Postfix

(Versión 2.0)

[Tutorial]

Fernando Limón

Madrid, Enero de 2003

Gracias a todos los que han colaborado en la elaboración de este documento. En especial a Ramón Pons Vivanco.

ÍNDICE

1.- Introducción.....1
1.1.- Arquitectura.....1
1.2.- Descripción de main.cf.....3
1.3.- Descripción de master.cf.....5
2.- Configuración.....10
2.1.- Estafeta de Primer Nivel.....10
2.2.- Estafeta de Segundo Nivel.....13
3.- Soporte de múltiples dominios.....15
4.- Soporte de Transportation Layer Security (TLS).....18
4.1.- Compilación.....19
4.2.- Configuración.....19
5.- Soporte de Simple Authentication and Security Layer (SASL).....22
5.1.- Compilación.....22
5.2.- SMTP AUTH en todos los puertos SMTP.....24
5.3.- SMTP AUTH en determinados puertos SMTP.....25
5.4.- Postfix como cliente SMTP AUTH.....27
6.- Integración de filtros por contenido.....29
6.1.- Solución simple.....29
6.2.- Solución básica.....32
6.3.- Solución avanzada.....34
7.- Mecanismos de seguridad.....37
7.1.- Notificaciones a Postmaster.....37
7.2.- Rechazo por no resolución inversa.....38
7.3.- Verificación de RBL.....38
7.4.- Requerimiento de HELO/EHLO y verificaciones.....39
7.5.- Filtrado por cabeceras.....39
7.6.- Filtrado por contenido.....40
7.7.- Limitación del tamaño de mensaje.....41
7.8.- Ejecución en modo chroot.....41

1.- Introducción.

En 1998 comenzó a difundirse el uso de un nuevo sistema de gestión de correo electrónico bajo la denominación de IBM Secure Mailer, aunque posteriormente pasaría a denominarse Postfix. Este producto se desarrolló en el centro de investigación Thomas J. Watson Research Center, de IBM.

El autor, *Wietse Zweitze Venema*, conocido por sus desarrollos software para la protección contra intrusiones de sistemas informáticos, había elaborado un sistema de gestión de correo electrónico cuyas características básicas eran rapidez, facilidad de configuración y, sobre todo, seguridad.

Son muchos los que en alguna ocasión han intentado probar Postfix con el fin de analizar sus prestaciones y facilidad de configuración. Desgraciadamente las configuraciones que se instalan en las distribuciones distan mucho de ser fácilmente comprensibles, al incluirse multitud de tablas y ficheros que no se utilizan.

Considero que lo más didáctico es partir de una configuración base sencilla y bien documentada, y que las necesidades puntuales harán que cada administrador profundice en las soluciones concretas que aporta Postfix a su problemática.

Éste es el objetivo de este documento: proporcionar una configuración base sencilla, comentada, que atienda a necesidades generales, y que pueda servir como núcleo de una configuración más compleja.

En la elaboración de esta guía se han tenido en consideración los criterios de seguridad y calidad planteados por la iniciativa Red Académica de Correo Electrónico (RACE) propuesta por RedIRIS.

Por tanto, no es objetivo de este documento proporcionar una descripción exhaustiva de directivas y opciones que soporta Postfix. Para ello ya existen numerosas referencias en Internet y en fondos bibliográficos.

En una primera parte se proporcionará una visión arquitectónica y funcional de Postfix, para posteriormente ir abordando problemáticas concretas y plantear solución a las mismas. Solución que en ocasiones puede que no sea la única, en cuyo caso entrará en juego la visión subjetiva del autor.

1.1.- Arquitectura.

Al contrario de Sendmail, que es un gestor de correo monolítico, en el diseño de Postfix se han disgregado los diversos tratamientos que se realizan sobre un mensaje a su paso por un Mail Transfer Agent (MTA), adjudicando cada tratamiento o grupo de tratamientos a un proceso independiente. El conjunto de todos estos procesos es Postfix.

Los procesos que conforman Postfix se comunican a través de *sockets* que se crean, por razones de seguridad, en un directorio de acceso restringido. La información que intercambian los diversos procesos es la mínima posible, limitándose en la mayoría de los casos a la referencia de la entrada en una cola y la relación de destinatarios, o a un simple identificador de estado.

La siguiente figura proporciona una visión global de los elementos que componen Postfix:

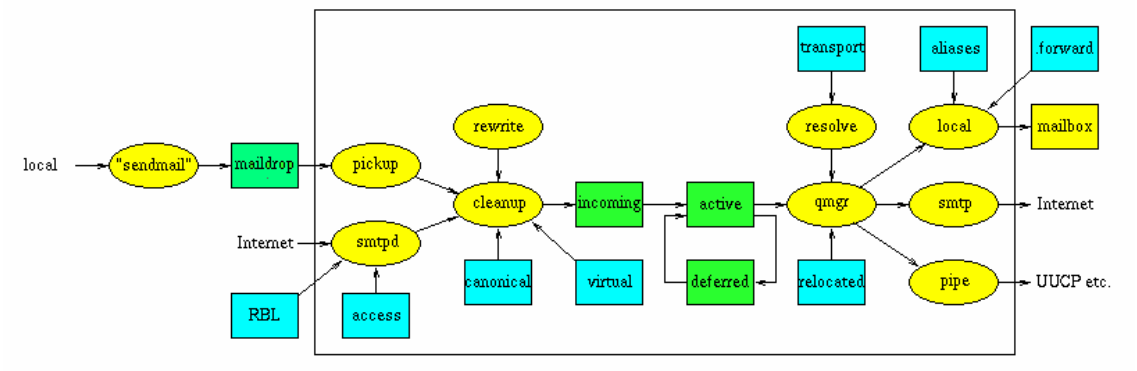

Figura 1

Postfix basa su funcionamiento en cuatro colas: *maildrop*, *incoming*, *active* y *deferred* (cuadrados coloreados en verde).

El correo que se genera de forma local se deposita en *maildrop* para su posterior proceso. El proceso *pickup* toma los mensajes que llegan a *maildrop* y los pasa a *cleanup*, que analiza las cabeceras de los mensajes y deposita éstos en la cola *incoming*.

En la cola *active* se encuentran aquellos mensajes que están en fase de encaminamiento, y en *deferred* los mensajes que por diversas causas no se pueden encaminar o están pendientes de reintentar su encaminamiento.

El proceso *qmgr* es el encargado de tratar los mensajes que llegan a la cola *incoming*, depositarlos en *active* y lanzar el proceso adecuado para su encaminamiento, como pueden ser *local*, *smtp* o *pipe*.

El correo procedente de otros sistemas se atiende a través del proceso *smtpd*, utilizando el protocolo SMTP, pudiendo utilizar accesos a servidores de RBL o tablas internas para aplicar las políticas de acceso a cada mensaje entrante.

Coloreadas de azul aparecen las tablas que, creadas por el administrador, sirven a los diferentes procesos para concretar el tratamiento que debe darse a cada mensaje. Se usan seis tablas: *access*, *aliases*, *canonical*, *relocated*, *transport* y *virtual*. Aunque no es obligatoria la existencia ni utilización de todas ellas.

La tabla *access* permite definir una relación explícita de sistemas a los que se les deben aceptar o rechazar sus mensajes. La utiliza el proceso *smptd*.

La tabla *aliases*, al igual que en Sendmail, define una serie de nombres alternativos a usuarios locales, y la consulta el proceso *local*.

El proceso *cleanup*, mediante la tabla *canonical* establece relaciones entre nombres alternativos y nombres reales, ya sean usuarios locales o no.

El proceso *qmgr* utiliza la tabla *relocated* para devolver los mensajes de usuarios que han cambiado de dirección: “User has moved to *new-email*”.

Con la tabla *transport*, que es utilizada por el proceso *trivial-rewrite*, se define la política de encaminamiento por dominios, subdominios e incluso por dirección concreta de usuario.

Para la gestión y soporte de dominios virtuales el proceso *cleanup* utiliza la tabla *virtual*. En ella se establecen las relaciones entre usuarios virtuales y reales, e incluso de dominios completos.

Todas estas tablas pueden usar alguno de los siguientes tipos de formato de base de datos:

- Fichero binario indexado (btree, hash, dbm, etc).
- Fichero de texto basado en expresiones regulares (regexp).
- Sistema externo de base de datos (NIS, LDAP, MySQL, etc).

Para conocer qué tipos de formato de base de datos soporta nuestra instalación, se puede usar la directiva **/usr/sbin/postconf –m**

Para indicar a Postfix el método de acceso a un determinado fichero se antepone al nombre del mismo el método de acceso. Así por ejemplo *hash:/etc/postfix/tabla* indica que */etc/postfix/tabla* es un fichero en formato *db*.

Para crear los ficheros binarios indexados, Postfix dispone de la directiva: **postmap**. Por ejemplo, para generar el correspondiente binario del fichero anterior se usaría la directiva **postmap /etc/postfix/tabla**, con lo que se crearía el fichero */etc/postfix/tabla.db* .

En el caso particular de la tabla *aliases*, existe la directiva **/usr/sbin/newaliases** .

Los ficheros que contienen expresiones regulares se tratan directamente desde Postfix.

1.2.- Descripción de main.cf.

El fichero donde se define gran parte del funcionamiento de Postfix es **main.cf**, que habitualmente se encontrará en el directorio */etc/postfix*.

No tiene una estructura concreta, por lo que se presenta en este documento responde a una distribución razonable de entre todas las posibles.

Para un análisis detallado de las directivas se recomienda consultar la página oficial de Postfix *http://www.postfix.org/*

```
#########################################################
#
# Fichero de configuracion main.cf
#
# Autores:
# Fernando Limon - flimon@fi.upm.es
# Ramon Pons Vivanco - rpons@laurel.datsi.fi.upm.es
#
# (Tuesday 07th of January 2003 12:02:20 PM) - Version 1.04
#
#########################################################

# NOMBRE, DOMINIO y SMARTHOST
myhostname =
mydomain =
relay =

# DIRECCION QUE APARECE EN EL FROM
myorigin = $mydomain

# CONFIGURACION DE TRANSPORTATION LAYER SECURITY (TLS)
smtp_use_tls = yes
smtp_tls_session_cache_database = sdbm:/etc/postfix/smtp_scache
smtp_tls_key_file = /etc/postfix/cert/key.pem
smtp_tls_cert_file = /etc/postfix/cert/cert.pem
smtp_tls_CAfile = /etc/postfix/cert/CertCA.pem
smtpd_use_tls = yes
smtpd_tls_session_cache_database = sdbm:/etc/postfix/smtpd_scache
smtpd_tls_key_file = /etc/postfix/cert/key.pem
smtpd_tls_cert_file = /etc/postfix/cert/cert.pem
smtpd_tls_CAfile = /etc/postfix/cert/CertCA.pem

# CONFIGURACION DE SIMPLE AUTHENTICATION AND SECURITY LAYER (SASL)
broken_sasl_auth_clients = yes
smtpd_sasl_local_domain =
smtpd_sasl_auth_enable =
smtpd_sasl_security_options = noanonymous

# UBICACION DE DIRECTORIOS
queue_directory = /var/spool/postfix
command_directory = /usr/sbin
daemon_directory = /usr/lib/postfix

# PROPIETARIO DE COLAS Y PROCESOS
mail_owner = postfix
setgid_group = postdrop

# TRATAMIENTO DE ALIAS
```

```
alias_database = hash:/etc/aliases
alias_maps = hash:/etc/aliases

# IDENTIFICACION DE USUARIOS LOCALES
local_recipient_maps = $alias_maps unix:passwd.byname

# ENVIO EN PARALELO A UN MISMO DESTINO
local_destination_concurrency_limit = 2
default_destination_concurrency_limit = 10

# OTROS PARAMETROS DE CONFIGURACION
notify_classes = resource, software, policy
transport_maps = hash:/etc/postfix/transport
disable_vrfy_command = yes
disable_dns_lookups = no
#relayhost = [$relay]
message_size_limit = 10485760
mailbox_size_limit = 0
maximal_queue_lifetime = 5d

# CONTROL DE CORREO ENTRANTE / SALIENTE
mynetworks = 127.0.0.1 192.168.0.0/24
mydestination = $myhostname localhost.$mydomain $mydomain
smtpd_client_restrictions = reject_unknown_client
smtpd_helo_required = yes
smtpd_helo_restrictions =
         reject_invalid_hostname
         reject_unknown_hostname
         reject_non_fqdn_hostname
smtpd_recipient_restrictions =
         permit_sasl_authenticated
         permit_mynetworks
         reject_unauth_destination
header_checks = regexp:/etc/postfix/header_checks
body_checks = regexp:/etc/postfix/body_checks
content_filter = vscan:

# VERSION
mail_name = Postfix-TLS-SASL/MJ-1.04
smtpd_banner = $myhostname ESMTP $mail_name ($mail_version)
```

1.3.- Descripción de master.cf.

El programa *master* se ejecuta de forma continua en el servidor de correo, y su función básica consiste en recibir indicaciones de unos procesos e iniciar otros. Es, por

así decirlo, el director de orquesta para los distintos programas que conforman Postfix. El fichero *master.cf*, que habitualmente se encuentra en el directorio */etc/postfix*, contiene la información necesaria para que el programa *master* llame de forma correcta a cada uno de los procesos.

Cada entrada en el fichero es un conjunto de ocho campos separados por blancos o tabuladores, y cuyo orden y significado es:

service	type	private	unprivileged	chroot	wakeup	maxprocess	command

Service: Nombre del servicio que se está configurando.

Type: Tipo de comunicación de transporte utilizado por el servicio.

Private: Restricciones de seguridad a procesos externos.

Unprivileged: Ejecución en modo no privilegiado.

Chroot: Indica si el servicio se ejecuta en un directorio de acceso restringido.

Wakeup: Segundos que deben transcurrir para que el proceso master despierte el servicio.

Maxprocess: Número máximo de procesos que puede usar el servicio.

Command: Nombre del programa a ejecutar y parámetros a pasar.

Service.

Cada servicio de Postfix debe tener una entrada en el fichero *master.cf*, que son:

bounce: Devuelve al remitente los mensajes rechazados.

bsmtp: Encamina mensajes mediante el protocolo BSMTP.

cleanup: Procesa el correo entrante y soluciona posibles problemas en las direcciones de correo.

cyrus: Encamina correo mediante Cyrus Mail.

defer: Encamina mensajes fallidos o reintenta encaminar mensajes que estén en la cola *defer*.

error: Fuerza que un mail sea rechazado.

flush: Mantiene el control de los mensajes que están pendientes de encaminar.

ifmail: Encamina mensajes mediante ifmail.

lmtp: Encamina mensajes mediante el protocolo LMTP.

local: Encamina mensajes a usuarios locales.

pickup: Gestiona los mensajes que están esperando en la cola *incoming*.

qmgr: Procesa los mensajes que están en la cola *incoming* y decide cuál debe ser el método de encaminamiento.

relay:	Recibe y encamina mensajes mediante el protocolo SMTP.
rewrite:	Rescribe o verifica que las direcciones están en formato FQDN.
showq:	Proporciona información sobre el estado de las colas.
smtp:	Recibe y encamina mensajes mediante el protocolo SMTP.
uucp:	Recibe y encamina mensajes mediante el protocolo UUCP.

Type.

Especifica el mecanismo utilizado por el proceso para comunicarse con otros módulos, que puede ser de tres tipos:

- Internet sockets (inet)

- Unix sockets (unix)

- *Pipes* con nombre (fifo)

Private.

Indica cuándo el canal de comunicación de un proceso debe estar accesible a procesos ajenos a Postfix.

Postfix utiliza dos subdirectorios: *public* y *private*, donde se crean los *pipes* con nombre de cada uno de los servicios, en función de que sean públicos o privados.

Unprivileged.

Especifica con qué privilegio de usuario se ejecuta el servicio. Si se especifica **y** (que es el valor por omisión) el servicio se ejecuta con los del usuario descrito en la directiva **mail_owner** de *main.cf*, que por defecto es **postfix**.

Si se indica **n**, el servicio se ejecuta con privilegios de **root**.

Chroot.

Se indica que el servicio se ejecuta en un entorno *chroot*, lo que proporciona niveles adicionales de seguridad. Esta opción se activa indicando **y** en este campo.

Una única restricción: los procesos *local* y *pipe* no pueden ejecutarse en modo *chroot*.

Wakeup.

Indica los segundos que deben transcurrir para que el proceso master envíe una señal para despertar el servicio correspondiente.

En la actualidad sólo los servicios *pickup*, *qmgr* y *flush* utilizan esta directiva.

Existe una opción adicional, que es añadir el símbolo **?** al final del valor que señala el intervalo. Con esto se indica al proceso *master* que sólo envíe la señal de

despertar al servicio si este se está ejecutando. En la actualidad sólo *flush* soporta esta funcionalidad.

Maxprocess.

Especifica el número máximo de procesos que puede tener en ejecución el servicio. En caso de no indicarse nada se admiten 50 procesos.

Commands.

Determina el programa que debe ejecutarse para dar soporte al servicio definido.

Todos los programas admiten las siguientes opciones:

-v Habilita mayor detalle de log.

-D Activa el modo debug.

Un ejemplo de fichero *master.cf* sería:

```
#
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (50)
# ==========================================================================
#
#
smtp       inet n       -       n       -       -       smtpd
pickup     fifo n       -       n       60      1       pickup
cleanup    unix n       -       n       -       0       cleanup
#qmgr      fifo n       -       n       300     1       qmgr
qmgr       fifo n       -       n       300     1       nqmgr
tlsmgr     fifo -       -       n       300     1       tlsmgr
rewrite    unix -       -       n       -       -       trivial-rewrite
bounce     unix -       -       n       -       0       bounce
defer      unix -       -       n       -       0       bounce
flush      unix n       -       n       1000?   0       flush
smtp       unix -       -       n       -       -       smtp
relay      unix -       -       n       -       -       smtp
showq      unix n       -       n       -       -       showq
error      unix -       -       n       -       -       error
local      unix -       n       n       -       -       local
virtual    unix -       n       n       -       -       virtual
lmtp       unix -       -       n       -       -       lmtp
#
#
cyrus      unix -       n       n       -       -       pipe
  flags=R user=cyrus argv=/cyrus/bin/deliver -e -m ${extension} ${user}
```

```
uucp       unix -       n       n       -       -       pipe
  flags=Fqhu user=uucp argv=uux -r -n -z -a$sender - $nexthop!rmail ($recipient)
ifmail     unix  -       n       n       -       -       pipe
  flags=F user=ftn argv=/usr/lib/ifmail/ifmail -r $nexthop ($recipient)
bsmtp      unix  -       n       n       -       -       pipe
  flags=Fq. user=foo argv=/usr/local/sbin/bsmtp -f $sender $nexthop $recipient
```

2.- Configuración.

Dentro de la arquitectura del soporte informático para la gestión de correo electrónico de una organización de nivel medio y/o grande, es habitual disponer de un sistema que gestione todo el tráfico entrante y saliente, aplicando cuantos mecanismos de seguridad y gestión se consideren oportunos. Este sistema es lo que se denomina una estafeta de primer nivel.

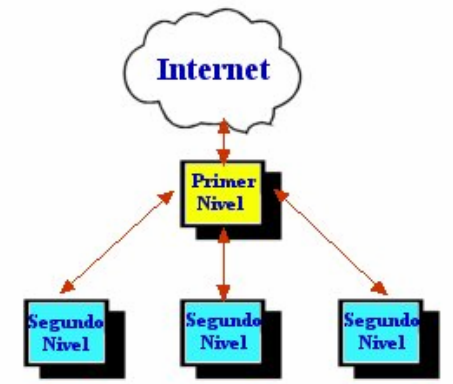

Figura 2

Una estafeta de primer nivel no debe proporcionar servicio directo a usuarios salvo, claro está, los estrictamente necesarios por naturaleza: *root*, *postmaster*, *abuse*, etc.

A un segundo nivel, y de ahí su nombre, se encuentran tantas estafetas como considere necesaria la institución, a través de las cuales los usuarios acceden al servicio de correo electrónico.

Con este planteamiento, la estafeta de primer nivel se encarga de recibir y encaminar el tráfico de correo electrónico hacia el exterior de la organización o hacia las estafetas de segundo nivel, que son quienes realmente proporcionan el servicio a los usuarios.

En una configuración de este tipo sólo la estafeta de primer nivel debe dialogar con el exterior, por lo que es más que aconsejable proceder a filtrar el puerto 25 (SMTP) al resto de sistemas de la red. De esta manera evitaremos problemas derivados de un uso incorrecto del correo electrónico, ya sea desde el exterior como del interior.

En organizaciones de tipo pequeño, puede ser razonable no adoptar este esquema a dos niveles, por lo que existirá un único sistema cuyo comportamiento será un híbrido entre ambos niveles.

2.1.- Estafeta de Primer Nivel.

Supongamos que se quiere configurar una estafeta de primer nivel para una organización de tipo medio, que gestionará el tráfico de correo electrónico para su dominio y algunos subdominios:

univ.es

depart1.univ.es

depart2.univ.es

Esto implica que bajo la estafeta de primer nivel existirán tres estafetas de segundo nivel, que darán servicio a los usuarios de:

@univ.es

@depart1.univ.es

@depart2.univ.es

Se trabajará sobre el fichero modelo de *main.cf* que se presentó anteriormente, haciéndose únicamente mención de aquellos parámetros que influyen en la configuración de una estafeta de primer nivel.

En primer lugar se definirán los parámetros básicos de la configuración:

```
# NOMBRE, DOMINIO y SMARTHOST
myhostname = relay.univ.es
mydomain = univ.es
#relay =
```

En este caso, el sistema a configurar es el propio *smarthost* (*relay*), por lo que no se definirá la variable **relay**.

```
# OTROS PARAMETROS DE CONFIGURACION
transport_maps = hash:/etc/postfix/transport
disable_dns_lookups = no
#relayhost = [$relay]
```

Dadas las peculiaridades de la estafeta que se está configurando, la tabla *transport* jugará un papel de vital importancia, por lo que se analizará posteriormente.

Evidentemente, dado que este sistema es quien debe dialogar con otros sistemas externos a la organización, deberemos hacer uso de los registros *Mail Exchange* (MX), aunque como se verá en la tabla *transport*, las comunicaciones internas se realizarán sin emplearlos. Es por ello que se indica **no** en el parámetro **disable_dns_lookups**.

Ya que no hay *smarthost*, tampoco se definirá el parámetro **relayhost**.

```
# CONTROL DE CORREO ENTRANTE / SALIENTE
mynetworks = 127.0.0.1 192.168.0.0/16
mydestination = $myhostname localhost.$mydomain
relay_domains = $mydomain
smtpd_recipient_restrictions = permit_mynetworks reject_unauth_destination
```

Con el parámetro **mynetworks** se define el rango de direcciones IP que conforman la Intranet de la organización especificado en formato Classless Internet Domain Routing (CIDR).

En **mydestination** se declaran todos los dominios que deben considerarse locales al sistema a efectos de encaminamiento del correo electrónico. En este caso **univ.es** no se incluye en esta relación, pues será otro sistema el que atienda esta dirección.

En **relay_domains** se declaran lo dominios (incluyendo todos los posibles subdominios) para los que se autorizará usar este sistema como *relay*. En esta versión de Postfix no es necesario indicar los valores por omisión ($mydestination), por lo que sólo se incluye **univ.es** (y todos sus subdominios).

Con el parámetro:

smtpd_recipient_restrictions = permit_mynetworks reject_unauth_destination

se está indicando a Postfix que admita sólo el correo que cumpla alguna de las siguientes condiciones:

1.- Procedente de clientes cuya dirección IP esté incluida en *mynetworks*.

2.- Procedente de cualquier sistema, cuyo destinatario esté incluido en *relay_domains*, *inet_interfaces*, *mydestination*, *virtual_alias_domains*, *virtual_mailbox_domains*.

que correspondería con el comportamiento deseado.

Un dato importante a tener siempre en cuenta a la hora de modificar el parámetro **smtpd_recipient_restrictions** es que las restricciones se analizan secuencialmente, y se detiene el proceso cuando una concuerda. En ocasiones esto se olvida y conduce a comportamientos supuestamente anómalos.

Ahora sólo falta definir el contenido de la tabla *transport*:

```
# TRAFICO LOCAL ...
relay.univ.es          local:
localhost.univ.es      local:

# TRAFICO INTERNO ...
univ.es                smtp:[smtp.univ.es]
depart1.univ.es        smtp:[smtp.depart1.univ.es]
.depart1.univ.es       smtp:[smtp.depart1.univ.es]
depart2.univ.es        smtp:[smtp.depart2.univ.es]
.depart2.univ.es       smtp:[smtp.depart2.univ.es]
```

Todos los mensajes cuyo destinatario no concuerde con alguna de las entradas definidas en esta tabla se encaminarán mediante protocolo SMTP y utilizando las entradas MX del DNS si las hubiera.

Sólo parece necesario destacar que el encerrar el nombre del sistema destino entre corchetes fuerza a un encaminamiento estático, pues no se hará uso de los registros MX.

Las entradas con un punto (.depart1.univ.es, por ejemplo) se han puesto para incluir cualquier posible subdominio que pudiera existir. En el caso de tener certeza de que no existe subdominio alguno se pueden quitar las respectivas entradas de la tabla.

2.2.- Estafeta de Segundo Nivel.

En este caso, la estafeta a configurar admitirá tráfico de la estafeta de primer nivel, así como de otras estafetas que pudieran existir dentro de la organización.

Dado que el tráfico es interno, lo habitual es que se utilice encaminamiento estático, es decir, no utilizando las entradas de registros MX del DNS, que es la política que seguiremos.

Por otra parte, todo el tráfico cuyo destino sea externo a la organización deberá encaminarse a través del *smarthost* obligatoriamente.

Veamos cómo se configuraría la estafeta que proporciona servicio al dominio **univ.es**.

Se definirán los parámetros básicos de la configuración:

```
# NOMBRE, DOMINIO y SMARTHOST
myhostname = smtp.univ.es
mydomain = univ.es
relay = relay.univ.es
```

En este caso sí que existe un sistema que hará de *smarthost*, por lo que será necesario definir la variable **relay** indicando el nombre del sistema que realiza tal labor.

```
# DIRECCION QUE APARECE EN ENVIO DE CORREO
myorigin  = $mydomain
```

Es lógico pretender que en el correo que se genere en esta estafeta aparezca como sistema origen **univ.es** y no **smtp.univ.es**. Para ello se utiliza el parámetro **myorigin**.

```
# OTROS PARAMETROS DE CONFIGURACION
transport_maps = hash:/etc/postfix/transport
disable_dns_lookups = yes
relayhost = [$relay]
```

Se deshabilitará el uso de los registros MX del DNS, por lo que se deberá indicar **yes** en el parámetro **disable_dns_lookups**.

También será necesario indicar a Postfix que todo el tráfico cuyo encaminamiento no pueda resolverse mediante la tabla *transport* debe enviarse a otra estafeta, que será la encargada de gestionarlo. Para ello se especificará el parámetro **relayhost** apuntando al *smarthost* de la organización. Aunque pueda parecer redundante, es recomendable que se indique el encaminamiento estático mediante el uso de corchetes, como puede verse en el ejemplo.

```
# CONTROL DE CORREO ENTRANTE / SALIENTE
mynetworks = 127.0.0.1 192.168.0.0/16
mydestination = $myhostname localhost.$mydomain $mydomain
smtpd_recipient_restrictions = permit_mynetworks reject_unauth_destination
```

Al igual que con la estafeta de primer nivel, con el parámetro **mynetworks** se define el rango de direcciones IP que conforman la Intranet de la organización.

Con el parámetro **mydestination** se declaran todos los dominios que se consideran locales al sistema a efectos de encaminamiento del correo electrónico. En este caso el destino a **univ.es** es local, por lo que está incluido.

Con el parámetro:

smtpd_recipient_restrictions = permit_mynetworks reject_unauth_destination

se actuará de igual forma que en las estafetas de primer nivel, indicando a Postfix que admita sólo el correo que cumpla alguna de las condiciones que se han expuesto anteriormente.

Sólo queda definir el contenido de la tabla *transport*:

```
# TRAFICO LOCAL ...
univ.es                 local:
smtp.univ.es            local:
localhost.univ.es       local:

# TRAFICO INTERNO ...
.univ.es                smtp:
```

Como se ha comentado, todos los mensajes cuyo destinatario no concuerde con alguna de las entradas definidas en esta tabla se encaminarán al *smarthost* directamente, y será éste el responsable de su encaminamiento.

Es importante destacar que el tráfico con destino a **univ.es** se trata como local, y el resto del tráfico con destino a otros sistemas de la Intranet se encamina directamente, vía SMTP, a través de la entrada **.univ.es**, pues de no existir ésta, se encaminaría a través del *smarthost*, lo que no sería deseable.

3.- Soporte de múltiples dominios.

Lo que se plantea en la primera parte de este apartado es analizar la forma de configurar Postfix para que actúe como *smarthost* de otros dominios y/o subdominios, usando para ello lo que Postfix denomina *sendmail-style virual domains*.

Suponga que se tiene el servidor de correo del dominio **univ.es**, y es necesario que este mismo sistema temporalmente acoja el dominio **fund-univ.es**, compartiendo ambos dominios los mismos usuarios, que por tanto serán únicos y cualquiera podrá recibir mensajes en una dirección u otra indistintamente. Transcurrido un cierto tiempo, parte de los usuarios migrarán su correo al sistema que definitivamente de soporte a **fund-univ.es**.

Es vital definir en el DNS un MX de **fund-univ.es** apuntando a **relay.univ.es**, así como que el *smarthost* encamine todo el tráfico de **fund-univ.es** a **smtp.univ.es**. Para ello, se modificará el fichero *main.cf* de **relay.univ.es** de la siguiente manera:

```
# CONTROL DE CORREO ENTRANTE / SALIENTE
mydestination = $myhostname localhost.$mydomain
relay_domains = $mydomain fund-univ.es
```

Y su correspondiente fichero *transport* quedará de la siguiente manera:

```
# TRAFICO LOCAL ...
relay.univ.es          local:
localhost.univ.es      local:

# TRAFICO INTERNO ...
univ.es                smtp:[smtp.univ.es]
fund-univ.es           smtp:[smtp.univ.es]
depart1.univ.es        smtp:[smtp.depart1.univ.es]
.depart1.univ.es       smtp:[smtp.depart1.univ.es]
depart2.univ.es        smtp:[smtp.depart2.univ.es]
.depart2.univ.es       smtp:[smtp.depart2.univ.es]
```

De esta manera se ha conseguido que *relay* encamine todo el correo con destino **fund-univ.es** a **smtp.univ.es**.

Por otra parte, será necesario modificar la configuración de la estafeta de segundo nivel para que admita el nuevo tráfico, y para ello se configurará el fichero *main.cf* de **smtp.univ.es** poniendo:

```
# CONTROL DE CORREO ENTRANTE / SALIENTE
mydestination = $myhostname localhost.$mydomain $mydomain fund-univ.es
```

El fichero *transport* podría quedar configurado de la siguiente manera:

```
# TRAFICO LOCAL ...
univ.es                 local:
smtp.univ.es            local:
localhost.univ.es       local:
fund-univ.es            local:

# TRAFICO INTERNO ...
.univ.es                smtp:
```

De esta manera, todos los mensajes con destino a **fund-univ.es** serán tratados como locales, y por tanto sus usuarios serán validados igualmente.

El problema fundamental en una configuración de este tipo es que no hay manera de separar grupos de usuarios locales según el dominio.

Postfix propone también otra forma de tratar la gestión de dominios virtuales, que denomina *postfix-style virtual domains*, en donde cada dominio virtual tiene su propio espacio de nombres de usuarios, independiente de los usuarios locales del sistema que lo soporta.

Usando este otro método, el fichero *main.cf* de **smtp.univ.es** quedaría:

```
# CONTROL DE CORREO ENTRANTE / SALIENTE
mydestination = $myhostname localhost.$mydomain $mydomain
```

Es decir, no es necesario hacer mención alguna al dominio **fund-univ.es** en el fichero de configuración. Al igual que ocurre con el fichero *transport*:

```
# TRAFICO LOCAL ...
univ.es                 local:
smtp.univ.es            local:
localhost.univ.es       local:

# TRAFICO INTERNO ...
.univ.es                smtp:
```

En el fichero *main.cf* es necesario añadir las directivas:

```
# CONTROL DE CORREO ENTRANTE / SALIENTE
virtual_alias_domains = fund-univ.es
virtual_alias_maps = hash:/etc/postfix/virtual
```

Y será necesario crear el fichero (tabla) ***/etc/postfix/virtual***, en donde se definen las relaciones entre usuarios virtuales y usuarios reales. Un ejemplo sería:

```
# /etc/postfix/virtual
gerente@fund-univ.es     rgarcia
gestion@fund-univ.es     laurar
tesorero@fund-univ.es    jheras@pastorro.com
gramos@fund-univ.es      gramos
```

Los dominios virtuales de Postfix tienen su propio espacio de nombres de usuarios, por lo que los usuarios locales no son visibles desde un dominio virtual. Será necesario especificar todos y cada uno de los usuarios de los dominios virtuales en el fichero *virtual*.

El acceso a esta tabla se basa en un fichero indexado, por lo que es necesario no olvidar usar la directiva **postmap /etc/postfix/virtual** cada vez que se modifique.

Siguiendo con el ejemplo, en el momento en que se pusiera en marcha el sistema **smtp.fund-univ.es**, la configuración de **smtp.univ.es** volvería a su estado inicial, y solo sería necesario modificar la configuración de **relay.univ.es**, en concreto el fichero *transport*:

```
# TRAFICO LOCAL ...
relay.univ.es          local:
localhost.univ.es      local:

# TRAFICO INTERNO ...
univ.es                smtp:[smtp.univ.es]
fund-univ.es           smtp:[smtp.fund-univ.es]
depart1.univ.es        smtp:[smtp.depart1.univ.es]
.depart1.univ.es       smtp:[smtp.depart1.univ.es]
depart2.univ.es        smtp:[smtp.depart2.univ.es]
.depart2.univ.es       smtp:[smtp.depart2.univ.es]
```

De esta manera, todos los mensajes con destino a **fund-univ.es** serán enviados desde el *relay* a **smtp.fund-univ.es** directamente, sin hacer uso de los registros MX.

Debe tenerse en mente que es importantísimo que exista en el DNS un MX de **fund-univ.es** apuntando a **relay.univ.es**.

Si existiesen más dominios y/o subdominios a los que fuera necesario realizar funciones de *relay* se procedería a configurarlos de igual forma.

4.- Soporte de Transportation Layer Security (TLS).

Habitualmente las comunicaciones mediante protocolo SMTP (Simple Mail Transport Protocol) se realizan sin utilizar mecanismos de cifrado, por lo que toda la información viaja en claro por Internet, lo que para cierto tipo de usuarios implica invalidar el uso de este medio.

En 1999, con aplicación no sólo a SMTP, se definió el protocolo TLS (Transportation Layer Security) basado en SSL (Secure Socket Layers), y cuya definición formal puede encontrarse en el *RFC-2246*. TLS básicamente proporciona cifrado en las comunicaciones y autentificación entre ambos corresponsales mediante el uso de certificados X.509.

La integración del protocolo TLS y SMTP se define en el *RFC-2487*, y se implementa en ESMTP (Extended Simple Mail Transport Protocol), en concreto en la negociación inicial (EHLO). El servidor ofrece la prestación de TLS mediante la opción STARTTLS, invitando al cliente a enviar la directiva STARTTLS y pasar a un estado de comunicaciones cifradas.

Hasta la fecha, las versiones estables de Postfix no proporcionan directamente soporte de TLS, aunque desde hace tiempo se dispone de las modificaciones necesarias, gracias a *Lutz Jänicke* que las ha creado y mantiene de manera impecable. En la página oficial de Postfix, en la sección *Add-on Software* se encuentra el enlace a su página.

Para saber si un servidor esta ofreciendo servicio TLS se puede establecer una conexión al puerto SMTP (25) e iniciar el dialogo:

```
$ telnet ca.fi.upm.es smtp
Trying 138.100.8.30...
Connected to ca.fi.upm.es.
Escape character is '^]'.
220 ca.fi.upm.es ESMTP Postfix-TLS-SASL (2.0.0.1)
ehlo simbad.fi.upm.es
250-ca.fi.upm.es
250-PIPELINING
250-SIZE 10240000
250-ETRN
250-STARTTLS
250-AUTH LOGIN PLAIN DIGEST-MD5 CRAM-MD5
250-AUTH=LOGIN PLAIN DIGEST-MD5 CRAM-MD5
250-XVERP
250 8BITMIME
```

Como puede apreciarse, una vez realizada la identificación inicial mediante la directiva EHLO, el servidor proporciona la relación de funciones que soporta, entre las que aparece el protocolo TLS mediante la opción STARTTLS.

4.1.- Compilación.

En la página de *Lutz Jänicke* se proporciona sobrada información de cómo aplicar las modificaciones a los fuentes de Postfix.

Es muy importante prestar atención a utilizar siempre las modificaciones y versiones adecuadas. Para ello *Lutz* mantiene una tabla en su página web en la que especifica la versión de parche (PFIXTLS), Potsfix y OpenSSL que debe utilizarse. Por ejemplo:

PFIXTLS	Postfix	OpenSSL	Date
0.8.12	postfix-2.0.0.1	0.9.6h	24 Dec 2002

Si se respeta esto no habrá problema en la compilación.

Previamente, y para borrar cualquier resto de compilaciones anteriores use la directiva:

```
$ make tidy
```

Una vez aplicado el parche habrá que compilar Postfix. Para ello, situados en el directorio de Postfix, y antes de usar la directiva **make** hay que generar los ficheros **Makefile** con las opciones adecuadas.

Para ello se utilizará la directiva:

```
$ make makefiles CCARGS="-DHAS_SSL -I/usr/local/ssl/include" \
AUXLIBS="-L/usr/local/ssl/lib -lssl -lcrypto"
```

Siendo **/usr/local/ssl** el directorio donde se ha instalado OpenSSL.

Una vez generados los ficheros Makefile se procederá a continuar con el proceso normal de compilación e instalación que indica Postfix en su documentación.

4.2.- Configuración.

Se recomienda que antes de proceder a configurar y activar el soporte de TLS, se verifique que Postfix esta funcionando sin problema alguno. No es recomendable incrementar los niveles de complejidad innecesariamente.

Como anteriormente se ha comentado, TLS se basa en el uso de certificados X.509, por lo que será necesario disponer de una clave privada y de un certificado firmado por alguna CA (Autoridad de Certificación):

- ✓ Certificado de la CA.
- ✓ Certificado del Servidor.
- ✓ Clave Privada del Servidor.

Normalmente, y evidentemente por razones de seguridad, la clave privada siempre se almacena protegida por una frase de acceso. Esto implica que cada vez que se accede a dicha clave privada será necesario proporcionar la frase.

En un servidor de correo esta circunstancia puede ser un inconveniente, por lo que será necesario disponer la clave privada sin protección de frase de acceso. Como contrapartida, es necesario tener un cuidado exquisito a la hora de configurar las protecciones de directorio y ficheros relacionados.

Suponiendo que la clave privada se encuentra en formato PEM en el fichero **keycon.pem**, con la directiva:

/usr/local/ssl/bin/openssl rsa -inform pem -in keycon.pem -text -out keysin.pem

se obtendrá en el fichero **keysin.pem** la clave privada sin frase de protección. Será este fichero el que se proporcionará a Postfix cuando se indique la clave privada del servidor.

Por otra parte, Postfix diferencia la faceta cliente y servidor, por lo que es necesario activar el soporte de TLS en la configuración para la parte cliente y para la parte servidor de manera separada. Las directivas que comienzan por "**smtp_**" corresponden a la faceta cliente, mientras que los que empiezan por "**smtpd_**" son los relativos a la faceta servidor.

En base a lo visto, una configuración básica de Postfix para soporte TLS sería:

```
# CONFIGURACION TLS
# Cliente
smtp_use_tls = yes
smtp_tls_session_cache_database = sdbm:/etc/postfix/smtp_scache
smtp_tls_key_file = /etc/postfix/cert/keysin.pem
smtp_tls_cert_file = /etc/postfix/cert/cert.pem
smtp_tls_CAfile = /etc/postfix/cert/CertCA.pem
# Servidor
smtpd_use_tls = yes
smtpd_tls_session_cache_database = sdbm:/etc/postfix/smtpd_scache
smtpd_tls_key_file = /etc/postfix/cert/keysin.pem
smtpd_tls_cert_file = /etc/postfix/cert/cert.pem
smtpd_tls_CAfile = /etc/postfix/cert/CertCA.pem
```

De esta manera, Postfix ofertará como servidor la posibilidad de establecer comunicación cifrada mediante la opción STARTTLS. Al mismo tiempo, y como

cliente, intentará establecer comunicación cifrada con todo servidor que le ofrezca dicha posibilidad.

Como puede apreciarse, los certificados y clave privada se han ubicado en el directorio **/etc/postfix/cert**:

```
drwxr-x---    2 root     root         4096 jun 29 20:46 /etc/postfix/cert
```

Las protecciones aplicadas a los ficheros son:

```
-rw-------    1 root     root         3877 jun 29 20:44 CertCA.pem
-rw-------    1 root     root         4200 jun 26 13:32 cert.pem
-rw-------    1 root     root         3640 jun 26 13:32 keysin.pem
```

Aunque el único con el que hay que tener especial cuidado es **keysin.pem**, pues los otros son públicos: Certificado de CA y Certificado de Servidor.

En el caso de que se establezcan conexiones con servidores cuyos certificados estén firmados por otras CA, sólo tendremos que concatenar en el fichero CertCA.pem los certificados de las respectivas CA.

5.- Soporte de Simple Authentication and Security Layer (SASL).

SASL (Simple Authentication and Security Layer) es una capa software que permite añadir soporte de autentificación en protocolos orientados a conexión, como puede ser SMTP.

Es habitual encontrar administradores de servidores de correo que tienen grandes problemas con usuarios itinerantes, o que por cualquier causa usan direcciones IP diferentes en cada conexión.

Una solución clásica es añadir, de una manera más o menos sofisticada (pop-before-smtp por ejemplo) la dirección IP asignada en ese instante a la relación de direcciones IP a las que se permite usar el servidor como *relay*.

Es razonable pensar que la solución al problema de itinerancia pasa por ofrecer un interfaz web para el acceso al correo. Solución que puede ser válida para determinadas circunstancias y para usuarios que manejan bajo volumen de mensajes, pero generalmente rechazada por los usuarios, acostumbrados a mayores y mejores prestaciones proporcionadas por otras herramientas no basadas en interfaz web.

SMTP AUTH es una solución distinta a las anteriores, donde el cliente se identifica mediante un **Usuario** y **Password** ante el servidor SMTP, y si ésta se realiza correctamente se autoriza el uso del servidor como *relay*. La autorización se hace a la persona y no al sistema informático que en ese momento le proporciona soporte.

Incluso se pueden llegar a establecer grupos de servidores de correo que utilicen SMTP AUTH para las conexiones entre ellos, como podrá verse posteriormente.

Postfix dispone de soporte de SMTP AUTH mediante el uso de las librerías Cyrus SASL. La nueva versión de Postfix 2.0 parece trabajar correctamente tanto con la versión 1.5.27 como con la 2.1.1.

5.1.- Compilación.

Aunque en este apartado se proporcionarán los pasos básicos para poder compilar Postfix con soporte SASL, es muy recomendable leer detenidamente el fichero SASL_README que encontrará en el directorio README_FILES de la distribución de Postfix.

En el caso que deba instalar SASL en su sistema, se recomienda ejecutar la directiva *configure* de SASL previo a la compilación con las siguientes opciones:

```
$ ./configure --enable-login --enable-plain --enable-cram --enable-digest \
--disable-krb4 --disable-gssapi --disable-anon --with-dblib=berkeley
```

Muchas de las opciones toman por omisión el valor indicado, pero de esta manera no queda lugar a dudas.

Analizando la directiva puede verse que se está configurando SASL para que soporte los mecanismos de autentificación LOGIN, PLAIN, CRAM y DIGEST.

Observe que en el caso de utilizar PLAIN o LOGIN, el *password* viajará en claro por la red, por lo que es más que aconsejable combinar el uso de SASL con TLS, pues una vez establecida la conexión cifrada entre cliente y servidor, da igual que el *password* viaje en claro o no.

No existe una forma única de realizar la compilación de Postfix debido, sobre todo, a la posible ubicación de las librerías empleadas, ficheros *include*, etc.

Como paso previo, y para borrar cualquier posible configuración anterior, ejecute la siguiente directiva en el directorio donde estén situados los fuentes de Postfix:

```
$ make tidy
```

El siguiente paso es generar los ficheros Makefile. Como ejemplo se presentan las directivas utilizadas para generar los mencionados ficheros en un entorno GNU/Linux y en Solaris 2.7. En ambos casos se incluye también soporte para TLS.

En GNU/Linux la directiva empleada sería:

```
$ make makefiles CCARGS="-DHAS_SSL -DUSE_SASL_AUTH \
-I/usr/local/ssl/include" AUXLIBS="-L/usr/local/ssl/lib -lssl -lcrypto -lsasl"
```

En Solaris 2.7 se usaría la siguiente directiva:

```
$ make makefiles CCARGS="-DHAS_SSL -DUSE_SASL_AUTH \
-I/usr/local/ssl/include -I/usr/local/include/sasl -I/usr/local/BerkeleyDB/include" \
AUXLIBS="-L/usr/local/lib -L/usr/local/ssl/lib -L/usr/local/lib/sasl \
-L/usr/local/BerkeleyDB/lib -lssl -lcrypto -lsasl -ldb"
```

Como puede apreciarse, en gran parte se depende de la instalación concreta de cada sistema.

En caso de utilizar la versión 2.1.1 de Cyrus SASL no debe olvidarse que tanto la librería como el directorio se denominan ***sasl2*** en vez de ***sasl***

Si se complican las cosas, puede ser aconsejable recurrir a algunas de las distribuciones ya compiladas para instalar que podrá encontrar en Internet, y que normalmente integran soporte TLS y SASL.

No es cometido de este documento analizar los posibles mecanismos de almacenamiento de claves que proporciona SASL, por lo que supondremos se utiliza el método propio de SASL: el fichero */etc/sasldb*.

Independientemente del método empleado, en el directorio */usr/lib/sasl* o en */usr/local/lib/sasl*, dependiendo de la instalación, deberá crearse el fichero *smtpd.conf*

```
-rw-r--r--    1 root     root           23 Jul  3 14:15 smtpd.conf
```

con el siguiente contenido:

```
pwcheck_method: sasldb
```

De esta manera estaremos indicando a SASL que para el proceso *smtpd* el método de verificación de *passwords* se utilizará *sasldb*.

Existen otras opciones, pero se recomienda utilizar la documentación de SASL.

5.2.- SMTP AUTH en todos los puertos SMTP.

En el primer supuesto, se entiende que quiere generalizarse el uso de SMTP AUTH, lo que permitirá que cualquiera que se identifique correctamente podrá utilizar la estafeta para gestionar su correo, independientemente de la dirección IP que esté utilizando en ese momento, o cualquier otra condición que quiera establecerse.

Para comenzar con la configuración, se editará el fichero *main.cf* y se pondrán las siguientes directivas:

```
# CONFIGURACION SASL
broken_sasl_auth_clients    = yes
smtpd_sasl_local_domain     = $myhostname
smtpd_sasl_auth_enable      = yes
```

La directiva **broken_sasl_auth_clients** se utiliza para dar soporte a antiguos clientes Microsoft que no soportan la versión estándar del protocolo AUTH.

Con la directiva **smtpd_sasl_local_domain** se define el dominio de búsqueda, que en terminología SASL se denomina **realm**. Este campo, junto con el nombre de usuario, es lo que se utilizará para realizar la búsqueda en el fichero */etc/sasldb*. Es por ello muy importante que cuando se creen los usuarios con las herramientas que proporciona SASL, de especifique correctamente el *realm*, pues de lo contrario puede que las búsquedas sean fallidas.

Por último, en este bloque de directivas aparece **smtpd_sasl_auth_enable**, que activa la utilización del protocolo SMTP AUTH.

En este momento sólo falta delimitar cuándo se autoriza la utilización de la estafeta a los usuarios que se identifiquen correctamente. Para ello, y también en el fichero *main.cf* se modificará la directiva **smtpd_recipient_restrictions** para que quede de la siguiente manera:

```
# CONTROL DE CORREO ENTRANTE / SALIENTE
smtpd_recipient_restrictions =
                      permit_sasl_authenticated
                      permit_mynetworks
                      reject_unauth_destination
```

Bajo esta nueva configuración la estafeta admitirá únicamente correo que cumpla alguna de las siguientes condiciones:

1.- Procedente de clientes cuya dirección IP esté incluida en *mynetworks*.

2.- Procedentes de clientes que se hayan identificado correctamente mediante SMTP AUTH.

3.- Procedente de cualquier sistema, cuyo destinatario esté incluido en *relay_domains*, *inet_interfaces*, *mydestination*, *virtual_alias_domains*, *virtual_mailbox_domains*.

Una vez realizadas las modificaciones en *main.cf* se hará que Postfix relea la nueva configuración mediante la directiva: **postfix reload**.

Si luego se establece una conexión *telnet* al puerto SMTP de la estafeta que se está configurando se obtendrá algo similar a:

```
$ telnet ca.fi.upm.es smtp
Trying 138.100.8.30...
Connected to ca.fi.upm.es.
Escape character is '^]'.
220 ca.fi.upm.es ESMTP Postfix-TLS-SASL (2.0.0.1)
ehlo simbad.fi.upm.es
250-ca.fi.upm.es
250-PIPELINING
250-SIZE 10240000
250-ETRN
250-STARTTLS
250-AUTH LOGIN PLAIN DIGEST-MD5 CRAM-MD5
250-AUTH=LOGIN PLAIN DIGEST-MD5 CRAM-MD5
250-XVERP
250 8BITMIME
```

donde como puede apreciarse se ofrece la opción AUTH LOGIN, y AUTH=LOGIN para los viejos clientes de Microsoft.

5.3.- SMTP AUTH en determinados puertos SMTP.

Tal y como se realizó la configuración anterior, Postfix aplica SMTP AUTH a todos los puertos por los que escuche el proceso *smtpd*.

Puede que el sistema que se esté configurando sea una estafeta de segundo nivel, que tenga el puerto SMTP (25) filtrado, pero que se tenga intención de abrir un puerto

opcional para que aquellos usuarios itinerantes que lo deseen puedan enviar sus mensajes normalmente, eso si, con la condición que se identifiquen previamente mediante SMTP AUTH. En este caso sólo habrá que activar SMTP AUTH para el puerto concreto.

La configuración del fichero *main.cf* deberá quedar:

```
# CONFIGURACION SASL
broken_sasl_auth_clients    = yes
smtpd_sasl_local_domain     = $mydomain
#smtpd_sasl_auth_enable      = yes
```

Es decir, se elimina/comenta la directiva **smtpd_sasl_auth_enable**.

La zona de *main.cf* de CONTROL DE CORREO ENTRANTE / SALIENTE quedaría exactamente igual. Y en este caso, será necesario modificar el fichero *master.cf* para incluir un nuevo puerto y activar el control de AUTH:

```
smtp    inet   n    -    n     -     -     smtpd
8025    inet   n    -    n     -     -     smtpd   -o smtpd_sasl_auth_enable=yes
```

Como puede apreciarse, se mantiene activo el servidor por el puerto SMTP (25), y en la segunda línea se activa el puerto 8025 (como ejemplo), que en este caso tiene como opción específica el tener activo SMTP AUTH.

De esta manera, las conexiones por el puerto SMTP funcionarán normalmente. Mientras que a las realizadas por el puerto 8025 se les permitirá el uso de la estafeta si se han identificado correctamente mediante SMTP AUTH, independientemente de la dirección IP de origen o la dirección de destino del mensaje.

Para verificar que Postfix se comporta conforme a lo previsto, se procederá a establecer una conexión al puerto SMTP e iniciar el diálogo:

```
$ telnet ca.fi.upm.es smtp
Trying 138.100.8.30...
Connected to ca.fi.upm.es.
Escape character is '^]'.
220 ca.fi.upm.es ESMTP Postfix-TLS-SASL (2.0.0.1)
ehlo simbad.fi.upm.es
250-ca.fi.upm.es
250-PIPELINING
250-SIZE 10240000
250-ETRN
250-STARTTLS
250-XVERP
250 8BITMIME
```

donde puede apreciarse que no se ofrece identificación por SMTP AUTH.

Si se establece la misma conexión por el puerto 8025:

```
$ telnet ca.fi.upm.es 8025
Trying 138.100.8.30...
Connected to ca.fi.upm.es.
Escape character is '^]'.
220 ca.fi.upm.es ESMTP Postfix-TLS-SASL (2.0.0.1)
ehlo simbad.fi.upm.es
250-ca.fi.upm.es
250-PIPELINING
250-SIZE 10240000
250-ETRN
250-STARTTLS
250-AUTH LOGIN PLAIN DIGEST-MD5 CRAM-MD5
250-AUTH=LOGIN PLAIN DIGEST-MD5 CRAM-MD5
250-XVERP
250 8BITMIME
```

En este caso sí se está ofreciendo la posibilidad de utilizar la identificación mediante SMTP AUTH.

5.4.- Postfix como cliente SMTP AUTH.

Como se ha comentado con anterioridad, mediante la configuración de la parte cliente SMTP de Postfix es posible crear grupos de servidores que realicen el intercambio de mensajes basándose en las garantías que ofrece la identificación mediante SMTP AUTH.

Para ello, en el fichero *main.cf* será necesario modificar la configuración para que quede:

```
# CONFIGURACION SASL
broken_sasl_auth_clients    = yes
smtpd_sasl_local_domain     = $mydomain
smtpd_sasl_auth_enable      = yes
smtp_sasl_password_maps     = hash:/etc/postfix/auth_passwd
smtp_sasl_auth_enable       = yes
```

Donde el fichero */etc/postfix/auth_passwd* tendrá el formato:

destino usuario:password

en el que **destino** será el nombre completo en el DNS del servidor al que se quiere conectar, y el binomio **usuario-password** corresponderán a un usuario y clave de acceso para identificarse mediante SASL. Es necesario dar de alta el usuario en el sistema servidor, y no olvidarse de generar el fichero *db* correspondiente mediante la directiva **postmap /etc/postfix/auth_passwd**.

Con esta configuración, cada vez que la estafeta vaya a enviar un mensaje a otra, revisará el fichero *auth_passwd*, y si el destino existe, intentará realizar la conexión identificándose mediante SMTP AUTH. En caso de no existir el destino en el fichero *auth_passwd* se procederá a establecer la conexión sin utilizar SMTP AUTH.

Es más que recomendable que en esquemas como éste se utilice simultáneamente soporte TLS para cifrar la totalidad de las comunicaciones. En el caso de no ser posible puede ser aconsejable activar las cláusulas de seguridad del soporte de SASL:

```
smtpd_sasl_security_options     =  noplaintext
smtp_sasl_security_options      =  noplaintext
```

y de esta forma evitar que se realicen identificaciones donde el *password* viaje en claro por la red.

6.- Integración de filtros por contenido.

Dentro del término de *filtros por contenido* pueden englobarse multitud de herramientas y utilidades, aunque las más comunes son programas antivirus y antispam.

Para este cometido Postfix dispone de la directiva **content_filter**, que permite definir para los procesos de entrada de correo (*smtpd* y *pickup*) qué filtro debe aplicarse.

Esta directiva puede definirse en el fichero *main.cf*, por lo que se aplicará a ambos procesos, o bien como opción de los mencionados procesos en el fichero *master.cf*.

Dependiendo de las características del programa de filtro a utilizar podrá integrarse de una manera más sencilla, aunque menos eficiente, o de forma más compleja pero también más eficiente.

Puede que en ocasiones sea necesario integrar más de un filtro, lo que suele llevar a soluciones más complejas y, en general, menos elegantes. Quizás en este caso la mejor opción sea disponer de un único filtro capaz de hacer múltiples tareas, como puede ser el recientemente aparecido *amavisd* (*http://www.amavis.org/*).

6.1.- Solución simple.

Suponga que se dispone de un programa que realiza ciertas labores de análisis sobre ficheros, y cuya entrada y salida se hace de manera estándar. Es probable que incluso sea necesario integrar el uso de este programa dentro de un *script* que permita acondicionar la información de entrada y salida.

La idea base es que Postfix, mediante un *pipe* pase la totalidad del mensaje al *script*, se realicen los tratamientos necesarios, y finalmente se encamine el mensaje como correo local a través de *pickup*.

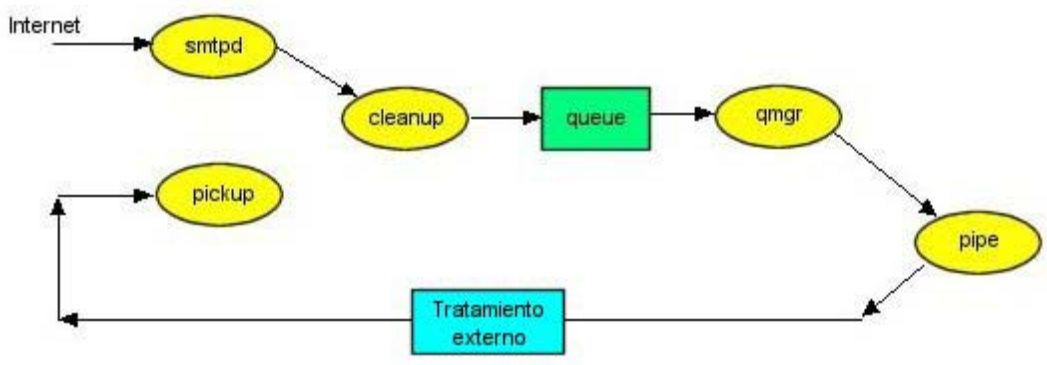

Figura 2

Un ejemplo claro de lo anterior sería *sweep*, el antivirus de Sophos.

El *script* que engloba todo el tratamiento externo podría ser similar a:

```
#!/bin/sh

# Definiciones
INSPECT_DIR=/var/spool/postfix/tmp
SENDMAIL="/usr/sbin/sendmail -i"

# Codigos de retorno <sysexits.h>
EX_TEMPFAIL=75
EX_UNAVAILABLE=69

# Comienzo del proceso
cd $INSPECT_DIR || { echo "\n $INSPECT_DIR no existe"; exit $EX_TEMPFAIL; }

# Borra los ficheros cuando termine el proceso
trap "rm -fr in.$$ tmp" 0 1 2 3 15

# Guarda el mensaje en un fichero
cat >in.$$ || { echo "\n No se puede salvar el mensaje en un fichero"; exit $EX_
TEMPFAIL; }

# Descompone el mensaje en sus componentes
/usr/local/bin/ripmime -i in.$$ -d $INSPECT_DIR/tmp
# Analiza la presencia de virus
/usr/local/bin/sweep -ss $INSPECT_DIR/tmp/* || { echo "\n Mensaje rechazado por
contener virus"; exit $EX_UNAVAILABLE; }

# Si todo bien, se encamina el mensaje
$SENDMAIL "$@" <in.$$

exit $?
```

En este ejemplo el mensaje se pasa al *script* a través de un *pipe*, se deposita en un fichero temporal, se procesa con *ripmime* para obtener en ficheros independientes cada una de las partes del mensaje MIME, se analizan todos los ficheros temporales con *sweep* (antivirus de Sophos), y si no se detectan virus, el mensaje se vuelve a encaminar a través de *pickup* mediante *sendmail* (alias de Postfix).

Otra ejemplo de *script* podría ser el utilizado para la herramienta diseñada para intentar limitar correo SPAM: SpamAssassin.

```
#!/bin/bash
/usr/local/bin/spamc | /usr/sbin/sendmail -i "$@"
exit $?
```

En este caso, el mensaje se pasa al *script* a través de un *pipe*, lo procesa SpamAssassin, y mediante otro *pipe*, e igualmente a través de *sendmail*, se encamina a *pickup*.

Como puede apreciarse, el flujo base del mensaje en ambos ejemplos es idéntico, conforme a lo representado en la Figura-2.

A la hora de configurar Postfix para que admita este tipo de configuración hay que tener mucho cuidado en no provocar bucles por culpa del filtrado.

En este caso en concreto, en el fichero *main.cf* no habrá que definir la directiva **content_filter**. En el fichero *master.cf* habrá que definir para el proceso que atiende el puerto SMTP la opción **content_filter** y definir cómo debe iniciarse la ejecución del filtro en sí.

De esta manera, *master.cf* quedaría de la siguiente forma:

```
#
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (50)
# ==========================================================================
smtp       inet n       -       n       -       -       smtpd -o content_filter=filtro:
pickup     fifo n       -       n       60      1       pickup
.
.
.
filtro     unix -       n       n       -       -       pipe
  flags=Rq user=seguro argv=/etc/postfix/filtro -f ${sender} -- ${recipient}
```

Donde se entiende que existe un usuario denominado *seguro*, que es el propietario del proceso por motivos de seguridad, y que el *script* a ejecutar es */etc/postfix/filtro*.

Como puede apreciarse, se define **content_filter=filtro:** para el puerto *smtp* (25), mientras que para *pickup*, como no se ha definido *content_filter* en *main.cf*, no se aplicará filtro alguno.

El proceso *filtro* se ejecuta bajo la propiedad del usuario *seguro*, y la comunicación con el proceso se realiza mediante un *pipe*, según se especifica el la última parte de *master.cf*.

En el ejemplo anterior el filtro sólo se aplicará para el puerto smtp (25). En el caso de que la estafeta atienda tráfico SMTP por varios puertos, se podría disponer de una configuración similar, aunque de aspecto distinto: se definiría el filtro en *main.cf*, lo que implicará que se aplique a todos los puertos, y se quitará explícitamente para *pickup* en el fichero *master.cf* poniendo en blanco dicha opción.

Si no se quitase la definición del filtro para el proceso *pickup* se produciría un bucle infinito: *pickup* -> filtro -> *pickup* -> ...

De esta manera, *main.cf* quedaría:

```
# CONTROL DE CORREO ENTRANTE / SALIENTE
content_filter = filtro:
```

Y *master.cf* quedaría:

```
#
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (50)
# ==========================================================================
8025       inet n       -       n       -       -       smtpd
      -o smtpd_sasl_auth_enable=yes
smtp       inet n       -       n       -       -       smtpd
pickup     fifo n       -       n       60      1       pickup
      -o content_filter=
.
.
.
filtro     unix -       n       n       -       -       pipe
  flags=Rq user=seguro argv=/etc/postfix/filtro -f ${sender} -- ${recipient}
```

6.2.- Solución básica.

Existe una solución más eficiente, pues abre menos ficheros y/o *pipes*, aunque al mismo tiempo algo más compleja de implantar. Desgraciadamente sólo es aplicable si el filtro a utilizar es capaz de entregar el resultado del proceso mediante protocolo SMTP.

Un ejemplo claro y muy utilizado últimamente en las estafetas de correo es AMaViS.

Cuando se configura AMaViS, existe una opción específica para Postfix que permite definir el puerto por el que éste espera recibir el resultado. En el fichero */etc/amavis.conf*:

```
# postfix-specific
my $smtp_port = "10025";
```

De esta manera se le indica a AMaViS que la salida la entregue a Postfix mediante protocolo SMTP por el puerto 10025.

Suponiendo configurado AMaViS de esta manera, habrá que indicar a Postfix que aplique el filtro a todos los puertos *smtpd* y *pickup*. Para ello sólo será necesario definir el filtro de contenido en el fichero *main.cf*.

Por otra parte, en master.cf será necesario indicar a Postfix que escuche también por el puerto 10025 y definir el filtro en sí.

Será muy importante anular la definición del filtro de contenido en el puerto 10025, pues de lo contrario se produciría un bucle infinito: 10025 -> filtro -> 10025 -> ...

En la Figura-3 se muestra cuál sería el recorrido que seguiría un mensaje utilizando este esquema.

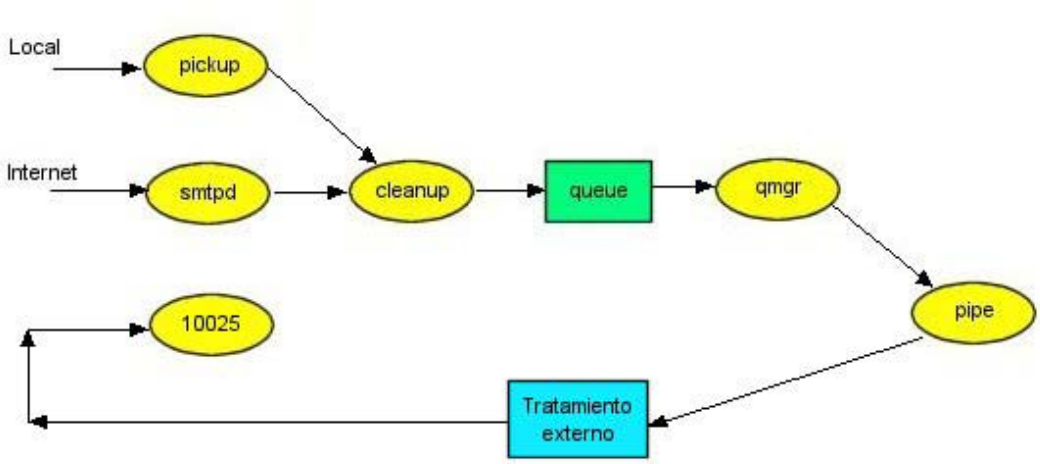

Figura 3

Como puede apreciarse, el filtro se aplicará en este caso tanto a los mensajes locales como externos, lo que en el modelo anterior no ocurría, pues sólo se aplicaba a los mensajes externos.

Por tanto, el fichero *main.cf* quedará de la siguiente forma:

```
# CONTROL DE CORREO ENTRANTE / SALIENTE
content_filter = filtro:
```

Y *master.cf* quedará:

```
#
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (50)
# ==========================================================================
10025      inet n       -       n       -       -       smtpd
      -o content_filter=
smtp       inet n       -       n       -       -       smtpd
pickup     fifo n       -       n       60      1       pickup
.
.
.
filtro      unix -       n       n       -      10       pipe
  flags=q user=vscan argv=/usr/sbin/amavis ${sender} ${recipient}
```

Como puede apreciarse, a través del puerto 10025 puede entrar correo desde el exterior sin que se aplique filtro alguno. Para hacer que sólo sea Postfix quien pueda utilizar dicho puerto, sería necesario modificar *master.cf* de la siguiente manera:

```
#
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (50)
# ==========================================================================
localhost:10025 inet    n       -       n       -       -       smtpd
      -o content_filter=
smtp       inet n       -       n       -       -       smtpd
pickup     fifo n       -       n       60      1       pickup
.
.
.
filtro      unix -       n       n       -      10       pipe
  flags=q user=vscan argv=/usr/sbin/amavis ${sender} ${recipient}
```

Es decir, en el puerto 10025 se declara explícitamente que sólo atiende conexiones por el interfaz localhost.

6.3.- Solución avanzada.

Es evidente que la solución ideal pasa por disponer de un filtro por contenido que sea capaz de recibir y entregar los mensajes a analizar utilizando el protocolo SMTP.

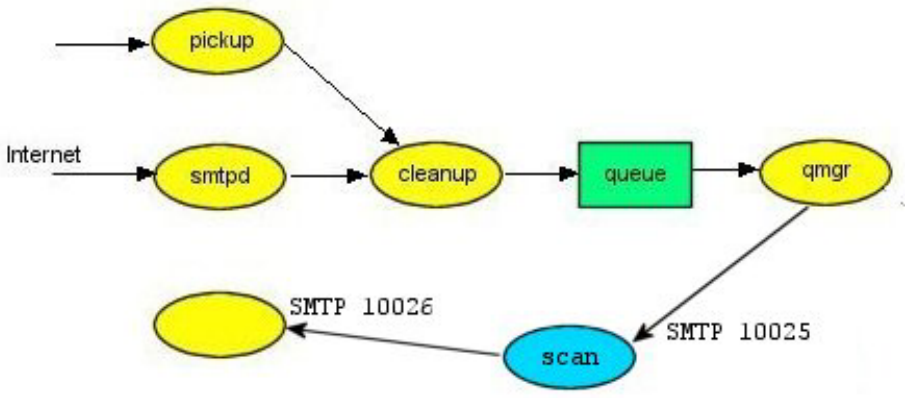

Figura 4

La versión 2.0 de Postfix aporta el mecanismo necesario que permite interactuar con filtros cuya entrada y salida se realice mediante SMTP.

Supongamos que se dispone, como puede verse en la Figura-4, de un programa de filtrado que se denomina *scan*, al que se le pasa los mensajes a analizar por el puerto 10025 mediante SMTP, y devolverá el resultado por el puerto 10026 también utilizando SMTP.

Se procederá a activar, por ejemplo, el filtro por contenido para todos los puertos que hablen SMTP, para lo que en *main.cf* se pondrá:

```
# CONTROL DE CORREO ENTRANTE / SALIENTE
content_filter = scan:localhost:10025
```

En el fichero *master.cf* se crearán los servicios correspondientes al proceso de entrega (*scan*) y recepción de los mensajes analizados:

```
#
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (50)
# ==========================================================================
localhost:10026 inet    n       -       n       -       -       smtpd
      -o content_filter=
smtp       inet n       -       n       -       -       smtpd
pickup     fifo n       -       n       60      1       pickup
.
.
.
scan       unix -       -       n       -      10       smtp
```

Sólo quedaría configurar el programa scan para que atienda por el puerto 10025, si es que éste se ejecuta como un demonio más en el sistema.

También es posible hacer que Postfix se responsabilice de activar la ejecución del programa de filtrado a través del servicio *spawn*, en cuyo caso el fichero *master.cf* quedaría de la siguiente forma:

```
#
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (50)
# ==========================================================================
localhost:10026 inet    n       -       n       -       -       smtpd
      -o content_filter=
smtp       inet n       -       n       -       -       smtpd
pickup     fifo n       -       n       60      1       pickup
.
.
.
localhost:10025 inet    n       n       n       -      10       spawn
      user=seguro argv=/path/scan localhost 10026
```

Donde *seguro* es un usuario desde el que no debe poderse hacer login, ni tener definido el directorio raiz ni shell.

7.- Mecanismos de seguridad.

Postfix, por propio diseño, proporciona niveles de seguridad en el servicio que no proporcionan otros gestores de correo.

Aún así, se han dispuesto una serie de directivas que permiten tener un mayor control de lo que ocurre, o especificar unos mayores niveles de exigencia para admitir un mensaje.

A continuación se presentan una serie de directivas y posibilidades que pueden ayudar a garantizar unos mayores niveles de calidad en el servicio de correo electrónico.

Todas estas directivas se definen en el fichero *main.cf*.

7.1.- Notificaciones a Postmaster.

Postfix puede notificar automáticamente, mediante un mensaje al usuario Postmaster, de aquellas incidencias que se han declarado de interés para su administración.

Mediante la directiva **notify_classes** el administrador indica a Postfix qué tipo de incidencias debe notificar, y que pueden ser:

bounce: Si un mensaje no puede ser encaminado, se envía otro mensaje al remitente y una copia al Postmater incluyendo el mensaje original. En el caso del Postmaster sólo se incluyen por razones de privacidad las cabeceras.

2bounce: En el caso en que la notificación de un mensaje de error de encaminamiento genere también un error de encaminamiento, se envía notificación a Postmaster.

delay: Se informa a Postmaster de que hay mensajes pospuestos por problemas en su encaminamiento.

policy: Se informa sobre peticiones rechazadas de entrega de mensajes. Normalmente se debe a que el interlocutor o el propio mensaje incumple la política definida para la aceptación de correo. Muy útil para conocer intentos fallidos de uso de la estafeta como relay.

protocol: Se informa sobre incidentes de protocolo.

resource: Se comunica a Postmaster que un mensaje no ha sido encaminado por problemas de recursos en el sistema.

software: Se comunica a Postmaster que un mensaje no ha sido encaminado por problemas de software.

Quizás en un primer momento se esté tentado en recibir todas las posibles notificaciones, aunque la experiencia demuestra que la configuración que al final suele adoptarse es:

notify_classes = resource, software, policy

7.2.- Rechazo por no resolución inversa.

Postfix permite delimitar los clientes a los que permitirá establecer una sesión SMTP. Para ello dispone de la directiva **smtpd_client_restrictions**, que entre otros criterios, permite rechazar las conexiones desde clientes cuya dirección IP no disponga de resolución inversa en el DNS.

La directiva a emplear será:

smtpd_client_restrictions = reject_unknown_client

La aplicación de esta medida ayuda a frenar la posible entrada de SPAM.

7.3.- Verificación de RBL.

Otro mecanismo de lucha contra el SPAM que en los últimos años está teniendo gran importancia son las Listas Negras (RBL – Realtime Blackhole List).

Postfix permite fácilmente verificar que el cliente que ha establecido la conexión SMTP o el remitente del mensaje no está inscrito en alguna de las listas negras que se designen.

En la directiva **smtpd_client_restrictions** activaremos el mecanismo de verificación:

smtpd_client_restrictions =

reject_rbl_client relays.ordb.org

reject_rhsbl_client relays.ordb.org

De esta forma, Postfix verificará que la traducción inversa de la dirección IP del cliente (reject_rbl_client) o el nombre de host del cliente (reject_rhsbl_client) no aparece inscrito en la RBL.

Este tipo de restricción también es aplicable en el comando MAIL FROM a través de la directiva **smtpd_sender_restrictions:**

smtpd_sender_restrictions =

reject_rhsbl_sender relays.ordb.org

7.4.- Requerimiento de HELO/EHLO y verificaciones.

Normalmente los sistemas de gestión de correo no obligan a iniciar el establecimiento de sesión mediante la directiva HELO o EHLO en caso de ESMTP. Es más, tampoco suelen proporcionar mecanismos que permitan analizar el nombre del sistema que está identificándose.

Postfix dispone de la directiva **smtpd_helo_required**, que dándole el valor **yes** obliga a que el sistema remoto inicie la sesión con la directiva HELO o EHLO. Por defecto este parámetro está a **no**.

Obligar a iniciar la sesión con HELO/EHLO ya en sí inutiliza algunas herramientas utilizadas para SPAM y algún que otro virus.

En conjunción con la anterior directiva se dispone también de la directiva **smtpd_helo_restrictions**, que permite establecer criterios que permiten continuar o no con la sesión. Aunque existen hasta doce criterios aplicables, la configuración básica recomendada es:

```
smtpd_helo_restrictions =
                      reject_invalid_hostname
                      reject_unknown_hostname
                      reject_non_fqdn_hostname
```

Mediante el criterio **reject_invalid_hostname** se rechazarán todas las sesiones en las que el nombre proporcionado en la directiva HELO/EHLO esté mal formado.

Con el criterio **reject_unknown_hostname** se rechazarán todas las sesiones en las que el nombre del sistema proporcionado no tenga una entrada A en un DNS o disponga de un registro MX.

Y por último, con el criterio **reject_non_fqdn_hostname** se rechazarán todas las sesiones en las que el nombre del sistema indicado en la directiva HELO/EHLO no esté en forma FQDN, es decir, que sea un nombre completo.

7.5.- Filtrado por cabeceras.

Un mecanismo que proporciona Postfix, y que se está utilizando ampliamente contra la propagación de virus, es la posibilidad de filtrar los mensajes en base a las cabeceras de los mismos y a patrones definidos mediante expresiones regulares.

De esta manera, todo mensaje que contenga una cabecera que cumpla un determinado patrón será rechazado automáticamente.

Un ejemplo de fichero de patrones podría ser:

```
#
# Headers Checks
#
# Virus: W32.SirCam [20/SEP/01]
/^date:.*/i                                             REJECT
/^Content-Disposition: Multipart message/               REJECT
#
# Virus Nimda (attach readme.exe) [20/SEP/01]
/^X-Unsent: 1/                                          REJECT
#
# Virus Badtrans [30/NOV/01]
/^From: ".*" <_/                                        REJECT
#
# Virus W32.Myparty.B@mm [29/ENE/02]
/^Subject: new photos from my party!/                   REJECT
#
#Virus W32/Frethem.K/J [15/JUL/02]
/^Subject: Re: Your password!/                          REJECT
#
```

Suponiendo que este fichero es **/etc/postfix/headers_checks**, para activar el filtrado por cabeceras será necesario incluir en el fichero *main.cf* la siguiente directiva:

header_checks = regexp:/etc/postfix/header_checks

No olvide hacer un *reload* de Postfix cada vez que modifique algún elemento de su configuración.

7.6.- Filtrado por contenido.

De igual manera que pueden filtrarse mensajes en base a patrones que se definen sobre las cabeceras de los mensajes, también se pueden definir patrones que se aplicarán al contenido de los mensajes.

```
#
# Body Checks
#
/Accept Credit Cards/            REJECT
/Nude Celebrities/               REJECT
/PRODUCT or SERVICE/             REJECT
/GUARANTEED!/                    REJECT
/Amateur Girls/                  REJECT
/FREE MEMBERSHIP/                REJECT
/bizinfo/                        REJECT
```

En este ejemplo se rechazan todos los mensajes que contengan en cualquier parte del cuerpo alguna de las palabras o frases indicadas.

Suponiendo que este fichero es **/etc/postfix/body_checks**, para activar el filtrado por contenidos en el mensaje será necesario incluir en el fichero *main.cf* la siguiente directiva:

body_checks = regexp:/etc/postfix/body_checks

7.7.- Limitación del tamaño de mensaje.

Limitar el tamaño de los mensajes que gestiona Postfix es tan simple como indicar el límite máximo en bytes, incluyendo cabeceras, mediante la directiva:

message_size_limit = 10240000

10 MB es el valor por defecto que asume Postfix.

Debe tenerse en consideración que si se incrementa mucho el **message_size_limit**, llegando a superar el valor de **mailbox_size_limit**, Postfix dará un error de configuración al arrancar:

fatal: main.cf configuration error: mailbox_size_limit is smaller than message_size_limit

El parámetro **mailbox_size_limit** controla el tamaño máximo del *mailbox* o del fichero *maildir*, y por defecto asume un valor de 51200000 bytes (50 MB).

Para deshabilitar cualquiera de estas dos límitaciones es suficiente con asignarles el valor cero con la respectiva directiva:

message_size_limit = 0

mailbox_size_limit = 0

en el fichero de configuración *main.cf*.

7.8.- Ejecución en modo chroot.

Postfix puede ejecutar la mayoría de sus procesos en modo *chroot*, es decir, los procesos sólo disponen de privilegios para acceder al directorio */var/spool/postfix*, y todas las referencias a ficheros se resuelven tomando este directorio como raíz.

No pueden ejecutarse en modo *chroot* el proceso *local*, *virtual* y todos los procesos *pipe*.

Aunque este no es un mecanismo que garantice la inviolabilidad, si que representa una barrera más, y por tanto es recomendable su adopción en instalaciones que requieran un nivel adicional de seguridad.

Dado que para cambiar a modo *chroot* es necesario copiar algunos ficheros, que cambiarán según el sistema operativo que se esté utilizando, Postfix proporciona en su distribución una colección de *scripts* para realizar esta labor.

A efectos de configuración, sólo es necesario modificar el fichero *master.cf* para indicar qué procesos deben ejecutarse en modo *chroot*.

Como ejemplo, ésta podría ser una configuración tipo trabajando en modo *chroot*:

```
#
# ==========================================================================
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (50)
# ==========================================================================
#
#
smtp       inet n       -       -       -       -       smtpd
pickup     fifo n       -       -       60      1       pickup
cleanup    unix n       -       -       -       0       cleanup
#qmgr      fifo n       -       -       300     1       qmgr
qmgr       fifo n       -       -       300     1       nqmgr
tlsmgr     fifo -       -       -       300     1       tlsmgr
rewrite    unix -       -       -       -       -       trivial-rewrite
bounce     unix -       -       -       -       0       bounce
defer      unix -       -       -       -       0       bounce
flush      unix n       -       -       1000?   0       flush
smtp       unix -       -       -       -       -       smtp
relay      unix -       -       -       -       -       smtp
showq      unix n       -       -       -       -       showq
error      unix -       -       -       -       -       error
local      unix -       n       n       -       -       local
virtual    unix -       n       n       -       -       virtual
lmtp       unix -       -       n       -       -       lmtp
#
#
cyrus      unix -       n       n       -       -       pipe
  flags=R user=cyrus argv=/cyrus/bin/deliver -e -m ${extension} ${user}
uucp       unix -       n       n       -       -       pipe
  flags=Fqhu user=uucp argv=uux -r -n -z -a$sender - $nexthop!rmail ($recipient)
ifmail     unix  -       n       n       -       -       pipe
  flags=F user=ftn argv=/usr/lib/ifmail/ifmail -r $nexthop ($recipient)
bsmtp      unix  -       n       n       -       -       pipe
  flags=Fq. user=foo argv=/usr/local/sbin/bsmtp -f $sender $nexthop $recipient
virus      unix  -       n       n       -       -       pipe
  flags=Rq user=postdrop argv=/etc/postfix/filtro -f ${sender} -- ${recipient}
```