, The Coroner Toolkit, sus versiones de portmap y rpcbind.... es un experto en seguridad informática, lo cual, como poco, nos tranquiliza a los usuarios de Postfix cuando pensamos en los múltiples agujeros que ha tenido Sendmail a lo largo de su historia. Siendo ya más objetivos, algunas de las virtudes de Postfix son:

- Diseño modular (no es un único programa monolítico)
- La seguridad ha sido un condicionante desde el comienzo de su diseño.
- Lo mismo cabe decir del rendimiento (seguramente Sendmail no se diseñó pensando que algún día habría sitios necesitaran procesar cientos de miles o millones de mensajes al día).
- Soporte para las tecnologías más usadas hoy día: LDAP, Bases de datos (MySQL y PostgreSQL), autenticación mediante SASL, LMTP, etc.
- Estricto cumplimiento de los estándares de correo-e (hasta donde se puede sin dejar a media Internet, que no los cumple, sin poder usar el correo-e).
- Soporte muy bueno para dominios virtuales.
- Facilidad de configuración.
- Compatibilidad hacia/desde fuera con Sendmail (`.forward`, `aliases`, suplanta `mailq`, `newaliases`, `/usr/lib/sendmail` con versiones equivalentes).
- Abundante documentación, y de calidad.
- Fácil integración con antivirus.
- Uso sencillo de listas negras.
- Soporta de forma nativa el formato de buzones Maildir original de qmail.
- Tiene múltiples formas de obtener información de 'lo que está pasando'para resolver problemas o simplemente, para aprender.
- Se pueden lanzar varias instancias de Postfix en la misma máquina con distintas configuraciones, usando cada una distintas direcciones IP, distintos puertos, etc.

- Filtrado de cabeceras y cuerpos de mensajes por expresiones regulares.
- Utilidades para varias cosas, como gestionar las colas de mensajes.

Por último, pero no menos importante, hay que decir que el código fuente de Postfix (por supuesto de dominio público) es un ejemplo de diseño, claridad y documentación, lo cual facilita su mantenimiento (por su autor o, en el futuro, por otros) así como la incorporación de nuevas capacidades, corrección de errores, etc.

Características

Diseño modular

El sistema Postfix está compuesto de varios procesos que se comunican entre sí, aparte de varias utilidades que puede usar el administrador para influir en el sistema u obtener información de él. Este diseño, junto con el fichero master.cf que permite configurarlos tiene algunas ventajas:

- Cada proceso corre con los mínimos permisos necesarios para realizar su tarea.
- Es más sencillo localizar cuál está fallando.
- Se puede activar la emisión de más información de depuración de forma independiente para cada programa. Esto es realmente útil para resolver problemas.
- Se puede definir ciertos parámetros para cada uno de ellos, como el número máximo de procesos simultáneos de un tipo, etc.
- Se pueden activar y desactivar algunos de ellos. Por ejemplo en una máquina dial-up que sólo envía correo podemos desactivar el proceso servidor SMTPD.
- Se puede insertar procesos externos entre ciertas partes del sistema lo cual es muy útil para anti-virus, filtrados, etc.
- Podemos, por ejemplo, lanzar un servidor SMTPD adicional en otro puerto o sobre otra IP, con distintas opciones de configuración de acceso.
- También podemos correr varias instancias de Postfix, con las únicas limitaciones de que ambas no compartan el directorio de colas y que usen distintos valores para myhostname. Con un poco de imaginación podemos hacer realmente maravillas con todas estas opciones.

Otro aspecto en el que Postfix es modular es en el sistema de colas de mensajes. Se mantienen las siguientes colas:

maildrop

Es donde van los mensajes enviados localmente, mediante la versión de Postfix de `/usr/lib/sendmail`.

incoming

Aquí van los mensajes recibidos por SMTP (tras recibir cierto procesamiento) y los que han pasado por la cola maildrop.

active

Los que se está intentando enviar en un momento dado.

deferred

Los que se ha intentado y no se han podido enviar.

Aunque parezca mucha sobrecarga tanta cola, en realidad Postfix las procesa de forma realmente eficiente. Un inconveniente de tanta modularidad es que no tenemos ningún equivalente a `sendmail -v`, pero activando las opciones de depuración y viendo los ficheros de log, no se echa de menos.

Seguridad

Seguridad frente a ataques contra el servidor y también contra el uso inadecuado (spam, etc). En cuanto a la primera acepción, no conozco ningún problema serio de seguridad que se haya detectado aún en Postfix. Respecto al tema de spam, relay, etc, Postfix soporta directamente el uso de listas negras. Si se juega mucho con las opciones de restricciones de acceso, sí hay que tener cuidado con las que se ponen y en qué orden. Podemos acabar dejando puertas abiertas, o por el contrario tener un *bunker* inutilizable. Afortunadamente este tipo de opciones tiene nombres muy descriptivos y están bien documentadas, por lo que es fácil hacerlo bien. En éste manual se proporcionará más información sobre los controles de acceso en Postfix. Para terminar, se puede instalar Postfix de forma que corra en modo *chroot*, lo que proporciona aún más seguridad.

Rendimiento

El rendimiento de Postfix es realmente muy bueno. Por ejemplo, cuando se produce algún atasco de correo, tras resolverse los mensajes salen a una velocidad mucho mayor de la que observábamos cuando teníamos Sendmail.

Conceptos básicos

Os recomiendo encarecidamente no saltarse esta sección o tenerla al menos presente cuando utilice a lo largo del manual algún acrónimo.

MTA

MTA es el acrónimo de Mail Transfer Agent, es decir, Agente de Transporte de Correos. En otras palabras, es el servidor SMTP en sí, no el software que usa el usuario para manejar el correo.

`http://es.wikipedia.org/wiki/MTA`.

SMTP

He mencionado SMTP pero no he dicho aún que era. SMTP es acrónimo de Simple Mail transfer Protocol (Protocolo Simple de Transferencia de Correo electrónico). Es un protocolo orientado a red, sin ningún tipo de autenticación, que apareció en 1982.

`http://es.wikipedia.org/wiki/SMTP`

`http://en.wikipedia.org/wiki/SMTP`.

MUA

Acrónimo de Mail User Agent, es el cliente de correo electrónico.

`http://es.wikipedia.org/wiki/MUA`.

PIM

Un PIM es acrónimo de Personal Information Manager (Manejador de Información Personal) y es mucho más que un MUA. Sirve para apuntar tareas, contactos, enviar y recibir correo, etc. Un ejemplo es Evolution o Microsoft Office Outlook.

`http://en.wikipedia.org/wiki/Personal_information_manager`.

IP

Una IP es un número que identifica a un ordenador en una red.

`http://es.wikipedia.org/wiki/Dirección_IP`

POP3

Es un acrónimo de Post Office Protocol (Protocolo de Oficina de Correos) versión 3. Es un protocolo para recogida de correo del servidor desde el MUA del cliente. Por defecto un servidor POP escucha en el puerto 110 TCP. A diferencia de otros, no requiere estar conectado permanentemente a internet, sino que podemos descargarnos los correos cuando necesitemos, dentro de la franja horaria de nuestro ISP, por ejemplo.

`http://es.wikipedia.org/wiki/POP`

`http://en.wikipedia.org/wiki/Post_Office_Protocol`

ISP

Acrónimo de Internet Service Provider (Proveedor de Servicios de Internet). Son empresas como Telefónica, Wanadoo, Jazztel, etc.

`http://es.wikipedia.org/wiki/ISP`

IMAP

Es un acrónimo de Internet Message Access Protocol (Protocolo de red de acceso a mensajes electrónicos). Se suele usar para el correo web pues permite compartir agendas, carpetas de mensajes, etc. Es más moderno que POP por lo que da más juego.

`http://es.wikipedia.org/wiki/IMAP`

log

Cuando me refiera al log, me refriré al registro del MTA, que en Debian se encuentra en `/var/log/mail.log`.

FQDN

Es el acrónimo de Fully Qualified Domain Name. Un FQDN es un nombre entendible por personas que incluye el nombre de la computadora y el nombre de dominio asociado a la misma. Por ejemplo, dada la computadora llamada `mail` y el nombre de dominio `i-david.com`, el FQDN será `mail.i-david.com`

`http://en.wikipedia.org/wiki/FQDN`

Registros DNS

Hablaré de los 4 principales con los que trabajaremos. Host A, MX, CNAME y PTR.

Host A es un registro que apunta a una dirección IP. Por ejemplo google.es apunta a 212.25.58.42

El registro MX se usa para el correo. Significa Mail eXchange. Su utilidad es que tras un mismo dominio (i-david.com por ejemplo) coexistan varios servidores con igual o difernete prioridad. La prioridad se denntifica con un número de 1 a 100, y cuanto más baja sea esta cifra, más prioridad tendrá. Eso implica que un servidor externo a nuestra red al enviarnos un correo consultará nuestros registros DNS y se lo enviará al de mayor prioridad. Si por casualidad éste estubiese caído lo intentaría con el siguiente en orden descendente.

Existe otro de las zonas directas que es el CNAME, también conocido como Alias. Un registro CNAME apunta a un Host A con lo que ello implica. Si necesitamos que ns.i-david.com, www.i-david.com, ftp.i-david.com apunten a la misma máquina, pero ahorrarnos trabajo al cambiar su IP, haremos que estos anteriormente mencinados apunten a un registro A, que a su vez apuntará a una IP. Modificando el registro A en cuestión estaremos modificando los tres Alias.

Y por otro lado, y que juega un papel muy importante en las restricciones y un sistema de correo en general es el registro PTR, que significa PoinTeR. Su utilidad es, a partir de una dirección IP, obtener el nombre de dominio. A esto se le denomina resolución inversa al contrario de los 3 registros anteriores que son de tipo directo.

localhost

Este es un nombre de máquina usado para dirigirnos a la propia máquina. En el fichero `/etc/hosts` tendremos una referencia a localhost con la dirección IP 127.0.0.1. Esta es la dirección del bucle local. El bucle local es una pseudo-interface de red que se suele utilizar para conectar entre sí servicios de la misma máquina.

Funcionamiento de un sistema de correo

Explicaré por encima como gunciona un sistema de correo, tanto Cliente-Servidor como entre dos servidores.

Cliente1-Servidor1

Nosotros como clientes conectaremos al servidor SMTP desde un puerto aleatorio nuestro a su puerto 25 TCP.

Los comandos básicos de SMTP son `EHLO/HELO`, `MAIL FROM`, `RCPT TO` y `DATA`, sin contar `QUIT`.

Un ejemplo sería:

```
# telnet localhost 25
Trying 127.0.0.1...
Connected to torpedin.i-david.com.
Escape character is '^]'.
220 torpedin.i-david.com ESMTP
ehlo i-david.com 250-torpedin.i-david.com
250-PIPELINING
250-SIZE 10240000
250-ETRN
250 8BITMIME
mail from:<user1@i-david.com>
250 Ok
rcpt to:<user2@example.com>
250 Ok
data
354 End data with <CR><LF>.<CR><LF>
Subject: Correo de prueba
Cuerpo del mensaje
Esta es la segunda linea
y esta la tercera
.
250 Ok: queued as 9BC117BFB6
quit
221 Bye
Connection closed by foreign host.
```

Con el EHLO o HELO saludamos, acto seguido con MAIL FROM indicamos quién envia el mensaje y con RCPT TO, a quien va destinado. Luego pasamos a DATA donde podemos poner el asunto del correo y el cuerpo del mensaje. Para una nueva linea damos intro cuantas veces queramos y para terminar el mensaje escribimos un punto en una nueva linea. Nos despedimos con QUIT.

Servidor1-Servidor2

Entre los servidores también se envían correos por SMTO como si de un cliente y servidor se tratase. Servidor1 resuelve example.com y lo envía a esa dirección IP. En el otro extremo el ser-

vidor estará con el puerto 25 a la escucha y recogerá el correo, pasará por las colas pertinentes y lo más normal es que lo entrege a un usuario suyo.

Servidor2-Cliente2

Una vez el correo ha sido depositado en el buzón del usuario el usuario con un cliente POP o IMAP ha de conectar al servidor. Los servidores POP e IMAP no son parte de Postfix sino que hay que recurrir a servidores específicos como es el caso de *courier-pop* o *courier-imap*.

Nuestro cliente conectará desde un puerto aleatorio al 110 del servidor en caso de POP o al 143 en caso de IMAP. Si el método usado es POP, los mensajes por defecto se borrarán del servidor a no ser que se indique lo contrario, en cambio mediante IMAP los mensajes se leen en el servidor directamente. Otra diferencia significante entre POP e IMAP es que si el IMAP marcamos un correo como leído lo marcamos en el servidor, en el caso de POP, lo marcamos en el cliente utilizando un ID. Esto conlleva a que, al configurar la conexión al buzón POP con otro cliente de correo, volverá a descargar los correos. Y la última diferencia que se me ocurre ahora es, que para un webmail necesitaremos un servidor IMAP.

Ambos, tanto POP (Más conocido como POP3) como IMAP tienen la version cifrada, POP3S e IMAPS, los cuales escuchan por defecto en el 995 y 993 respectivamente.

Sistema de correo con Postfix y Courier

Esta parte del manual se centrará en la creación y gestión de un sistema de correo virtual sobre Debian Sarge (Stable), para manejar diferentes dominios y buzones de correo. Se trabajará contra MySQL porque para esta tanea no necesitamos nada más potente y porque es de las más usadas, además de ser libre y gratuita. Para POP3 e IMAP usaremos Courier.

Configuración previa del sistema, DNS y cortafuegos

Configuramos correctamente los ficheros `/etc/mailname` y `/etc/hostname` de nuestro sistema. Revisamos `/etc/network/interfaces` que tenga la configuración de red adecuada. El sistema DNS (Domain Name Service) es muy importante en un sistema de correo electrónico por lo que en `/etc/resolv.conf` escribiremos los DNS de nuestro ISP, por ejemplo.

```
nameserver 80.58.0.33
```

El fichero `/etc/mailname` y `/etc/hostname` deberáin ser por ejemplo, torpedin.i-david.com.Ahora al intentar resolver torpedin.i-david.com debería resolver a nuestra IP pública, sino, deberíamos crear un HOST A en el panel de nuestro dominio. A continuación una pista.

```
Nombre: torpedin
Tipo: HOST A
Destino: Nuestra IP pública
Prioridad: Es igual
```

A continuación crearemos otro registro. Registro MX que significa Mail Exchange.

```
Nombre: @
Tipo: MX
Destino: torpedin.i-david.com
Prioridad: 1 (Por ejemplo)
```

La prioridad del registro MX es muy importante, pues cuanto más baja sea la cifra más prioridad tendrá y de ella depende, cuando tengamos varios registros MX, a cual enviar primero. Si el servidor de correo que reside en el primer MX está caido, lo intentará con el de la siguiente prioridad más baja, por ejemplo 4.

Por lo que al cortafuegos se refiere, para que podamos recibir correos tanto de otros servidores como de nuestros clientes deberemos publicar el servicio SMTP (Puerto 25) en nuestro cortafuegos. Para POP3, POP3S, IMAP e IMAPS usaremos el 110,995,143 y 993 respectivamente. Si se tiene pensado usar webmail deberemos publicar el servidor web que por defecto escucha en el puerto 80 TCP.

NOTA: Antes de poner el sistema en producción no deberemos publicar el servidor SMTP por razones de seguridad.

Software necesario

A día 20 de Mayo de 2006 el software necesario es el que describiré a continuación. Este manual es probable que sea válido para una futura Debian Etch si los paquetes no varían mucho.

Postfix, Postfix-TLS y Postfix-MySQL se ocuparán de ofrecer SMTP, soporte TLS y SASL y conexión a MySQL, respectivamente. Por otro lado tenemos los componentes de Courier que nos ofrecen el servicio POP, POP3S, IMAP e IMAPS, y la autenticación contra MySQL. Necesitaremos también el SGBD (Sistema Gestor de Bases de Datos) MySQL. Para tener autenticación sobre SMTP necesitaremos una implementacion Cyrus, y eso nos proporciona libsasl2-modules y libsasl2-modules-sql. Y finalmente unas utilidades como dnsutils que nos proporcionará `dig` y `host` para probar los DNS, y mailx que nos ofrecerá el comando `mail` para enviar correos desde terminal.

Hay un problemilla. La versión de Postfix no viene con el parche VDA. VDA es un parche que se le aplica al postfix para implementarle cuotas virtuales, por supuesto, en MySQL. Podemos encontrar más información en `http://web.onda.com.br/nadal/`.

Si no queréis cuotas virtuales para el buzón de correo, pasaremos a la instalación de Postfix estándar situada más abajo.

Instalación de Postfix-VDA

```
# apt-get install apt-build
```

Nos preguntará varias cosas a las que contestaremos la opción por defecto. Para situarnos sobre la contestación deseada usaremos el tabulador.

Actualizaremos la lista de paquetes

```
# apt-get update && apt-build update
```

Visitaremos `http://web.onda.com.br/nadal` para bajarnos el parche más adecuado para nuestra versión.

```
# apt-cache show postfix|grep Version
```

A día de hoy es la 2.1.5-9, asi que:

```
# wget http://web.onda.com.br/nada[...]postfix-2.1.5-trash.patch.gz
```

Limpiamos la caché de descarga del apt-get

```
# apt-get clean
```

```
# cd /var/cache/apt-build/build# gunzip postfix-2.1.5-trash.patch.gz
```

```
  # apt-build install postfix --patch /var/cache/apt-build/build/postfix-2.1.5-tr
-p 1
```

Se pasará un rato compilando. Cuando termine lo instalaremos.

```
  # cd /var/cache/apt-build/repository
```

```
  # dpkg -i postfix_2.1.5-9_i386.deb postfix-mysql_2.1.5-9_i386.deb
postfix-tls_2.1.5-9_i386.deb
```

Si no aparece el asistente, podemos llamarlo con `dpkg-reconfigure postfix`

Ahora instalamos el resto:

```
  # apt-get install courier-pop courier-authmysql courier-ssl
courier-imap courier-pop-ssl courier-imap-ssl libsasl2-modules
libsasl2-modules-sql mysql-server dnsutils mailx
```

Instalación de Postfix estándar

```
  # apt-get install postfix postfix-tls postfix-mysql courier-pop
courier-authmysql courier-ssl courier-imap courier-pop-ssl
courier-imap-ssl libsasl2-modules libsasl2-modules-sql mysql-server
dnsutils mailx
```

Configuración después de la instalación

Una vez descarge los paquetes y se ponga a instalarlo nos aparecerá un asistente de configuración para Postfix. Damos intro para continuar y contestamos lo que nos pregunte.
El tipo de configuración será **Internet Site**

Where should mail for root go: Aquí pondremos otro nombre de usuario. La idea es que los correos salientes enviados como root no llegen como root, sino como el alias que le creemos. Por defecto es NINGUNA o nobody, pero pondremos por ejemplo, postmaster.

¿Nombre de correo? Nuestro nombre de máquina en formato FQDN, osea torpedin.i-david.com.

¿Destinos para los cuales aceptar correo? Cuando nos llege un correo si el dominio al que va dirigido está en esta lista, nos lo quedaremos y lo entregaremos al usuario. Por defecto pondrá "torpedin.i-david.com, torpedin.i-david.com, localhost.i-david.com, localhost", que es correcto.

¿Forzar actualizaciones síncronas en la cola de correo? Ahí explica qué es y cual escoger dependiendo de lo que necesitemos. Por fiabilidad elegiremos "Si".

Ahora nos saldrá el asistente de configuración de Courier-SSL para avisarnos que se creará un certificado X.509, dónd elo guardará y que para usarlo ha de estar firmado por una Unidad Certificadora (como Verising por ejemplo), aunque en éste manual al no ser ricos para pagar a una CA para que nos lo firmen, nos lo autofirmaremos, ya que cifra de todos modos y sirve para ver su resultado.

A continuación el asistente de MySQL donde nos dice qué va a hacer. Una ver terminado aplicará las configuraciones que le hemos mandado, además de generar certificados y arrancar los servicios.

Configuración básica de Postfix

Procuraré ir poco a poco, configurando el main.cf a medida que tengamos lo demás configurado. Los ficheros principales de Postfix son el master.cf, donde se configuran los procesos de van a arrancarse y sus parámetros. No suele tocarse a no ser que usemos Postfix con algun sistema de filtrado o depuración. Por otro lado tenemos el main.cf donde se configuran todos los parámetros relacionados con la función que debe realizar Postfix.

smtpd_banner = $myhostname ESMTP

Por defecto da más información de la que debería, así que además, para cumplir el RFC, lo dejaremos como en la línea anterior.

mail_name = (8.13.6/8.13.6)

Es el software que corre nuestro servidor. Por defecto es Postfix, pero a mi personalmente no me gusta que alguien al ver las cabeceras del mensaje vea cual es mi MTA, así que lo cambio para que parezca la última versión de Sendmail en este momento.

myhostname = torpedin.i-david.com

Será el FQDN de nuestro servidor.

mydomain = i-david.com

Será el nombre de nuestor dominio.

myorigin = $mydomain

Será, para el correo local, el origen del correo. Para sitios pequeños recomientan $myhostname.

mydestination = $mydomain, $myhostname, localhost.$mydomain, localhost

Son los destinos de correo que nosotros manejaremos. Esta opción es muy importánte pues define que correo es para él y cual no. Si es para él lo entrega al buzon de correo de su usuario.

relayhost =

Esta es una opción propia de una configuración *SmartHost*, que funciona como reenviador de correo y por lo tanto no diremos más de ella, tan solo lo comento para evitar confusiones.

mynetworks = 127.0.0.0/8

Con esta opción marcamos la intranet, siendo éstas las redes en las que residen clientes de confianza. A los clientes de confianza se les permite hacer *relay*, que no es más que enviar desde una dirección externa a nuestra juristicción a otra externa. Por ejemplo desde `user@example.com` a `user@example2.com`. Como podría usarse con fines no apropiados, es recomendable dejar solamente 127.0.0.0/8 por seguridad.

local_transport = virtual

De este modo todo correo será entregado mediante protocolo SMTP. Si fuese *local* intentaría entrgarseloa un usuario local

mailbox_size_limit = 0

Es el tamaño máximo del buzón de correo (Bien sea *Maildir* o *Mailbox*) de un usuario local. Usaremos 0 para que sea ilimitado, se definirá en bytes y nunca ha se ser menor que la opción `message_size_limit` del main.cf, o Postfix dará un error al arrancar.

message_size_limit = 20971520

Con esta opción definiremos el tamaño máximo de un correo. En este caso, 20 MB.

inet_interfaces = all

Serán las direcciones en las que escuchará Postfix: all, loopback-only, direcciones IPv4 o IPv6.

home_mailbox = .Maildir/

Será el fichero (*Mailbox*) o directorio (*Maildir*) donde depositará los dorreos de sus usuarios locales. Aunque el sistema de correo con usuarios locale sno es el propósito de éste manual, solamente comentar 2 cosas. La primera, que no es lo mismo `.Maildir` que `.Maildir/`, pues la barra final indica que es un buzón de tipo *Maildir* en vez de tipo *Mailbox*. La segunda es que esto solamente afecta a los buzones de usuarios locales ya que para los virtuales se cogerá el buzón de la base de datos SQL.

Aprovecho ahora para explicar la diferencia entre Maildir y Mailbox de la que he hablado antes. Mailbox digamos que es el método de almacenamiento de correos más antiguo. Maildir fue implementado por primera vez en QMail. Yo personalmente prefiero un buzón de tipo Maildir ya que si hay un fallo en el disco duro o partición, obtendremos un fichero corrupto si usamos Mailbox, y en cambio usando el método Maildir perderemos a lo sumo, unos cuantos correos, sin contar que para Courier-IMAP lo necesitaremos.

strict_rfc821_envelopes = yes

Esta opción configurada indica si cumplir estrictamente el RFC821 y no contenga *basurilla* que el RFC822 permite. Software mal programado no podrá enviar correos. Yo lo tengo activado y hasta ahora ningún problema.

disable_dns_lookups = no

Esta habilitará o deshabilitará la resolución DNS. Es conveniente habilitar las resoluciones sino nuestro MTA se armará un buen lio.

smtpd_delay_reject = yes

Con esta opción haremos que Postfix revise los controles de acceso justo después del comando RCPT TO. Será util pues registrará todo en el log y para analizarlo será más provechoso.

disable_vrfy_command = yes

Deshabilita el comando VRFY. Éste comando sirve para comprobar que el usuario existe en el sistema de correo. Podemos probarlo conectando al SMTP.

```
# telnet localhost 25
```

Trying 127.0.0.1...
Connected to torpedin.i-david.com.
Escape character is '^]'.
220 torpedin.i-david.com ESMTP
`EHLO google.es`
250 torpedin.i-david.com
`VRFY usuario`

smtpd_helo_required = yes

Esto hará que Postfix exiga al cliente/servidor remoto comenzar una sesión SMTP saludando antes con `HELO` o `EHLO`.

maximal_queue_lifetime = 14d

Tiempo que permanecerá el mensaje en cola antes de ser devuelto como imposible de enviar. Daremos un buen plazo por si acaso.

Creo que no se me olvida nada. Ahora deberemos recargar la configuración de Postfix con `postfix reload` o bien reiniciarlo con `/etc/init.d/postfix restart`.

Iremos a una terminal y ejecutaremos:

```
# echo Prueba|mail -s "Test"echo@rediris.es
```

Lo cual, al revisar el log del correo con la orden `tail -f /var/log/mail.log` (Visualizaremos el log en tiempo real) veremos, entre otras cosas:

```
from=<root@i-david.com>, size=305, nrcpt=1 (queue active)
to=<echo@rediris.es>, relay=chico.rediris.es[130.206.1.3],
   delay=12, status=sent (250 2.0.0 k4LICbTO011429 Message
   accepted for delivery)
removed
```

Configuración de Courier POP3 y Courier POP3S

Con aterioridad hemos instalado mediante el comando `apt-get` los paquetes *courier-pop* y *courier-pop-ssl*.

Sus ficheros de configuración residen en `/etc/courier/` con el nombre de `pop3d` y `pop3d-ssl`, de los cuales comentaré sus opciones más importantes. Comenzaré por el `pop3d`.

PIDFILE=/var/run/courier/pop3d.pid

Es el .pid del demonio y lo dejaremos por defecto.

MAXDAEMONS=40

Como dice el fichero de configuración, es el número máximo de demonios arrancados.

MAXPERIP=4

Es el número máximo de conexiones por IP. Si se van a conectar clientes detrás de una máquina que haga NAT, habrá que configurarlo concienzudamente.

DEBUG_LOGIN=0

Con esta opción configuraremos el nivel de debug. Ahora en pruebas podemos dejarlo a 1 o incluso a 2 (muestra usuario y contraseña en el log) peor en producción recomendable dejarlo a 0.

POP3AUTH, POP3AUTH_TLS, POP3AUTH_ORIG y POP3AUTH_TLS_ORIG

Se usará para definir los métodos de autenticación. En nuestro caso usaremos el método PLAIN. Podemos dejarlo como viene por defecto.

PORT=110

Será el puerto en el que escuchará el servicio POP. Por defecto ya dijimos que era el 110 pero como a él solamente conectan clientes, podemos ponerlo en otro puerto siempre y cuando el cliente se configure correctamente.

ADDRESS=0

Si queremos que atienda peticiones de internet, deberemos dejarlo a 0, osea todos los orígenes.

MAILDIRPATH y MAILDIR

Ambas opciones se usan para definir el directorio que hace de buzón de correo a partir de `$HOME` del usuario. Ésto solamente es válido cuando trabajamos con usuarios locales ya que cuando son virtuales atacamos la base de datos MySQL para obtenerlo. Se queda por defecto.

Ahora continuaré con el `pop3d-ssl` que tendrá menos opciones ya que la mayoría las hemos configurado en el anterior fichero.

SSLPORT=995

Éste será el puerto en el que escuchará el POP3S.

SSLADDRESS=0

Atenderá como POP3D, a todos los orígenes para la conexión TLS.

POP3_TLS_REQUIRED=0

Por compatibilidad no requeriremos TLS, sino que será opcional.

TLS_PROTOCOL=SSL3

Por compatibilidad también usaremos SSLv3 peor jamás se os ocurra usar SSLv2 pues tiene fallos de diseño. También podemos usar TLS1 que es más seguro.

TLS_CERTFILE=/etc/courier/pop3d.pem

Ruta al certificado SSL del servidor POP3. Lo dejaremos por defecto, ya que tenemos un certificado que se creó en la post-instalación. Si queremos unos personalizado lo sobreescribiremos y con eso bastará.

Podemos dejar el resto de opciones por defecto.

Configuración de Courier IMAP y Courier IMAPS

La configuración será similar al Courier-POP por lo que las opciones repetidas no las explicaré. Los ficheros en cuestión son `/etc/courier/imapd`.y `/etc/courier/imapd-ssl`.

IMAP_USELOCKS=1

Como dice la ayuda, es recomendable si compartimos directorios IMAP, supongo que así no podrá eliminarse un mensaje si otor client elo está leyendo o editando, por ejemplo.

IMAP_EMPTYTRASH=Trash:7

Eliminará los mensajes del directorio Trash a los 7 días.

Si usásemos **IMAP_EMPTYTRASH=Trash:7,Sent:30** eliminaría además de los mensajes de la papelera a los 7 días, los enviados a los 30.

SENDMAIL=/usr/sbin/sendmail

Es la ruta hasta el binario `sendmail`, el que nos proporciona Postfix.

IMAPDSTART=YES

Si arrancarlo o no, evidentemente sí. La verdad no se si tiene alguna utilidad aparte de no tener que editar los script de unicio del nivel de ejecución para hacer que arranque o no en el inicio del sistema.

MAILDIRPATH=Maildir

Como ocurría con el servicio POP3, será el directorio que hará de buzón de correo a partir del `$HOME` del usuario. Si ejecutamos `echo $HOME` podremos ver el resultado. Esto es solamente válido para usuarios locales ya que cuando usamos autenticación contra MySQL se obtiene de las tablas todo, incluido el directorio buzón de correo.

Respecto al fichero `imapd-ssl`, no tiene opciones que difieran mucho de su compañero.

IMAPDSSLSTART=YES

Con esta opción activaremos o no, si ofrecer SSL sobre el puerto 993 que es por el que trabaja IMAP.

IMAPDSTARTTLS=YES

Al igual que la anterior implementará o no el soporte, en este caso TLS, más fiable que SSLv3

IMAP_TLS_REQUIRED=0

Esta opción requerirá una conexión TLS a todo cliente que intente conectar.

TLS_PROTOCOL=SSL3

Escogemos ese por compatibilidad y solamente escogeremos SSL2 si fues estrictamente necesario.

TLS_STARTTLS_PROTOCOL=TLS1

Que método de cifrado usar cuando usamos la extensión `STARTTLS`.

TLS_CERTFILE=/etc/courier/imapd.pem

Será la ruta hasta el certtificado que se ha generado en la instalación. Aunque podemos usar ese mismo, luego veremos como crearnos nuestor propio certificado y firmarlo nosotros mismos (Advertirá al cliente, peor lo que importa es que la comunicación sea cifrada)

Conexión de Courier con MySQL

Hasta ahora tan solo hemos configurado los servicios POP3, IMAP y sus respectivos protocolos bajo SSL. Esto sería válido para funcionar con usuarios locales, pero desde mi punto de vista, sería un engorro tener un usuario local para cada usuario de correo. Además para ofrecer un servicio virtual con muchos usuarios y varios dominiops lo más cómodo es usar una base de datos SQL, por ejemplo.

Como comenté al principio, usaremos MySQL ya que es la más usada, libre y además gratuita. Aunque el ser la más usada no parezca una razón con fundamento, no es así, ya que podríamos usar MySQL para nuestras webs en PHP, para el servicio de correo, para nuestro servidor jabber, para las configuraciones del webmail, etc. Otra opción sería trabajar contra PostgreSQL.

Vayamos al grano. Necesitamos modificar un fichero de configuración de Courier que nos proporciona el paquete `courier-authmysql` y otro que nos proporciona el paquete `courier-authdaemon`. Todo esto será inútil si no tenemos una tabla en MySQL para conectar a ella, además de un usuario y los permisos adecuados.

Primero, por seguridad estableceremos una contraseña para el usuario `root` de MySQL.

```
# mysqladmin password Contrasenya
```

Ahora cada vez que conectemos con MySQL tendremos que introducir el parámetro `-p` para que nos solicite contraseña.

Comencemos por crear la base de datos llamada *correo*.

```
# mysqladmin -p create correo
```

Una vez creada la base de datos, crearemos la estructura interna, pero como esta documentación no está orientada a crear bases de datos en MySQL, usaremos la plantilla que se adjunta con este manual. Nos situaremos en el home de root.

```
# cd /root
```

A continuación copiaremos el fichero SQL al directorio actual, suponiendo que lo copiamos de un CD, cuya unidad lectora es `/dev/hdc`.

```
# mkdir -p /mnt/cdrom
# mount -t iso9660 /dev/hdc /mnt/cdrom -o ro
# cp /mnt/cdrom/database.sql ./
```

Con el primer comando crearemos el directorio `/mnt/cdrom` por si no existe. Con el segundo montaremos la unidad en el directorio que creamos en el paso anterior y con el tercero lo copiaremos a nuestro directorio actual, que no debería ser otro que `/root`.

Ahora importaremos la base de datos a partir del fichero sql.

```
# mysql -p correo <database.sql
```

Nos devolverá al prompt de bash. Deberemos ahora entrar al servidor, crear un usuario y darle permisos.

```
# mysql -p mysql>show databases; mysql>GRANT SELECT,INSERT,UPDATE,DELETE
on correo.* to user@localhost identified by 'pass';
```

Lo que hemos hecho en el primer comando ha sido abrir la consola MySQL. En el segundo hemos listado las bases de datos que maneja el servidor para comprobar que hemos creado la base de datos *correo*. Si no fuese así, podemos eliminar la base de datos y volver a crearla. La eliminamos con el comando `mysqladmin drop BaseDeDatos` y contestamos que si cuando nos pregunte. Acto seguido la creamos de nuevo e importamos las tablas. Continúo con el tercer paso, en el cual le damos permisos para consultar, insertar, actualizar y eliminar campos de todas las tablas de la base de datos al usuario `user`, cuya contraseña es `pass`.

Ahora vamos a crear un grupo y un usuario local con un uid y gid de valor 1100, por ejemplo.

```
# groupadd -g 1100 vmail
# useradd -u 1100 -g vmail -s /bin/false
```

El primer comando creará un grupo `vmail` con gid 1100. El segundo creará el usuario `vmail` con uid 1100 y que pertenecerá al grupo `vmail`. Además su shell será `/bin/fase` para que no pueda logearse. Que el usuario y grupo sean iguales no es pura coincidencia, porque en realidad podríamos poner los que quisiesemos, al igual que con uid y gid, pero por tener un cierto orden lo he hecho así.

Ya disponemos de un usuario local sobre el que crearemos los virtuales. Lo que haremos ahora será crear los buzones de correo, por razones didácticas nada más, ya que una vez configuremos el Postfix correctamente, la primera vez que llege un correo al usuario se le creará un buzón. Recordemos que usaremos el formato maildir. Simplemente por tenerlo algo orzanizado, seguiremos la estructura `/home/vmail/dominio/usuario`. Vamos a ello.

```
# mkdir -p /home/vmail/i-david.com# mkdir -p /home/vmail/i-david.com
```

La opción -p del comando mkdir hará que, si le ordenamos crear `i-david.com` dentro de `/home/vmail` pero alguno de ellos no existe, lo cree. Ya tenemos el directorio del dominio dentro del cual irán los buzones de usuario, que se crean con el comando `maildirmake`.

```
# cd /home/vmail/i-david.com# maildirmake usuario1# maildirmake usuario2
# chown -R vmail:vmail /home/vmail# chmod -R 700 /home/vmail
```

Nos situamos con el primer comando en el directorio donde estarán los buzones de correo de nuestro dominio. Con el segundo y tercer comando creamos los buzones, y con el cuarto y quinto hacemos dueño de los directorios y subdirectorios al usuario y grupo vmail, y permitimos acceso total solo a su dueño, respectivamente.

Nos encontraremos con 3 directorios dentro de cada buzón.

new

Este es el directorio donde va a parar el correo sin leer.

cur

Aquí se encontrarán los correos ya leídos.

tmp

En este otro, los correos que se están entregando. Una vez depositado en `tmp`, se moverá a `new`.

Nos queda aplicar unos permisos adecuados a los buzones y configurar la conexión a MySQL. Quizá parezca que lo he mezclado eun poco, pero opino que mejor ir por partes, poco a poco, dejando el sistema local preparado para proceder al virtual. Continuemos con la autenticación contra MySQL.

El fichero es `/etc/courier/authmysqlrc`, cuya configuración es muy sencilla.

MYSQL_SERVER localhost

Será la dirección del servidor MySQL. Suponiendo que como es nuestro caso, lo tengamos instalado en nuestra máquina, usaremos localhost.

MYSQL_USERNAME user

Es el usuario con el que conectaremos a la base de datos. Es el `user@localhost` del `GRANT SELECT, UPDATE...`

MYSQL_PASSWORD pass

Y esta es la contraseña que le pusimos al usuario. Es la `IDENTIFIED BY 'pass'` del `GRANT SELECT, UPDATE...`

MYSQL_SOCKET /var/run/mysqld/mysqld.sock

Será la ruta hasta el socket de MySQL. Cuando MySQL no escucha en la interfaces se usa este método.

MYSQL_PORT 3306

Será el puerto del servidor MySQL remoto, en caso de que usasemos uno.

MYSQL_DATABASE correo

Este campo se rellenará con el nombre de la base de datos, que si recordáis se llamaba `correo`.

Los siguientes campos se rellenan con valores especificos de la base de datos. Podéis verlo en el fichero `database.sql`

MYSQL_USER_TABLE users

Será la tabla de la base de datos donde están los usurios, sus contraseñas, la ruta de su buzón de correo, etc.

MYSQL_CLEAR_PWFIELD clear

Será el campo de la tabla users donde se almacenarán las contraseñas en texto plano. Esto no tiene nada que ver con que se transfierna en texto plano, que se puede paliar con conexiones TLS. Se refiere a que la contraseña del usuario se guarda como texto plano en la base de datos. No explicaré como almacenarlo encriptado pues es muy engorroso de modificar pues hay que tocar PAM y demás módulos de autenticación del sistema.

DEFAULT_DOMAIN i-david.com

Este será el dominio por defecto, de modo que si iniciamos sesión como usuario1 interpretará que lo hacemos como usuario1@i-david.com. A gusto del administrador.

MYSQL_UID_FIELD uid

El campo en la tabla users donde estarán los uid de usuario.

MYSQL_GID_FIELD gid

El campo en la tabla users donde estarán los gid de grupo.

MYSQL_HOME_FIELD homedir

De aquí obtiene el directorio home del usuario

MYSQL_MAILDIR_FIELD maildir

Muy importante este campo pues hace referencia al buzón de correo

MYSQL_WHERE_CLAUSE postfix='y'

No la he probado pero tiene toda la pinta de que, si el campo postfix del usuario no es *y*, no le dejará loguear. Si da problemas se comenta poniendo una # delante.

Bueno, una vez tenemos configurado el fichero con los datos de la conexión MySQL, vamos a editar el motor principal. En el fichero `/etc/courier/authdaemonrc` que nos proporciona el paquete `courier-authdaemon`.

authmodulelist=" authpam authmysql"

Reiniciaremos ahora los servicios pertinentes. Courier y Postfix si hemos modificado algo.

```
/etc/init.d/courier-authdaemon restart
/etc/init.d/courier-pop restart
/etc/init.d/courier-pop-ssl restart
/etc/init.d/courier-imap restart
/etc/init.d/courier-imap-ssl restart
/etc/init.d/postfix restart
```

Con todo preparado deberiamos de poner estableder una conexión con el servicio POP3 y autenticarnos.

```
# telnet localhost 110
Trying 127.0.0.1...
Connected to torpedin.i-David.com.
Escape character is '^]'.
+OK Hello there.
user usuario1@i-david.com
+OK Password required.
pass pass
+OK logged in
```

Si hubiese algun problema podríamos revisar el log del correo que se encuentra en `/var/log/mail.log`. Errores comunes son, por ejemplo, tener mal configurados los permisos, olvidarnos de crear los buzones de correo o lo de siempre, introducir mal la contraseña.

Como veis ya tenemos el servicio POP3, al igual que el IMAP funcionando con autenticación contra MySQL. Vamos a configurar ahora el cliente de correo para enviarnos un correo y poder leerlo. No voy a proporcionar imágenes pero si una explicación de lo imprescindible.

Crearemos una cuenta de correo en nuestro cliente de correo con la sigueinte información para el servicio POP3 (Puede variar ligeramnete):

Nombre de la cuenta

Será el nombre con el que identificaremos la cuenta. EL que queramos.

Usuario

Será el usuario con el que nos logearemos en el servidor. Por ejemplo `usuario1@i-david.com`

Contraseña

Es evidente su utilidad. Recordar que distingue mayúsculas de minúsculas.

Servidor

Será la dirección IP o nombre de nuestro servidor POP3. Por ejemplo 192.168.1.162 o torpedin.i-david.com.

Puerto

Dejamos el que nos pone por defecto, 110.

Autenticación

Suelen ser los métodos por los que podemos autenticar contra el servidor. Pinchamos en *Comprobar qué soporta el servidor* y veremos que quizá estén disponibles PLAIN y LOGIN. Escogemos el que queramos.

Eso era referente al POP3 o *recepción de correo*. Ahora el *Envío de correo* o SMTP.

Nombre y **servidor** serán los mismos si queremos

Dirección de correo electrónico

Esto no tiene nada que ver con el usuario para autenticarnos. Esta dirección de correo electrónico es la que aparecerá, al leer el correo en el campo *De:*.

Puerto

En SMTP se usa el 25 por defecto.

Este servidor requiere autenticación

Nos aseguramos de que esta casilla está desmarcada pues aún no hemos implementado ningún método de autenticación sobre SMTP, lo que produciría un error.

Una vez configurado deberíamos poder enviar correo por nuestro SMTP y recibirlo en nuestra bandeja de correo al recibirlo mediante POP3. En el fichero `/var/log/mail.log` podemos ver algo como:

```
uid=1100 from=<usuario1@i-david.com>
message-id=<20060530175826.6A09A7C013@torpedin.i-david.com>
from=<usuario1@i-david.com>, size=313, nrcpt=1 (queue active)
to=<usuario1@i-david.com>, relay=virtual, delay=0, status=sent
  (delivered to maildir)
```

Fijáos. **maildir** no mailbox. Si lo hubiesemos configurado mal saldría mailbox así que tendríamos que modificarlo.

Autenticacion SASL sobre SMTP

Hasta ahora nuestro SMTP era inseguro. Inseguro en el sentido de que cualquiera podría enviar correos a través de nosotros. Podríamos ser utilizados para phishing o enviar SPAM con el peligro de caer nuestra dirección IP en una lista negra.

SMTP (Simple Mail Transfer Protocol) apareció en 1982 y era orientadoa red, sin ningún tipo de autenticación. Ahí es donde entra SASL (Simple Authentication and Security Layer) que proporciona a los protocolos de red, entre ellos SMTP, un método de autenticación. No confundamos autenticación con cifrado ya que aunque usemos SASL si no usamos comunicaciones seguras la contraseña viajará en claro.

¿Por qué SASL y no otro método?

Como he comentado SMTP es un protocolo orientado a red sin ningún tipo de autenticación. Podríamos intriducir en una base de datos los servidores que nos podrían enviar correo así como las IP's de los clientes, pero si las conexiones son dial-up con IP dinámica y el cliente itinerante puede ser un caos además de imposible de mantener. Si a esto le añadimos la de miles de servidores que hay por el mundo, mejor ni hablamos. Se podría optar por una interfaz web sobre IMAP pero no es muy cómoda para el trabajo diario, sobretodo cuando envías mucho correo. Se pensó entonces en el *pop-before-smtp*, de modo que el cliente autentique contra el servidor POP. Esto dejaría durante un tiempo relay abierto a esa IP. Pienso yo que con la proliferación del NAT eso sería un fallo de seguridad pues desde una IP pública pueden enmascararse muchas IP's. Si permitimos relay a la IP pública, desde cualquier máquina detrás de la máquina que enmascara podría enviar correo.

Los paquetes necesarios para esto son `libsasl2`, `libsasl2-modules` y `libsasl2-modules-sql`, que hemos instalado con anterioridad. Crearemos entonces el fichero `/etc/postfix/sasl/smtpd.conf` con la siguiente información:

pwcheck_method: auxprop

Indica el método con el que el servidor valida las contraseñas de los clientes. Si usasemos `saslauthd` se usaría la base de datos local alojada en `/etc/sasldb2`, pero no nos interesa, queremos usar MySQL.

auxprop_plugin: sql

Como es evidente, debemos indicarle el plugin que necesitamos para conectar con una base de datos de tipo SQL.

sql_engine: mysql

Exactamente MySQL

mech_list: LOGIN PLAIN

Permitiremos estos dos métodos de autenticación.

sql_hostnames: localhost

Con esta opción indicaremos el servidor de bases de datos MySQL al que conectaremos.

sql_user: user

Así como el usuario con el que conectaremos.

sql_passwd: pass

Y la contraseña que le hemos asignado con anterioridad.

sql_database: correo

Pero tambien necesitamos saber en qué base de datos hacer las consultas

sql_select: SELECT clear FROM users WHERE email = '%u@%r'

Esta opción es muy importante pues es la colsulta SQL que hará SASL en el servidor MySQL para obtener la contraseña del usuario. Obtenemos el valor del campo `clear` de la tabla `users` cuando el email de unicio de sesión sea el indicado.

NOTA: Cuando di el curso me di cuenta que un error muy común es poner entre sql_database y los dos puntos un espacio. Eso dará lugar a error.

Ya tenemos configurado SASL, pero necesitamos aún activarlo en Postfix, además de usar unas tablas MySQL para que sea más dinámica su configuración. Por ejemplo para que el proceso `local` sepa donde entregar el correo necesita saber cual es el maildir del usuario. Como ahora el maildir estará donde queramos (aunque nosotros lo hemos organizado bien) en vez de en el `$HOME` del usuario, así que tendrémos que hacer que consulte la base de datos para obtenerlo. Comencemos con las conexiones MySQL de Postfix.

Para activar SASL en postfix hay que añadir unas opciones al fichero de configuración

smtpd_sasl_auth_enable = yes

Activará el protocolo de autenticación SASL en Postfix.

smtpd_sasl2_auth_enable = yes

Activará el protocolo de autenticación SASL2 en Postfix.

smtpd_sasl_security_options = noanonymous

Deshabilita métodos que permiten autenticarse como anónimo.

broken_sasl_auth_clients = yes

Los clientes viejos como Outlook Express 5.0 se llevan más con SASL por lo que hay que aplicar esta directiva para que no falle la autenticación.

Ahora tenemos que aplicar una restricción para que solo los autenticados con SASL y nuestras redes puedan enviar. Las restricciones las veremos más adelante con más calma.

```
smtpd_recipient_restrictions = permit_sasl_authenticated, permit_mynetworks,
reject
```

Ahora vamos a crear los ficheros que utilizará Postfix. Para organizarlo un poco, dentro de `/etc/postfix` crearemos un directorio llamado `mysql`. Ya sabéis:

```
# mkdir mysql
```

Vamos a crear los ficheros `mysql-aliases.cf`, `mysql-transport.cf`, `mysql-virtual-maps.cf`, `mysql-virtual.cf`, `mysql-relocated.cf` y `mysql-mydestination.cf`. Os recuerdo que el usuario de MySQL era `user`, su contraseña era `pass` y la base de datos se llamaba `correo`. Su contenido será el siguiente, respectivamente:

```
# mysql-aliases.cf
user = user
password = pass
dbname = correo
table = alias
select_field = destination
where_field = alias
hosts = unix:/var/run/mysqld/mysqld.sock

# mysql-transport.cf
user = user
password = pass
dbname = correo
table = transport
select_field = destination
where_field = domain
hosts = unix:/var/run/mysqld/mysqld.sock

# mysql-virtual-maps.cf
user = user
password = pass
dbname = correo
table = users
select_field = maildir
where_field = email
additional_conditions = and postfix = 'y'
hosts = unix:/var/run/mysqld/mysqld.sock

# mysql-virtual.cf
user = user
password = pass
dbname = correo
table = virtual
select_field = destination
where_field = email
hosts = unix:/var/run/mysqld/mysqld.sock

# mysql-relocated.cf
user = user
password = pass
dbname = correo
table = relocated
select_field = destination
where_field = email
hosts = unix:/var/run/mysqld/mysqld.sock

# mysql-mydestination.cf
user = user
password = pass
```

```
dbname = correo
table = transport
select_field = domain
where_field = domain
hosts = unix:/var/run/mysqld/mysqld.sock
```

Antes de nada vamos a configurar correctamente los permisos de estos ficheros ya que actualmente todos podrían leer los ficheros y por lo tanto ver los datos de la conexión.

```
chown -R root:postfix /etc/postfix/mysql
chmod -R 750 /etc/postfix/mysql
```

Estas tablas, una vez escritas y guardadas las tendremos que usar ahora en Postfix. Postfix en Debian viene enjaulado en el directorio `/var/spool/postfix` para que si aparece algún fallo a repercusión sea menor. Esto nos plantea un pequeño problema, pues la conexión con MySQL la hacemos por socket local que está situado en `/var/run/mysqld/mysqld.sock` y el proceso smtp de Postfix, entre otros está en `/var/spool/postfix`. Es imposible acceder, lo que haremos será un enlace duro entre el socket real y el que estará en la jaula.

Creamos los directorios necesarios dentro de la jaula.

```
# mkdir -p /var/spool/postfix/var/run/mysqld
```

Ahora creamos el enlace duro.

```
# ln /var/run/mysqld/mysqld.sock
/var/spool/postfix/var/run/mysqld/mysqld.sock
```

Un detalle. Si el origen y el destino están en diferentes particiones necesitaremos usar `ln -s` con los mismos argumentos del anterior.

Vamos a probar la conexión, para asegurarnos de que está bien hecho.

```
# mysql -p -S /var/spool/postfix/var/run/mysqld/mysqld.sock
```

Nos pedirá contraseña y deberíamos poner conectar y acceder a la consola MySQL.

Bien, los procesos de Postfix ya podrán conectar a MySQL. Ahora vamos a configurar Postfix para que tenga configuraciones almacenadas en las tablas de la base de datos. Configuraciones como dónde está cada buzón de usuario, redirecciones de correo, cómo enviar el correo... sin entrar en engorrosas tablas locales.

Definiremos cual es la tabla de alias, aunque en realidad no la usaremos pues está orientado a correo local y veremos como podemos hacer todo lo que necesitamos sin falta de estos alias. Si os fijáis, no ponemos el fichero a secas, sino que ponemos `mysql:/` para que postfix sepa que se trata de un fichero de conexión con MySQL. Hay otros tipos de tablas como `hash:/`, `regexp:/` o `ldap:/`, esta última para trabajar contra un servidor LDAP.

```
alias_maps = mysql:/etc/postfix/mysql-aliases.cf
```

Aquí los alias virtuales que introduciremos en la base de datos MySQL. Cuando enviemos un correo a usuario1@i-david.com que se entrege a adminmta@i-david.com, por ejemplo. Esto también nos servirá para ofrecer redirecciones de correo. Supongamos que un usuario no quiere tener cuenta de correo en nuestro servidor, sino recibir su correo en su email personal. Bastaría con redirigir mperez@i-david.com a manolo.perez@gmail.com, por ejemplo. Muy útil sin duda.

```
virtual_alias_maps = mysql:/etc/postfix/mysql-virtual.cf
```

Estos los los mapas de los buzones virtuales, es decir, dónde se encuentra el Maildir de cada usuario.

```
virtual_mailbox_maps = mysql:/etc/postfix/mysql-virtual-maps.cf
```

El transporte para usuarios locales será mediante SMTP

```
local_transport = virtual
```

Lo mismo ocurrirá con el correo de los dominios que manejamos.

```
virtual_transport = virtual
```

Con esta definiremos la lista de usuarios de nuestras máquinas. El `alias_maps` serán los alias para usuario locales. Si le enviamos un correo al usuario *snort* lo recibirá *root*, por ejemplo. Luego tenemos `virtual_mailbox_maps` con el que definiremos usuarios virtuales de los dominios y con `unix:passwd.byname` podremos usar los usuarios de nuestro sistema, aunque en este caso no lo usarémos.

```
local_recipient_maps = $alias_maps $virtual_mailbox_maps
$virtual_alias_maps
```

Esta opción sirve para asignar un UID a los usuarios virtuales. Podriamos hacer un fichero para contactar con MySQl y obtener el UID, pero como hemos creado todos los usuarios virtuales con un UID 1100, lo usaremos estático. Servirá para que al crear el buzón de correo haga dueño al usuario y grupo correspondientes con el UID y después aplique permisos.

```
virtual_uid_maps = static:1100
```

Lo mismo con el GID

```
virtual_uid_maps = static:1100
```

```
virtual_alias_maps = mysql:/etc/postfix/mysql-virtual.cf
```

Será a partir del directorio que definamos, donde se crearán bozones de correo automáticamente. Esto junto con el homedir y el maildir de la base de datos hará que podamos crear buzones de correo automáticamente. Por defecto se deja vacío, pero lo configuraremos con la sigueinte ruta para que por despiste al crear un usuario virtual nos equivocamos, no se cree un buzón fuera de ese directorio

```
virtual_mailbox_base = /home/vmail
```

Aquí debemos configurar el fichero preparado para conectar con MySQL y consultar la información requerida con el método de transporte. Principalmente se usan 3 métodos. `local:`, `virtual:` y `smtp:`. El primero se utiliza para la entrega de correos locales, el segundo para entrega vía SMTP y el tercero lo veremos más adelante.

```
transport_maps = mysql:/etc/postfix/mysql-virtual.cf
```

Ahora necesitamos probar la configuración, así que reiniciamos Postfix como hemos hecho con anterioridad y mandaremos un correo a un usuario.

```
# echo "Esto es un mensaje de prueba" | mail -e -s "Este es el asunto
del mensaje" usuario1@i-david.com
```

La salida del comando `echo` la redirigimos mediante una tubería al comando `mail` y lo enviará a el usuario definido. Postfix se dará cuenta de que `i-david.com` está en la opción `$mydestination` por lo que supondrá que es para él y lo entregará en el buzón de correo del usuario `usuario1` del dominio. Consultará la base de datos MySQL y sabrá que tiene que entregarlo en `$homedir/$maildir`, osea, `/home/vmail/./i-david.com/usuario1/`, por ejemplo. Si no existe creará el directorio y los subdirectorios, hará dueño al usuario que corresponda con el UID y GID correspondiente y aplicará permisos adecuados. Acto seguido el correo pasará por el directorio `tmp` y casi instantáneamente si es un correo pequeño, a `new`. El correo ha sido entregado y permanecerá en el servidor hasta que se recoja mediante POP3.

Cuotas en el buzón de correo

Para terminar explicaré como configurar cuotas en el buzón de correo, muy útiles cuando queremos que no se excedan con los correos que almacenan.

Deberemos crar un fichero nuevo llamado `mysql-quota.cf` para que Postfix sepa obtener la información adecuada de la base de datos. Será muy parecido a los que ya hemos hecho.

```
# mysql-quota.cf
user = user
password = pass
dbname = correo
table = users
select_field = quota
where_field = email
hosts = unix:/var/run/mysqld/mysqld.sock
```

Ahora lo único que necesitaremos será hacer referencia a este fichero en el `main.cf` para que Postfix sepa como conectar con MySQL.

```
virtual_mailbox_limit_maps = /etc/postfix/mysql/mysql-quota.cf
```

De nuevo aplicaremos permisos adecuados.

```
chown -R root:postfix /etc/postfix/mysql
chmod -R 750 /etc/postfix/mysql
```

Si queremos establcer una cuota de disco global (en bytes) usaremos para 100 MB lo siguiente:

```
virtual_mailbox_limit = 100000000
```

Tenemos otra opción interesante que es usar el límite establecido en MySQL ignorando el `virtual_mailbox_limit` anteriormente establecido en el `main.cf`

```
virtual_mailbox_limit_override = yes
```

Para tener un sistema totalmente virtual añadiremos:

```
mailbox_transport = virtual
```

Ahora si queremos aprovechar estas cuotas en IMAP añadiremos al fichero lo siguiente:

```
virtual_mailbox_limit_inbox = yes
virtual_maildir_extended = yes
virtual_create_maildirsize = yes
```

Pero aún nos falta una cosa, acticar la cuota en Courier. Así que vamos a abrir el fichero `/etc/courier/authmysqlrc` y descomentar la línea:

```
MYSQL_QUOTA_FIELD      quota
```

Que antes tenia una # delante.

Procederemos ahora a probar estas cuotas para que veamos como funciona, pero antes tenemos que modificar el usuario para establecersela.

```
# mysql -p
mysql> use correo
mysql> select * from users;
```

Nos fijaremos en la columna de la izquierda, `id` que nos mostrará un número. El `id` de `usuario1@i-david.com` es **2**, por ejemplo, asi que:

```
mysql>update users set quota=200000 where id=2;
```

Ahora el usuario `usuario1@i-david.com` tendrá una cuota de 200 KB (Recordemos que se escribe en bytes). Para probar ahora podemos escribir un mensaje nuevo con mucho texto o adjuntar un fichero. Sería recomendable un fichero de 110 KB aproximadamente para que, al enviar un segundo fichero nos de error por exceso de cuota.

El error viene a ser similar a éste:

```
maildir delivery failed:Sorry, the user's maildir has overdrawn his diskspace quota, please try again later.
```

Controles de acceso

En la gestión de un servidor de correo hoy en día cada vez es más importante el tema de la seguridad en diferentes aspectos. En este documento nos limitaremos a los mecanismos de los que dispone Postfix para controlar en qué condiciones acepta un mensaje.

Un correo pued eentrar de forma loical mediante el programa `/usr/sbin/sendmail` que es una interfaz compatible en su mayor parte con el original de Sendmail o mediante SMTP. En el primer caso el mensaje se acepta siempre, en el segundo caso es donde entran en acción las restricciones, denominadas en la documentación de Postfix como **UCE Controls**.

Existen 4 parámetros para configurar el acceso, cada uno de los cuales puede puede tomar como valor una serie de restricciones. Cada restricicón puede devolver uno de los siguientes tres valores: Aceptar, Rechazar, No-Sé. Un ejemplo de tabla de restricciones estática sería:

```
a.b.c.d OK
e.f.g.h REJECT
i.j.k.l 450 Fallo Temoporal
```

Si la dirección IP del cliente es `a.b.c.d` el mensaje será aceptado. Si aquella es e.f.g.h será rechazado. Si es i.j.k.l será rechazado por un fallo temporal, con lo que volverá a intentarlo más tarde. Si es cualquier otra dirección IP, el resultado será `No-Sé`, por lo que analizará otra restricción, si la hubiese.

Existen las restricciones siguientes, que introduciremos a nuestro gusto en el fichero `main.cf`

smtpd_client_restrictions

De ahora en adelante denominadas **C**, son restricciones que controlan la conexión con el servidor.

smtpd_helo_restrictions

De ahora en adelante **H**, controlarán el saludo del cliente o servidor que nos entrega el mensaje.

smtpd_sender_restrictions

De ahora en adelante **S**, controlan el que envia el mensaje, es decir, la orden `MAIL FROM`.

smtpd_recipient_restrictions

De ahora en adelante **R**, que controlan a quien va dirigido el mensaje, osea, el `RCPT TO`.

Postfix analiza las restricciones del siguiente modo:

- En cuanto una restricción devuelve algo que implica aceptar el mensaje, se evalúan las restricciones que quedan.
- En caso de que una restricción implique rechazar el mensaje, no se analizarán más restricciones y el resultado será rechazar.

- En caso de que una restricción devuelva No-Sé, se evalúan las siguientes restricciones. Si se llega al final, el resultado neto es No-Sé, que en este caso significa aceptar. Si no se desea que el No-Sé final implique aceptación, se puede poner `reject` como restricción final.

Resultado de la evaluación de restricciones

Éste es un punto que no está bien explicado en la documentación de Postfix y confunde a muchos administradores. Si una RC evalúa a rechazar el mensaje, éste es rechazado, sin que haga falta evaluar las RCs de las fases posteriores. Pero si evalúa a aceptar o No-Sé, sí se evalúan dichas RCs. Si queremos rechazar conexiones por dirección/hostname del cliente (caso de un bombardeo o un indeseable, por ejemplo, o permitir conexiones sólo de una serie de máquinas internas) o por nombre de host declarado en HELO/EHLO o por dirección de correo origen (casos estos poco frecuentes por lo fácil de falsear la información proporcionada) entonces sí tiene sentido poner restricciones en las tres primeras RCs. En caso contrario no lo tiene.

Esa es la razón por la que mucha gente pone todas las restricciones en la última RC.

La idea básica es que hasta que no se pasa la fase RCPT TO, normalmente no se puede saber si el mensaje debe ser aceptado o rechazado, pues sea quien sea la máquina cliente y la dirección origen, si el destinatario es local normalmente se aceptará, aunque existen casos que no es así. Por tanto, la idea es no rechazar nada antes de que pase esa fase, salvo que tengamos razones concretas para ello.

A continuación explicaré las directivas más comunes e importantes.

Restricciones orientadas a la IP del cliente

reject_unknown_client

Se puede introducir en los cuatro tipos de restricciones que vimos arriba (CHSR). Cuando un cliente o servidor nos entrega un correo, nuestro MTA comprobará si la IP a.b.c.d tiene un registro PTR (Resolución inversa). Si tiene tal registro en su zona, devolverá un nombre de dominio y nuestro MTA aceptará el mensaje, si no es así lo rechazará. Aunque hay que tener en cuenta que casi todos los ISP ofrecen resolución inversa de sus redes, no todos los hacen, por lo que es muy recomendable usarlo con cautela.

permit_mynetworks

Al igual que la anterior se puede usar en los cuatro tipos de restricciones. Depende de la variable `$mynetworks`, es decir, si una red o equipo está incluida en esta variable, se recibirán correos.

check_client_access

Comprobará de una tabla los clientes que permitimos y los que denegamos basándonos en su dirección IP. Un ejemplo sería `check_client_access = hash:/etc/postfix/filtros/clientes`. Eso usará la tabla clientes, pero primero deberemos crearla.

Crearemos un fichero `/etc/postfix/clientes` e introduciremos dentro de él algo como esto:

```
192.168.0.20          OK
192.168.0.25          REJECT
```

Ahora que tenemos el fichero en texto plano tenemos que generar la base de datos para que Postfix lo entienda.

```
# cd /etc/postfix
# postmap clientes
```

Esto último nos generará un fichero clientes.db binario, pero recordad que se usará `hash:/etc/postfix/filtros/clientes` y no `hash:/etc/postfix/filtros/clientes.db`. Comentar que esta opción es muy poco práctica a la vez que costosa de mantener.

reject_maps_rbl

RBL, acrónimo de Relay Black List, son unas bases de datos libres y gratuitas en internet, aunque algunas veces de pago que se vasa en, cuando el cliente o servidor conecta a nuestro servidor, comprobar que no está en una lista negra. Si lo está, será rechazado. Desde mi punto de vista es mal método si no las usamos con cabeza, ya que no todas son iguales ni tienen la misma política. Unas por ejemplo incluirán en su lista negra a clientes de Telefónica, Jazztel y demás compañias que tengan IP dinámica, en cambio otra tendrá la política de banear solamente a los servidores que son Open Relay. Un servidor Open Relay es aquel que permite enviar correo desde el exterior con destino un dominio externo (que nosotros no manejamos), y sin autenticar. Son servidores mal configurados y que se suelen utilizar para enviar SPAM.

`http://www.abuse.net/relay.html`

`http://www.declude.com/Articles.asp?ID=97`

permit_sasl_authenticated

Muy importante, ya que gracias a ella podremos permitir a los que se hayan autenticado correctamente con SASL y luego rechazar lo que queramos.

Restricciones orientadas al saludo inicial

reject_invalid_hostname

El mensaje será rechazado si el HELO es inválido, como por ejemplo `-1272067392>`

reject_unknown_hostname

No lo he comentado, así que aprovecho ahora. El HELO debe ser un FQDN, como por ejemplo `torpedin.i-david.com` o `google.es`, pero no todos los servidores/clientes saludan correctamente. El MTA intentará resolver el nombre de dominio (como cuando usamos en consola `# host google.es`) y si no tiene un registro A o MX asociado, rechazará el correo.

reject_non_fqdn_hostname

Como he mencionado antes el HELO ha de ser un FQDN, por lo que con esta restricción, si no cumple la condición, el correo será rechazado.

Restricciones orientadas al remitente

reject_unknown_sender_domain

Esta comprobación se basa en verificar que, si nos envían un correo desde, por ejemplo `ejemplo@nx.com`, `nx.com` tiene un registro A o MX asociado. Si no lo tiene se rechazará el correo. Es una opción que veo muy útil y poco probable de recibir falsos positivos, a no ser que nuestro servidor DNS se demorase en contestar y no resolviese el nombre.

reject_non_fqdn_sender

Esta otra exigirá que el dominio del servidor que nos envía el correo sea FQDN. Por ejemplo, se rechazaría un correo de `ejemplo@pruebas` pero se aceptaría de `ejemplo@pruebas.com`

Restricciones orientadas al destinatario

reject_unknown_recipient_domain

Aunque este tipo de restricciones no las usarémos apenas pues son muy útiles en un reenviador de correo (Relay Host), pero para nada en nuestro caso. Al enviar un correo a `ejemplo@dominio.com`, si `dominio.com` no existe rechazará el mensaje.

reject_non_fqdn_recipient

Por último esta restricción que exije que el dominio al que se envía el correo sea un FQDN. El ejemplo sería muy parecido al de la restricicón `reject_non_fqdn_sender`.

Consejos

Hay que tener cuidado con las restricciones que aplicamos pues algunas pueden rechazar correos legítimoos prque quien nos envía el correo saluda mal, por ejemplo con `localhost.localdomain`. Decir también que toda restricción que implique hacer una consulta DNS tendrá un retardo, lo que tarde en resolver el dominio u obtener el registro PTR asociado a la dirección IP. Debéis de planificar qué necesitáis y que no necesitáis.

Ejemplos

```
smtpd_recipient_restrictions =
permit_sasl_authenticated,
reject_invalid_hostname,
reject_unknown_recipient_domain,
reject_unknown_sender_domain,
reject_invalid_hostname,
reject_non_fqdn_recipient,
reject_non_fqdn_sender,
reject_non_fqdn_hostname,
reject
```

Podemos ponerlo todo en una línea, pero por comodidad lo estructuraremos en varias, incluso con tabulaciones si queremos.

Filtros de correo

Ha llegado la hora de ver los filtros de correos. Comenzaré por los filtros que implementa Postfix, muy sencillos y en gran parte, fáciles de configurar.

Filtros integrados de Postfix

Las opciones a incluir en el fichero `main.cf` son `header_checks` y `body_checks`. Como su nombre indica `header_checks` filtrará mediante expresiones regulares (tablas de tipo regexp:/ evidentemente) las cabeceras del mensaje, y `body_checks` se ocupa de filtrar el cuerpo. Esto no se usará como antispam ni como antivirus, sino como un método adicional de protección. Sabemos que muchos gusanos que se envian por correo electrónico suelen usar el mismo asunto en el mensaje, o palabras clave en el cuerpo del mensaje, de modo que podríamos utilizar estos filtros para filtrar esos mensajes no deseados.

Añadiremos la característica añadiendo al `main.cf` la linea:

```
header_checks = regexp:/etc/postfix/filtros/cabeceras
```

Y dentro del fichero `cabeceras` añadiremos:

```
/^Subject: Hola/       REJECT
/^From: root@prueba\.red/i       REJECT
```

Ahora si el asunto de un pensaje contiene Hola, Holas, hola, HOLA o cualquiera de sus variantes, será rechazado. En el segundo caso, si el correo es enviado por root@pruebas.red, será rechazado. Esos 2 ejemplos creo que servirán para hacerse una idea dle funcionamiento de este tipo de filtros

NOTA: Si os habéis dado cuenta ya hemos visto tablas MySQL `mysql:/`, tablas binarias `hash:/` y tablas de expresiones regulares `regexp:/`.

Por otro lado, los `body_checks` que filtrarán el cuerpo del mensaje basándose como el anterior, en expresiones regulares.

Crearemos un fichero `cuerpo` dentro del directorio `/etc/portage/filtros` y escribiremos en él lo siguiente:

```
/no me ha pagado/ REJECT
/me cae mal/ REJECT
```

Ambas se ocuparán de filtrar ese texto en el mensaje. Si no recuerod mal, no distinge mayúsculas y minúsculas.

Configuración de Amavis como filtro de correo

Amavisd-new es una armazón de filtrado que utiliza otros aplicaciones para el filtrado de correo. Las ventajas son la sencillez y modularidad de su configuración. En este caso utilizaremos dos aplicaciones auxiliares, ClamAV para el filtro de correo con virus y Spamassassin para el filtrado de correo no deseado. Spamassassin incorpora filtros bayesianos, lo cual da la

posibilidad de hacerle aprender que correo es deseado y cual no. Lo malo de los bayesianos es su larga curva de aprendizaje.

La web oficial de Amavis es `http://www.amavis.org`
La web de ClamaAV es `http://www.clamav.net`
La web de Spamassassin es `http://spamassassin.apache.org`

Procederemos a instalar Amavis con Spamassassin y ClamAV, además de unzip para examinar ficheros ZIP.

```
# apt-get install amavisd-new clamav clamav-daemon spamassassin unzip
```

Tras instalar nos preguntará como queremos actualizar la base de datos antivirus de ClamAV. Seleccionaremos el método `cron`.

A continuación nos pregunta cual será el servidor con el que sincronizar la base de datos. Es conveniente usar uno próximo a nosotros, por lo que escogeremos `db.es.clamav.net (Spain)`. Una vez seleccionado, pulsaremos `TAB` para ponernos sobre `Aceptar` y dar `ENTER`.

Ahora nos pedirá datos del proxy (Si utilizásemos). Continuamos dejándolo vacío pues doy por supuesto de que no utilizamos.

Por último nos pregunta si queremos recargar la base de datos cada vez que se actualizan las reglas. Contestaremos `no` pues de eso se ocupará a Amavis, y al reiniciar Amavis se reiniciará todo el sistema de filtrado.

En este paso el ClamAV actualizará su base de datos automáticamente.

El concepto es el siguiente. Un correo ha de llegar a nuestro puerto 25 (Postfix). Este será enviado al puerto 10024 (Por ejemplo) de Amavis, y Amavis una vez analizado nos lo enviará de nuevo a otro puerto (El 10025 por ejemplo) de Postfix sin restricción alguna entre ellos. La cuestión es ¿Porque a otro puerto? Es sencillo, si Amavis devuelve el correo analizado a Postfix por el 25, Postfix volverá a entregárselo y entraría en un bucle infinito. Si hacemos que Postfix reciba correo de Amavis en otro puerto, el problema se solucciona.

Lo primero es añadir al `master.cf` un nuevo servicio, el Postfix secundario. Porque queremos, lo pondremos en el puerto 10025.

```
127.0.0.1:10025 inet n - n - 20 smtpd
-o content_filter=
-o local_recipient_maps=
-o relay_recipient_maps=
-o smtpd_restriction_classes=
-o smtpd_client_restrictions=
-o smtpd_helo_restrictions=
-o smtpd_sender_restrictions=
-o smtpd_recipient_restrictions=permit_mynetworks,reject
-o mynetworks=127.0.0.0/8
-o strict_rfc821_envelopes=yes
-o smtpd_error_sleep_time=0
-o smtpd_soft_error_limit=1000
-o smtpd_hard_error_limit=1001
```

Lo que estamos consiguiendo con esto es poner a la escucha en localhost (Solo podremos acceder por bucle local), en el puerto 10025 con unas opciones muy permisivas. Aquí el ser poco estricto da igual pues lo que queremos es que este proceso de Postfix no sea nada restrictivo.

En Debian, Amavis corre como un servicio más y se arranca, detiene y reinicia con `/etc/initd.d/amavis` start—stop—restart.

Si ejecutamos `# netstat -atn|grep LISTEN` veremos como hay un servidor escuchando en el puerto 10024.

Parece que ya está todo preparado. Postfix en el 25 escuchando, Amavis en el 10024 y una segunda instancia de Postfix en el 10025. Nos queda algo muy importante, hacer que se reenvien los correos.

. Como de costumbre tendremos que editar el fichero `main.cf`. Hay que añadir una opción para que los correos que llegen se reenvien al Amavis.

```
content_filter = smtp:[127.0.0.1]:10024
```

Ahora los correos llegarán a Postfix y serán, una vez aceptados (Si se aceptan), enviados al puerto 10024.

Ahora una vez Amavis analice el correo haremos que nos lo envíe al puerto 10025, para lo cual tenemos que editar el fichero `/etc/amavis/amavisd.conf`.

En la línea 105 tendremos la opción necesaria comentada con una # delante. Lo descomentamos y nos aseguramos de que coincide con nuestra configuración.

```
$forward_method = 'smtp:127.0.0.1:10025';
```

Con esto bastará. Reiniciamos Amavis y Postfix

```
# /etc/init.d/postfix restart
 # /etc/init.d/amavis restart
```

Configuración de ClamAV para su correcto funcionamiento con Amavis

Si recordáis, hemos instalado el paquete `clamav-daemon`. Tendremos que arrancarlo (Aunque despues de la instalación ya lo hace él solo). Para que todo se integre correctamente, ClamAV tiene que ejecutarse como el mismo usuario de Amavis (El usuario `amavis`) y además tener permiso para escribir en el directorio `/var/run/clamav`.

Deberemos primero configurar ClamAV para que se ejecute como usuario `amavis`. Editamos `/etc/clamav/clamd.conf` para que la línea:

`User clamav`

Quede así:

`User amavis`

Cuando reiniciemos ClamAV con la orden `/etc/init.d/clamav-daemon restart` no nos dará error alguno peor en realidad no ha modido crear el socket al que conectará Amavis. Debemos de darle permiso de escritura al usuario `amavis` en el directorio `/var/run/clamav`, y qué mejor forma que hacerle dueño.

```
# chown -R amavis:amavis /var/run/clamav
```

Ya está todo listo, reiniciaremos los servicios `amavis` y `clamav-daemon` como ya vimos con anterioridad.

Antes de terminar esta sección quiero comentar que podemos escanear el correo entero o solamente los adjuntos. Las opciones están en el fichero `/etc/clamav/clamd.conf` y son `ScanMail` y `ScanArchive`. Las comentamos o descomentamos a nuestro gusto.

Configuración de Spamassassin

Es hora de configurar el filtro de correo Spamassassin. Como lo usaremos con Amavis, editaremos el fichero `etc/spamassassin/local.cf` y `/etc/amavis/amavisd.conf`, este último llendo a la línea 1098 y revisando las siguientes opciones.

NOTA: Dos ficheros de configuración pueden generar confusión pero ambos son necesarios. Tan solo recordar que las opciones de `amavisd.conf` prevalecen sobre las de `local.cf`.

$sa_local_tests_only = 0;

Lo desactivaremos para que haga todo tipo de restricciones, requieran conexión a internet o no. Es útil si usásemos listas negras en SpamAssassin

$sa_timeout = 30;

Cuando se realiza algún tipo de consulta a internet puede que el servidor no responda, así que podemos establecer un tiempo antes de dar tiempo de espera agotado.

$sa_mail_body_size_limit = 150*1024;

Establece el tamaño máximo en bytes del cuerpo de un mensaje para que no sea anlizado. Como dice el fichero de configuración, solamente uno de cada 100 mensajes de SPAM sobrepasa los 64 KB. Le hemos asignado 150*1024 bytes, 150 KB. La verdadera utilidad es no consumir recursos si el mensaje es muy grande pues podemos estar casi seguros de que no será SPAM.

$sa_tag_level_deflt = -100;

Spamassassin añade unas cabeceras a los mensajes cuyo valor por defecto en cuanto a nivel de probabilidad de SPAM se refiere, es 4. Es interesante tener siempre esas cabeceras presentes así que lo configuraremos a -100. Esto hara que si el valor es mayor que -100 se añadiran pues es muy raro que una vez analizado se un valor inferior a -100.

$sa_tag2_level_deflt = 5.0;

Este es el nivel que marcará si se detecta SPAM o no. Por defecto es 6.3, aunque yo uso 5 y me va de maravilla. Debemos de probar enviando correos y recibiendo para observar si con cierto nivel tenemos falsos positivos.

$sa_kill_level_deflt = $sa_tag2_level_deflt;

Con esta opción configuraremos cual será el valor con el cual se rechazará el correo. Lo más normal es, que si en la opción anterior configuramos un nivel 5.0 o superior como si fuese SPAM, usemos ahora ese valor para rechazar el correo.

$sa_dsn_cutoff_level = 10;

Cuando un correo es detectado como SPAM, un mensaje se envía a postmaster. Si sobrepasa este nivel, 10, no se enviará ese aviso y se rechazará sin más.

$sa_spam_subject_tag = '*SPAM***';**

Con esta opción añadiremos al asunto el texto que pongamos.

$sa_spam_modifies_subj = 1;

Sin esta opción activada la anterior no servirá de nada

Configuración de decisiones de Amavis

Amavis, usando las aplicaciones como ClamAV o Spammassassin puede detectar si el contenido del mensaje es deseado o no, peor solamente nosotros podremos decirle qué hacer con él. Si nos situamos en la línea 493 veremos unas opciones.

```
$final_virus_destiny = D_DISCARD; # (defaults to D_DISCARD)
$final_spam_destiny = D_PASS; # (defaults to D_BOUNCE)
```

Indicarán qué hacer en caso de que el correo sea detectado como SPAM o como contenedor de virus. Claramente si es un virus, lo más recomendable es descartarlo y si es SPAM, quizá descartarlo o dejarlo pasar. Personalmente yo lo dejo pasar por una sencilla razón: El correo tendrá la cabecera `X-SPAM-Flag: Yes` si es SPAM y con mi cliente de correo electrónico lo que es supuestamente SPAM, lo envío a otro directorio. La ventaja es la rapidez con la que Amavis gestionará los correos y que no perderé correos a causa de falsos positivos. Las desventajas es que el correo se recibe de todos modos peor como dije antes, podemos organizarlo gracias a las cabeceras.

Manejo de un dominio alternativo

En esta sección explicaré como manejar el correo de un segundo dominio. Cómo configurar las entradas MX en el panel del dominio (en casi todos es igual) lo he hecho al principio. Vamos a cambiar las tornas y poner un ejemplo real. Mi dominio, `elgura.com`, con un servidor de correo Postfix con todo lo que hemos implementado hasta ahora. Otro dominio cuyo nombre es `i-david.com`, de un amigo. Sus webs están en otro servidor pero el correo llega a mi servidor. Cuando hacemos las consultas DNS responde lo siguiente:

```
# host elgura.com
elgura.com has address 83.35.200.4
elgura.com mail is handled by 1 mail.elgura.com.

# host i-david.com
i-david.com has address 212.89.30.54
i-david.com mail is handled by 1 mail.elgura.com.
```

Como véis el registro MX de ambos dominios (`mail is handled by`) apunta al FQDN de mi servidor de correo `mail.elgura.com`.

Para trabajar con otro dominio (crear el nuevo registro en la tabla transport, crear usuarios, etc) usaremos la plantilla adjunta `plantilla_users.sql`.

Una vez escrita y editada a nuestro gusto ejecutamos, suponiendo que está en `/root`:

```
# cd /root/
 # mysql -p <plantilla_users.sql
```

Si necesitáis crear redirecciones podéis basaros en el fichero `database.sql` para añadir elementos a la plantilla.

Lo que pretendo con estas plantillas es que no necesitéis aprender mucho SQL.

Recursos

`http://www.postfix.org` Página Oficial de Postfix

`http://www.postfix.org/uce.html` Postfix UCE Controls

`http://www.uco.es/ccc/sistemas/postfix/` Documentos de la Universidad de Cordoba acerca de Postfix

`http://web.onda.com.br/nadal` Parche VDA para postfix

`http://www.postfix.org/postconf.5.html` Manual de Postconf donde están todas las opciones disponibles

Licencia

Créditos

Realizado por Ricardo Bartolomé Méndez `ricardo@elgura.com`

Documento realizado con LaTeX