

Seguridad en redes de comunicaciones

Instalación de una pasarela segura con
Debian Sarge

Francisco José Calvo Fernández
fran@glomedia.net

Versión 0.2.1

(c) Glomedia Consulting

Índice general

I Pasarela de correo con Postfix	3
1. Instalación del servidor de correo	4
1.1. Configuraciones previas	4
1.2. Paquetes	4
1.3. TLS	6
1.3.1. Creación de los certificados	6
1.3.2. Configuración de Postfix	7
1.3.3. Uso del canal seguro	8
1.4. Autenticación con SASL	8
1.4.1. Pruebas	11
2. Integración con Clamav	12
2.1. Instalación de Clamav	12
2.2. Instalación de Amavis-new	13
2.3. Configuración de Postfix	14
3. El Spam	16
3.1. Sistema básico de filtrado	16
3.1.1. Pasos a seguir	16
3.1.2. Filtrado por asunto	16
3.1.3. Filtrado por contenido	17
3.2. Apache SpamAssassin	17
3.3. Prueba del sistema	18
4. Sistema de entrega segura	20
4.1. IMAP seguro mediante Cyrus IMAP	20
4.1.1. Instalación y configuración básica	20
4.1.2. Conexión con Postfix	23
4.1.3. Configuración de los buzones de correo	25
4.1.4. Prueba	26
4.2. Instalación segura de Apache	27
4.3. Instalación del webmail	28
4.3.1. Sistema base	28

<i>ÍNDICE GENERAL</i>	2
II Proxy seguro con Squid	30
5. Squid	31
5.1. Instalación de Squid	31
5.2. Instalación de SquidClamAV	32

Parte I

Pasarela de correo con Postfix

Capítulo 1

Instalación del servidor de correo

1.1. Configuraciones previas

Es muy recomendable crear un usuario que reciba los avisos que genere el sistema de correo:

```
adduser acorreo
```

1.2. Paquetes

Para realizar una instalación básica de Postfix en Debian GNU/Linux Sarge hemos de instalar los siguientes paquetes:

```
apt-get install postfix postfix-doc postfix-pcre mime-codecs
```

Reconfiguramos el sistema:

```
dpkg-reconfigure postfix
```

Podremos responder con las siguientes respuestas:

1. En primer lugar se nos pregunta por el tipo de servicio que deseamos ofrecer, si somos un servidor que aceptará correo de internet, hemos de seleccionar *Internet Site*:

```
General type of configuration? Internet Site
```

2. Los correos que llegan al superusuario deberían de estar redirigidos a otro usuario sin más privilegios, ésto se configura mediante los *alias* que localizamos en */etc/aliases*, no obstante, se nos preguntará por ello, redirigimos al usuario antes creado.

Where should mail for root go? **acorreo**

3. El paso siguiente nos preguntará por el **Mail name**, se trata del dominio que figurará tras el usuario y la clásica *arroba* (@).

Mail name? **glomedia.net**

4. A continuación se nos pregunta por la lista de dominios que el servidor considerará como *suyos*, y en los que por tanto, atenderá el correo entrante. Los hosts han de estar separados por comas.

Other destinations to accept mail for? **glomedia.net, gmfcentral.glomedia.net, localhost.glomedia.net, localhost**

5. Se nos preguntará si deseamos trabajar en modo *síncrono*, en este modo el procesamiento del correo es más lento, pero ofrece la garantía de que no se perderán datos si el servidor *muere de forma repentina* (por ejemplo, un corte de luz). No obstante, si utiliza un sistema de archivos tipo *Journaling* (*ext3, reiserfs, xfs, jfs...*) en su Linux, esta tarea es redundante y no necesaria.

Force synchronous updates on mail queue? **No**

6. A continuación se nos pregunta por las redes en las que el servidor de correo podrá realizar *relay*, únicamente hemos de introducir la red *loopack*:

Local networks? **127.0.0.0/8**

Tenga en cuenta que si desea utilizar Postfix como servidor de correo **saliente** desde otras máquinas, será necesario dar permiso a la red desde donde se desea enviar el correo, también puede fijar una simple dirección IP. Si desea fijar más de una IP o dirección de red puede introducirlas separadas por espacios en blanco.

7. Se nos pregunta por el límite de los buzones de los usuarios, si seleccionamos 0 no existirá límite.

Mailbox size limit? **0**

8. El último paso nos pregunta por el carácter que define una extensión a las direcciones locales, lo dejamos tal cual por defecto.

Local address extension character? **+**

Toda la configuración se almacena en ficheros de texto que se encuentran en el directorio */etc/postfix*, resaltando **main.cf** y **master.cf**, en cualquier momento podremos abrirlos (cuando añada las líneas al archivo */etc/postfix/master.cf*, asegúrese de que al final de las líneas no existan espacios en blanco) y personalizarlos a nuestro gusto o volver a lanzar el asistente.

Ahora podremos instalar algún cliente de correo y probar si nuestro Postfix funciona de forma correcta. Si desea enviar directamente usando Telnet:

```
telnet localhost 25
  Trying 127.0.0.1...
  Connected to localhost.
  Escape character is '^]'.
  220 gmfccentral.glomedia.net ESMTP Postfix (gmfccentral.glomedia.net)
  HELO localhost
  250 gmfccentral.glomedia.net
  MAIL FROM: <root@glomedia.net>
  250 Ok
  RCPT TO: <acorreo@glomedia.net>
  250 Ok
  DATA
  354 End data with <CR><LF>.<CR><LF>
  Este es un mensaje de pruebas enviado a traves de telnet.
  .
  250 Ok: queued as 5F496BEE9
  QUIT
  221 Bye
  Connection closed by foreign host.
```

Es muy recomendable probar la secuencia anterior desde otras máquinas no autorizadas y ver la respuesta del sistema.

1.3. TLS

Para poder cifrar el canal de comunicaciones se va a utilizar **TLS**, *Postfix* no lo implementa directamente sino vía *OpenSSL*, lo primero que por tanto, hemos de realizar es la instalación de dicho paquete:

```
apt-get install openssl
```

Posteriormente hemos de instalar el módulo que permite trabajar con TLS a *Postfix*:

```
apt-get install postfix-tls
```

Ya lo tenemos todo listo para securizar nuestro canal SMTP.

1.3.1. Creación de los certificados

Posteriormente hemos de generar los certificados, en este caso firmados por nosotros mismos, más que nada para ahorrar dinero (si desea obtener certificados gratuitos, puede ir a <http://www.cacert.org>). Los pasos para generar el certificado consisten en:

- Crear la autoridad certificadora:

```
/usr/lib/ssl/misc/CA.pl -newca
```

Respondemos a las cuestiones de forma adecuada (recomendamos no dejar en blanco ningún campo), por defecto, el certificado de la autoridad certificadora se guarda en un directorio llamado *demoCA* dentro del directorio donde ejecutemos el comando y se denomina *cacert.pem*.

- Realizar la petición de un certificado, responda a las preguntas y no añada una contraseña al certificado para que el servidor no se quede bloqueado esperándola al iniciarse:

```
/usr/lib/ssl/misc/CA.pl -newreq-nodes
```

El fichero se generará en el mismo directorio antes mencionado y se llamará *newreq.pem* (clave privada).

- Firmar el certificado, lo que también generará la clave pública almacenada en *newcert.pem*, que se enviará al cliente para establecer la comunicación segura:

```
/usr/lib/ssl/misc/CA.pl -sign
```

Ahora hemos de copiar los tres ficheros anteriores al directorio *ssl* de *Postfix* y ajustar permisos:

```
mkdir /etc/postfix/ssl
cp demoCA/cacert.pem /etc/postfix/ssl/
cp newcert.pem /etc/postfix/ssl/
cp newreq.pem /etc/postfix/ssl/
chown root /etc/postfix/ssl/newreq.pem
chmod 400 /etc/postfix/ssl/newreq.pem
```

1.3.2. Configuración de Postfix

Tan sólo es necesario editar el fichero */etc/postfix/main.cf* y agregar las siguientes líneas, las cuales, como puede observar indican la ruta a los certificados generados:

```
# Deseamos activar TLS
smtpd_use_tls = yes
# Sólo autenticación vía TLS
smtpd_tls_auth_only = yes
# Rutas a los certificados y entidades certificadoras
smtpd_tls_key_file = /etc/postfix/ssl/newreq.pem
smtpd_tls_cert_file = /etc/postfix/ssl/newcert.pem
smtpd_tls_CAfile = /etc/postfix/ssl/cacert.pem
# Nivel de registro, bajar a 1 en producción
smtpd_tls_loglevel = 3
smtpd_tls_received_header = yes
# Tiempo de espera para la caché de seiones
smtpd_tls_session_cache_timeout = 3600s
# Generador de entropía, para grandes
```

```
# cargas usar hardware específico
tls_random_source = dev:/dev/urandom
```

De este modo restringimos que las autenticaciones al envío se puedan hacer únicamente usando TLS (si no desea este compartamiento, comente la directiva *smtpd_tls_auth_only*). Además, obligamos a que todas las comunicaciones con el demonio *smtpd* se hagan a través de TLS. Ahora reiniciamos el servidor:

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

Ya podemos comprobar el correcto funcionamiento del mismo:

```
$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 gmfccentral.glomedia.net ESMTP Postfix (Sistema de correo en GMF)
EHLO localhost
250-gmfccentral.glomedia.net
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250 8BITMIME
STARTTLS
220 Ready to start TLS
QUIT
QUIT
Connection closed by foreign host.
```

1.3.3. Uso del canal seguro

La configuración es muy sencilla, siempre que el cliente soporte TLS, en nuestro caso hemos seleccionado el *KMail* de *KDE*, hemos de configurar una nueva cuenta de correo saliente, indicar el servidor de correo, su puerto y especificar que se desea utilizar TLS. El soporte en otros clientes es muy similar y no ha de ser problemático. Ahora, si disponemos de un *sniffer*, veremos cómo el correo fluye cifrado desde nuestra máquina y hasta el servidor SMTP.

1.4. Autenticación con SASL

SASL va a permitir a *Postfix* autenticar a clientes SMTP remotos y autenticar el servidor SMTP frente a servidores SMTP que pidan autenticación.

En primer lugar, hemos de instalar los paquetes *postfix-tls* (que ya tendremos instalado, hemos de resaltar que aunque no deseemos soporte para TLS, realmente este

paquete también ofrece soporte SASL a Postfix) *libsasl2-modules*, *sasl2-bin* *openssl* (ya hemos de tenerlo si hemos seguido la sección anterior):

```
apt-get install libsasl2-modules sasl2-bin
```

Para habilitar el soporte Cyrus-SASL en Postfix, hemos de añadir las siguientes líneas a **/etc/postfix/main.cf**:

```
# SOPORTE SASL PARA CLIENTES SMTP
smtpd_sasl_auth_enable = yes
smtpd_sasl_security_options = noanonymous
smtpd_sasl_local_domain = $myhostname
broken_sasl_auth_clients = yes
# Sólo permitimos clientes autenticados
smtpd_recipient_restrictions = permit_sasl_authenticated,
    permit_mynetworks,
    reject_unauth_destination
```

Ahora es el momento de configurar el propio Cyrus-SASL, para ello hemos de crear el fichero */etc/postfix/sasl/smtpd.conf*, en él vamos a indicar que deseamos utilizar la base de datos de SASL para almacenar los usuarios y contraseñas de éstos, además de los métodos de autenticación aceptados:

```
pwcheck_method: auxprop
mech_list: digest-md5 cram-md5
```

Esta configuración permite que en el sistema no tengan que existir realmente los usuarios, obteniendo una mayor independencia y seguridad, así como el almacenamiento del *hash md5* de las contraseñas, en lugar de almacenarlas en texto claro. Otros métodos de autenticación son *login* y *plain*.

Tras las configuraciones, sólo hemos de añadir usuarios a la base de datos de *SASL*, es muy importante que conozcamos el valor de la variable *myhostname* descrita en el fichero **/etc/postfix/main.cf**, ya que se utilizará en el proceso, en nuestro caso el valor es **gmfcentral.glomedia.net**:

```
saslpasswd2 -c -u gmfcentral.glomedia.net -a smtpauth usuarioDemo
```

Lo cual creará el usuario *usuarioDemo* para la tarea *smtpauth*, la que permite autenticar en el servidor. Esto creará el fichero */etc/sasl2/sasl2*, propiedad de *root*; hemos de asegurarnos de que:

- El grupo al que pertenece el fichero sea **sasl**.
- El usuario *postfix* pertenezca al grupo **sasl**.

Podremos ver si el usuario se ha creado de forma correcta mediante:

```
sasldblistusers2
```

Es muy importante recalcar el hecho de que en Debian, *Postfix* se ejecuta en un entorno *Chroot*, lo que puede complicar enormemente la gestión del SASL, de forma que vamos a desactivar dicha característica, si desea un *Postfix* realmente enjaulado le recomendamos use *debootstrap* para enjaular una *Debian* al completo e instalar allí *Postfix*, será más eficiente, seguro y menos complejo. Edite el fichero */etc/postfix/master.cf* tal y como se indica a continuación:

```
# Línea comentada, sin CHROOT
# smtp      inet  n       -       -       -       -       smtpd
# No deseamos el CHROOT, ya estaremos en uno y complica la administración
smtp      inet  n       -       n      -       -       smtpd
```

Sólo nos quedará reiniciar *Postfix*:

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

Si desea ver si el mecanismo SASL se carga de forma correcta, proceda como sigue:

- Desactive **temporalmente** la autenticación *TLS* fijando la directiva *smtpd_tls_auth_only* a **no** (fichero *etc/postfix/main.cf*), si no lo hace, pese a **estar usando SASL**, **no se mostrará ningún mensaje** ya que antes hay que autenticarse vía *TLS*:

```
smtpd_tls_auth_only = no
```

- Relea la configuración de *Postfix*:

```
/etc/init.d/postfix reload
```

- Conecte vía *telnet* al servidor de correo y haga *EHLO*, ha de ver 250-AUTH DIGEST-MD5 CRAM-MD5:

```
telnet gmfccentral.glomedia.net 25
Trying 192.168.0.60...
Connected to gmfccentral.glomedia.net.
Escape character is '^]'.
220 gmfccentral.glomedia.net ESMTP Postfix (Sistema de correo en GMF)
ehlo localhost
250-gmfccentral.glomedia.net
250-PIPELINING
250-SIZE 10240000
250-VERFY
250-ETRN
250-STARTTLS
250-AUTH DIGEST-MD5 CRAM-MD5
250 8BITMIME
```

- Vuelva a fijar la directiva *smtpd_tls_auth_only* a **yes**

```
smtpd_tls_auth_only = yes
```

1.4.1. Pruebas

Tenga en cuenta que la configuración que hemos establecido crea los usuarios sólo para autenticar frente al servidor de correo electrónico, **los usuarios SASL no tienen buzón**, al menos de momento, aunque es posible asignárselo. Para probar, puede utilizar algún cliente, por ejemplo *KMail*, en sus opciones de correo saliente explicita que *el servidor requiere autenticación*, como método puede seleccionar *DIGEST-MD5*, aunque *KMail* puede autoconfigurarse. Pruebe a enviar un mensaje utilizando como servidor saliente el recientemente configurado y observará que se le pide contraseña y login, introduzca la del usuario *SASL*.

Capítulo 2

Integración con Clamav

ClamAV es una herramienta antivirus GPL para UNIX. El propósito principal de este software es la integración con los servidores de correo. El paquete proporciona un servicio multihilo flexible y escalable, un analizador de línea de comandos y una utilidad para la actualización automática via Internet.

Los programas están basados en una librería distribuida con el paquete *Clam AntiVirus*, la cual puede ser usada por su propio software. Y lo más importante, la base de datos se mantiene actualizada constantemente. Otras características destacables son el soporte de firmas digitales en la actualización de la base de datos, el análisis durante el acceso bajo *Linux* y *FreeBSD*, la detección de más de 20000 virus, gusanos y troyanos, el soporte integrado para archivos comprimidos con *Rar*, *Zip*, *Gzip* y *Bzip2* y formatos de correo *Mbox*, *Maildir* y ficheros planos de correo

2.1. Instalación de Clamav

Hemos de instalar los siguientes paquetes para sacar el máximo jugo al sistema antivirus¹:

```
apt-get install unrar lha arj unzoo zip unzip bzip2 gzip cpio file lzop
```

Posteriormente instalamos el propio sistema antivirus libre:

```
apt-get install clamav clamav-base clamav-daemon clamav-freshclam libclamav1
```

En el proceso de instalación se nos preguntará por:

1. El método de actualización de la base de datos del antivirus, de ello se encargará el programa **freshclam**, pudiendo seleccionar:

¹Asegúrese de tener *non-free* en su repositorio de */etc/apt/sources.list*

- a) **daemon**, *freshclam* estará siempre ejecutándose como demonio, es la opción recomendada si tenemos conexión permanente a Internet. Es la que hemos escogido nosotros.
 - b) **ifup.d**, *freshclam* se actualizará cuando el equipo se conecte a Internet.
 - c) **cron**, es el que permite el mayor nivel de personalización. *Freshclam* será ejecutado periódicamente por el *cron*.
 - d) **manual**, hemos de ejecutar *freshclam* de forma manual.
2. Posteriormente se nos pregunta por el servidor espejo de Clam más cercano a nosotros, en el momento de creación de este manual, no existe ningún *mirror* en España, seleccionamos ***db.pt.clamav.net (Portugal)***. Si fuera necesario, introduzca la configuración para el proxy.
 3. Se nos preguntará el número de actualizaciones por día, está bien con 12, o cada 2 horas.
 4. En el paso siguiente se nos pregunta si deseamos que el demonio *freshclam* notifique al antivirus que la base de datos de virus ha sido actualizada, es muy recomendable, en caso contrario podremos tener problemas ya que trabajaremos con una base de patrones obsoleta.
 5. Se nos creará un usuario y grupo llamado ***clamav***.

No es preciso modificar ningún fichero de configuración, pues la instalación ya nos deja todo lo que necesitamos funcionando adecuadamente, pero si observamos el siguiente error en el */var/log/mail.log*:

```
gmfcentral amavis[13147]: (13147-01) Clam Antivirus-clamd FAILED -
unknown status: /var/lib/amavis/amavis-20040818T163812-13147/parts:
Access denied.
ERRORn gmfcentral amavis[13147]: (13147-01) WARN: all primary virus
scanners failed, considering backups
```

Entonces deberemos añadir el usuario clamav al grupo amavis, tal que:

```
adduser clamav amavis
```

Y la directiva *AllowSupplementaryGroups* al fichero */etc/clamav/clamav.conf*. Si desea reconfigurar el sistema de actualizaciones, puede ejecutar:

```
dpkg-reconfigure clamav-freshclam
```

2.2. Instalación de Amavis-new

Instalar el ***Amavis-new*** es tan simple como ejecutar:

```
apt-get install amavisd-new
```

Se nos instalará el sistema y sus múltiples dependencias; Amavis se encargará de conectar *Clamav* con *Postfix*. El sistema de instalación creará un usuario y grupo llamado **amavis** con directorio de trabajo **/var/lib/amavis**.

El fichero de configuración de *Amavis* (*/etc/amavis/amavisd.conf*) es muy extenso, de momento, es suficiente con activar las siguientes directivas (se fijan a 1):

- **warnvirussender**, si está activada, se envía un mensaje de correo al usuario que envió el correo infectado.
- **warnbannedsender**, si está activada, se envía un mensaje de correo al usuario que envió el correo con ficheros adjuntos *baneados*.
- **banned_filename_re**, podemos personalizar que tipos de adjuntos se permitirán y cuales no en base a su extensión.

Por otra parte es recomendable fijar el dominio en la directiva *\$mydomain*, así como la dirección de correo a la que se enviarán los avisos:

```
$mydomain = 'glomedia.net';
$virus_admin = "virusalerta\@$mydomain";
```

Tras esto ya podemos reiniciar el servicio mediante:

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

Es muy recomendable observar los mensajes que aparezcan en */var/log/mail.log*, donde se informa de todos los módulos cargados al iniciar, sobre todo hemos de prestar especial atención a la detección de *Clamav*.

...

```
Feb 14 22:19:18 gmfccentral amavis[3791]: Using internal av scanner
code for (primary) Clam Antivirus-clamd
```

```
Feb 14 22:19:18 gmfccentral amavis[3791]: Found secondary av scanner
Clam Antivirus - clamscan at /usr/bin/clamscan
```

2.3. Configuración de Postfix

Las modificaciones a realizar en *Postfix* son muy sencillas. En primer lugar, editamos el fichero */etc/postfix/master.cf* y le añadimos estas líneas:

```
127.0.0.1:10025 inet n - n - - 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=1001  
-o smtpd_hard_error_limit=1000  
smtp-amavis unix - - n - 2 lmtp  
-o lmtp_data_done_timeout=1200  
-o lmtp_send_xforward_command=yes
```

Y en el fichero `/etc/postfix/main.cf` tan sólo debemos añadir esta línea:

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

De este modo añadimos un filtro de contenido a *Postfix*, el cual redirigirá el tráfico al puerto `10024` de la interfaz *loopback*. Una vez *Amavisd-new* haya finalizado su trabajo, devolverá el mensaje a *Postfix* a través del puerto `10025`, donde hemos habilitado un *smtpd*.

Para concluir, tan sólo queda reiniciar el servidor *Postfix* mediante el comando:

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

Si probamos a enviar un correo, veremos que en su cabecera se ha añadido la línea:

```
X-Virus-Scanned: by amavisd-new (Debian) at example.com
```

Podremos cambiar este banner mediante las directivas **X_HEADER_TAG** y **X_HEADER_LINE** en el fichero de configuración de *Amavis*.

Si tiene a mano un virus, pruebe a enviarlo, el usuario ha de recibir un mensaje informando del incidente y el destinatario no recibirá nada. Puede encontrar la demo *EICAR* en http://www.eicar.org/anti_virus_test_file.htm

Capítulo 3

El Spam

3.1. Sistema básico de filtrado

3.1.1. Pasos a seguir

Para empezar, debemos añadir al fichero */etc/postfix/main.cf* dos líneas para indicarle que realice chequeos, **tanto en las cabeceras** como como en el **cuerpo**:

```
header_checks = regexp:/etc/postfix/regexp.header
body_checks = regexp:/etc/postfix/regexp.body
```

Ahora, deberemos editar ambos ficheros (*regexp.header* y *regexp.body*), para indicarle qué ha de filtrar, y qué hacer con los mensajes que coincidan con el filtro.

3.1.2. Filtrado por asunto

Para filtrar en base al *subject* del mensaje hemos de editar *regexp.header*, como sigue:

```
--regexp.header--
# No queremos ver nada relativo a Gane Millones ya!!
/^Subject: Hágase millonario sin esfuerzo!$/ REJECT
# Aceptamos todo lo de hotmail
/^From: postmaster@hotmal.com$/ ACCEPT
# Y rechazamos todo lo que tenga sex en el subject
/^Subject:.*sex.*$/ REJECT
--regexp.header--
```

Básicamente, el fichero se procesa de arriba a abajo y la primera expresión que coincide con el mensaje, se aplica y listo.

Si no tienes claro qué es cada cosa, puedes echarle un ojo a los artículos sobre expresiones regulares que podrás encontrar en <http://bulmalug.net/body.phtml?nIdNoticia=736>.

3.1.3. Filtrado por contenido

Si lo que queremos filtrar es el **cuerpo** del mensaje, hacemos lo mismo, pero con el fichero *regexp.body* que hemos definido en *main.cf*:

```
--regexp.body--
# SirCAM
#/^Te mando este archivo para que me des tu punto de vista$/ REJECT
# Virusetes
/filename=".*\.{1,4}\.(exe|com|bat|lnk|pif|vbs)/ REJECT
/^begin [0-9]{3} .*\.{1,4}\.(exe|com|bat|lnk|pif|vbs)$/ REJECT
--regexp.body--
```

Como vemos, el formato es exactamente igual. La única diferencia es que estas expresiones regulares se chequean contra el cuerpo del mensaje, y **no** contra la cabecera.

Una vez creados los ficheros, solo tenemos que ejecutar */etc/init.d/postfix reload* para que postfix recargue la configuración. Puedes mirar los *logs* de *postfix* para comprobar cómo se filtra la basura.

3.2. Apache SpamAssassin

SpamAssassin es un filtro de correo que trata de identificar el spam mediante el análisis del texto y el uso en tiempo real de algunas listas negras a través de Internet. A partir de su base de datos de reglas, utiliza un amplio abanico de pruebas heurísticas en las cabeceras y el cuerpo de los correos para identificar el *spam*.

Una vez identificado, el correo puede ser opcionalmente marcado como *spam* o más tarde filtrado usando el cliente de correo del usuario. *SpamAssassin* normalmente identifica acertadamente entre un 95 y un 99% del *spam*, dependiendo del tipo de correo que se reciba. Para instalarlo en *Debian Sarge* es suficiente con ejecutar:

```
apt-get install spamassassin spamc
```

Debido a que la configuración con la cual se ejecuta *SpamAssassin* al ser llamado desde *Amavisd-new* es la que se establece en el fichero de configuración de este último, no tendremos que configurar en demasía, tan sólo, para que *amavisd-new* haga uso de *Spamassassin*, en el archivo de configuración de *amavisd-new* se **ha de comentar** la línea:

```
# @bypass_spam_checks_acl = qw( . );
```

En el fichero */etc/default/spamassassin* podemos observar que el demonio de *SpamAssassin* no se ejecuta por defecto, ese es el comportamiento que nos conviene:

```
ENABLED=0
OPTIONS="-c -m 10 -a -H"
```

La ventaja principal de ejecutarlo como demonio sería su eficiencia, pues las comunicaciones se establecerían a través del puerto 783 en lugar de tener que arrancar un ejecutable cada vez que se tuviera que analizar un correo. En cambio, se correrían ciertos riesgos de seguridad, pues el paquete Debian nos deja una configuración por defecto que hace que se ejecute como *root* (en la documentación se explica como cambiarlo para que se ejecute como un usuario no privilegiado). Entonces, una posible vulnerabilidad a causa de un error en el código podría dar al atacante permisos de *root*.

En cambio, debido a que se usará *SpamAssassin* a través de *Amavisd-new*, éste será llamado a través del módulo de *Perl Mail::SpamAssassin*, manteniendo *Perl* el motor de reglas siempre cargado en memoria y consiguiendo **la misma eficiencia que con el demonio**. De hecho, este es el comportamiento por defecto de los paquetes Debian de estos dos softwares.

Si se desean utilizar los filtros bayesianos del *SpamAssassin* (lo cual es muy recomendable hacerlo si se quiere tener un alto porcentaje de acierto) será preciso entrenarlo. Según el manual, varios miles de mensajes deben ser proporcionados a *SpamAssassin*, tanto de *spam* como de *ham* (correo no-spam). Para ello se usa la herramienta *sa-learn* (puede ejecutar *man sa-learn* para su documentación), de forma general, para entrenar sobre el *spam* o *ham*, hemos de indicar el directorio donde se encuentran los mensajes correspondientes:

```
sa-learn --spam <directorio>
sa-learn --ham <directorio>
```

Asimismo, *sa-learn* tiene una opción que permite pasarle un fichero que contenga una lista de directorios, uno en cada línea, en los cuales buscará el tipo de correo que le especifiquemos.

El parámetro, *-folders=file*, es muy útil si queremos recoger una lista de buzones de usuarios que sabemos con seguridad que sólo guardan *spam* o *ham* y utilizarlos para continuamente mejorar nuestros filtros desde un tarea del *cron*, pues esta herramienta mantiene una lista de los correos que ya ha analizado y se los salta cada vez, haciendo este proceso bastante eficiente.

Es importante tener en cuenta que la base de datos bayesiana se encuentra en */var/lib/amavis/.spamassassin*, pues *SpamAssassin* es llamado a través de *Amavisd-new* (módulo *Mail::SpamAssassin* de Perl). Por lo tanto, cuando queramos usar la herramienta *sa-learn* deberemos hacerlo siempre con el usuario *amavis* que fue creado en el proceso de instalación de *Amavis*.

Para concluir, hemos de reiniciar *Amavis*:

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

3.3. Prueba del sistema

Para determinar si el sistema funciona correctamente tan sólo hemos de probar a enviar un mensaje con el contenido del fichero */usr/share/doc/spamc/sample-spam.txt*:

```
$ /usr/bin/mail acorreo@glomedia.net
```

Subject: Prueba para el control antispam
This is the GTUBE, the
Generic
Test for
Unsolicited
Bulk
Email

If your spam filter supports it, the GTUBE provides a test by which you can verify that the filter is installed correctly and is detecting incoming spam. You can send yourself a test mail containing the following string of characters (in upper case and with no white spaces and line breaks):
XJS*C4JDBQADN1.NSBN3*2IDNEN*GTUBE-STANDARD-ANTI-UBE-TEST-EMAIL*C.34X
You should send this test mail from an account outside of your network.

.
Cc: [enter]

El mensaje ha de ser rechazado por el servidor de correo, enviando un aviso al origen de dicho correo.

Capítulo 4

Sistema de entrega segura

El objetivo a conseguir será tener un sistema de entrega de correos seguro; constará de dos partes, en una primera se configurará un servidor IMAP seguro y en la segunda, un sistema de WebMail accesible por todos nuestros clientes mediante HTTPS, de esta forma garantizamos la seguridad desde el momento en el que se escribe el mensaje y hasta que sale del servidor SMTP y desde que llega al servidor y se entrega al cliente.

4.1. IMAP seguro mediante Cyrus IMAP

4.1.1. Instalación y configuración básica

Para instalar el servidor Cyrus IMAP (vamos a omitir el servidor POP3) ejecutaremos el siguiente comando:

```
apt-get install cyrus21-admin cyrus21-common cyrus21-doc cyrus21-imapd
```

Opcionalmente, también podemos instalar el paquete **cyrus21-clients**, que proporciona la herramienta *imtest*, la cuál nos será útil para comprobar el buen funcionamiento de nuestro servidor.

Al instalar estos paquetes se crea una jerarquía de directorios en */var/spool/cyrus/mail* que almacenará los buzones de correo. Todos estos buzones tendrán como propietario al usuario **cyrus** y al grupo **mail**. Existen únicamente dos ficheros de configuración para toda la plataforma Cyrus:

- */etc/cyrus.conf*, donde se determinan los servicios que se ejecutarán al arrancar el sistema Cyrus.

Este fichero de configuración consta de tres partes claramente diferenciadas:

- **START**: esta sección lista los scripts que se ejecutarán antes de que se arranquen los servicios. Su uso más característico es inicializar las bases de datos y lanzar los servicios de larga ejecución.

- **SERVICES:** esta sección es el corazón del fichero */etc/cyrus.conf*, pues describe los procesos que deberán lanzarse para atender las conexiones que los clientes hagan a ciertos sockets, bien sean tipo TCP o UNIX.
- **EVENTS:** esta sección lista los procesos que deberían ejecutarse a intervalos específicos, de modo similar a los trabajos del *cron*. Típicamente se usa para llevar a cabo tareas programadas de limpieza y mantenimiento.

Sólo hemos de editar la sección de *SERVICES*, de comentar la línea que activa IMAPS, comentar la línea que intenta activar POP3 y limitar IMAP sólo desde *localhost*:

```
imap          cmd="imapd -U 30" listen="localhost:imap" prefork=0 maxchild=100
imaps         cmd="imapd -s -U 30" listen="imaps" prefork=0 maxchild=100
# pop3        cmd="pop3d -U 30" listen="pop3" prefork=0 maxchild=50
```

Una cuestión importante en la configuración es la decisión de usar sockets TCP o sockets UNIX. En la configuración que se acaba de presentar se ha optado por la segunda opción debido a que se considera que van a ejecutarse todos los servicios en la misma máquina. En un caso como éste, es mejor usar sockets UNIX debido, principalmente, al mejor rendimiento que ofrecen y a que simplifican la configuración en general. En cambio, en un contexto donde los servidores *Postfix* y *Cyrus IMAP* estén en máquinas diferentes, será necesario usar sockets TCP.

Ahora ya podemos reiniciar el servidor Cyrus:

```
/etc/init.d/cyrus21 restart
```

- */etc/imapd.conf*, donde se especifican los parámetros de configuración del servicio IMAP de Cyrus.
 - Cada una de las líneas de tiene el formato *opción: valor*, donde *opción* es el nombre de la opción a configurar y *valor* el valor al cual se está estableciendo esa opción. A continuación se detallan algunas de las opciones más relevantes y sus valores recomendados:
 - *altnamespace*: esta opción viene por defecto con el valor *no*, forzando que las subcarpetas de usuario se creen debajo de *inbox*; si se cambia a *yes*, las subcarpetas del usuario se crearán a la misma altura que *inbox*. Dejaremos el valor por defecto.


```
altnamespace: no
```
 - *lmtp_downcase_rcpt*: esta opción, que sirve para forzar que el nombre de usuario se convierta a minúsculas, viene por defecto comentada, es decir, con valor *no*. Debido a que *Cyrus* diferencia mayúsculas y minúsculas, es una buena idea trabajar con los nombres de usuario siempre en minúsculas (el valor por defecto asume que el usuario es consciente de lo que está haciendo). Descomentamos la línea y fijamos el valor a *yes*.

```
lmtp_downcase_rcpt: yes
```

- o **admins**: esta opción permite definir los usuarios que tendrán permisos de *administrador* (flag *a* de la ACL de un buzón) sobre todos los buzones del sistema. El usuario *cyrus*, y únicamente él, es la opción más recomendable, por lo que bastará con **descomentar** la línea del fichero.

```
admins: cyrus
```

Este usuario se va a autenticar mediante el método SASL, por lo que debe añadirse a la base de datos */etc/sasl/db2*, como ya hemos visto. Dejamos este paso para más adelante.

- o **lmtp_admins**: esta opción permite especificar una lista de usuarios, separados por espacios, que tendrán la categoría de administradores LMTP, es decir, que podrán enviar correo a través de LMTP por TCP/IP (además de aquellos definidos en la opción *admin* anterior). Si se van a usar sockets UNIX no es necesario descomentar esta opción, pues el usuario *postman* (valor por defecto) es autenticado automáticamente.

```
#lmtp_admins: postman
```

- o **allowanonymouslogin**: esta opción permite el acceso anónimo a los buzones en cuyas ACLs se haya añadido al usuario *anonymous*. Carece de sentido a menos que se quieran implementar grupos de noticias, por lo que se dejará su valor por defecto **no**.

```
allowanonymouslogin: no
```

- o **umask**: esta opción permite definir los permisos con los cuáles se guardarán los ficheros y subdirectorios dentro de */var/spool/cyrus/mail*. Por defecto tiene el valor *077* (lectura y escritura para el propietario, nada para el resto), pero es conveniente permitir que el grupo (*mail* por defecto) tenga también permisos de lectura, pues de ese modo otras aplicaciones podrán leer el contenido de los emails, por ejemplo *amavisd-new* (bastará con que añadamos al usuario con el cuál se ejecutan a ese grupo).

- o **allowplaintext**: mediante esta opción decidimos si vamos a permitir uso del mecanismo de autenticación **sasl** PLAIN. Lo fijamos a **no**.

```
allowplaintext: no
```

- o **sasl_mech_list**: esta es la lista de los mecanismos de autenticación que se van a soportar. Es útil para evitar que se prueben todos los plugings existentes y para definir el orden de los mismos. Dejamos la línea comentada, ya que no deseamos utilizar PLAIN (recuerde que estamos usando *digest-md5* y *cram-md5*, otra opción es fijar esta variable con dichos valores).

```
#sasl_mech_list: PLAIN
```

- o **sasl_minimum_layer**: el SSF (del inglés, *security strength factor*) mínimo que el servidor permitirá negociar al cliente. Un valor igual a 1 requiere protección de integridad; un valor más algo requiere algún tipo de cifrado. Dejamos comentada la línea:

- ```
#sasl_minimum_layer: 0
```
- **sasl\_maximum\_layer**: valor máximo del SSF que el servidor permitirá negociar al cliente. Dejamos comentada la línea:

```
#sasl_maximum_layer: 256
```
  - **sasl\_auxprop\_plugin**: Esta opción nos permite especificar los plugins del *auxpropd* que deseamos cargar, en el caso de estar usando *sasl\_pwcheck\_method: auxprop*. Es necesario descomentar esta línea para que use *sasldb*.

```
sasl_auxprop_plugin: sasldb
```
  - **lmtpsocket, idlesocket y notifysocket**: estas tres opciones especifican las rutas para los tres sockets UNIX que utiliza Cyrus. Los valores por defecto son correctos.

Además, ya que deseamos activar SSL sobre IMAP, hemos de configurar las directivas **tls\_\*** de */etc/imapd.conf*, en nuestro caso vamos simplemente a decomentar las líneas indicadas a continuación y a reutilizar los certificados creados para *Postfix*:

```
tls_cert_file: /etc/ssl/certs/cyrus-global.pem
tls_key_file: /var/imap/cyrus-global.key
tls_ca_file: /etc/ssl/certs/cyrus-imapd-ca.pem
tls_ca_path: /etc/ssl/certs
tls_session_timeout: 1440
tls_cipher_list: TLSv1:SSLv3:SSLv2:!NULL:!EXPORT:!DES:!LOW:@STRENGTH
```

Copiamos los certificados creados para *Postfix* y ajustamos permisos:

```
mkdir /var/imap
chown cyrus:mail /var/imap
chmod 750 /var/imap
cp /etc/postfix/ssl/newcert.pem /etc/ssl/certs/cyrus-global.pem
cp /etc/postfix/ssl/cacert.pem /etc/ssl/certs/cyrus-imapd-ca.pem
cp /etc/postfix/ssl/newreq.pem /var/imap/cyrus-global.key
chown cyrus:mail /var/imap/cyrus-global.key
chmod 600 /var/imap/cyrus-global.key
```

Ahora, de nuevo, podemos reiniciar el servidor *Cyrus*:

```
/etc/init.d/cyrus21 restart
```

#### 4.1.2. Conexión con Postfix

Usaremos el protocolo *LMTP* para la comunicación entre *Postfix* y *Cyrus*, ya que ofrece un muy buen rendimiento. Usaremos el socket */var/run/cyrus/socket/lmtp*, por lo que debemos asegurarnos de que el servicio *lmtpunix* está habilitado en el fichero */etc/cyrus.conf* y que *Postfix* tiene acceso a ese fichero. Asimismo, *Cyrus* requiere que las entregas por *LMTP* estén autenticadas, y asume que las que se hagan a través del

socket Unix son de confianza y las preautentica como si vinieran del usuario *postman* (ficticio). Por lo tanto, nos aseguraremos de que el fichero */etc/postfix/master.cf* contenga esta línea:

```
service type private unpriv chroot wakeup maxproc command + args
(yes) (yes) (yes) (never) (100)
=====
lmtpl unix - - n - - lmtpl
```

Para conseguir que *Postfix* entregue los correos a Cyrus a través de *LMTP* deberemos configurar un transporte en el primero. Para ello, añadimos a */etc/postfix/main.cf*:

```
Transporte para entregar correos a Cyrus
mailbox_transport = lmtpl:unix:/var/run/cyrus/socket/lmtpl
```

Hemos de asegurarnos de que aparece la línea siguiente en */etc/cyrus.conf*, la cual habilita un servicio *lmtpl* de *Cyrus* escuchando en ese socket:

```
lmtplunix cmd="lmtpl" listen="/var/run/cyrus/socket/lmtpl" prefork=0 maxchild=20
```

Además, vamos a evitar que *Postfix* busque los usuarios en el sistema, ahora sólo valdrán los de *SASL*, añadiendo la línea siguiente a */etc/postfix/main.cf*:

```
Evitamos buscar usuarios en el sistema
local_recipient_maps =
```

Usamos *dpkg-statoverride* para asegurarnos de que la configuración de los permisos del socket no es sobrescrita por los paquetes de *Cyrus*. Recuerde que *Postfix* ejecuta el transporte LMTP con el usuario definido en */etc/postfix/master.cf*, por defecto *postfix*. Para ello ejecutamos los siguientes comandos:

- Cree un grupo llamado *lmtpl*:

```
addgroup lmtpl
```

- Agregue el usuario *postfix* a ese grupo:

```
adduser postfix lmtpl
```

- Corrija los permisos del directorio del socket:

```
dpkg-statoverride --force --update --add cyrus lmtpl 750 /var/run/cyrus/socket
```

- Reinicie *Postfix* y *Cyrus*:

```
/etc/init.d/postfix restart
/etc/init.d/cyrus21 restart
```

### 4.1.3. Configuración de los buzones de correo

La administración de los buzones de correo se realiza mediante el programa *cyradm*. Deberá usarse uno de los administradores definidos en el fichero */etc/imapd.conf* para conectarse, en nuestro caso acabamos de definir ese usuario como *cyrus*. El par usuario/contraseña será el definido en la base de datos de usuarios */etc/sasl2* mediante el programa *sasldbpasswd2*, tal y como ya hemos estudiado, procedemos, por tanto, a darlo de alta:

```
sasldbpasswd2 -c cyrus
```

En estos momentos, nuestra base de datos de usuarios contiene al usuario *usuarioDemo* y al usuario *cyrus*:

```
gmfccentral:~# sasldblistusers2
cyrus@gmfccentral.glomedia.net: userPassword
usuarioDemo@gmfccentral.glomedia.net: userPassword
```

Ahora ya se puede acceder a la administración de los buzones de Cyrus, mediante */usr/bin/cyradm -user cyrus localhost*. Una vez dentro, el comando *help* nos mostrará una descripción de los comandos disponibles y sus alias. De entre todos, los de uso más frecuente son:

- **cm**, permite crear buzones de correo (*cm user.usuarioDemo*).
- **dm**, permite borrar buzones de correo (*dm user.usuarioDemo*).
- **lam**, lista las ACL de un buzón de correo (*lam user.usuarioDemo*).
- **sam**, establece las ACL sobre un buzón de correo (*sam user.usuarioDemo lrs*).  
Los posibles valores son:

- **l (lookup)**: el usuario puede ver que el buzón de correo existe.
- **r (read)**: el usuario puede leer el buzón de correo. El usuario puede seleccionar el buzón, leer los datos contenidos, llevar a cabo búsquedas y copiar mensajes de ese buzón.
- **s (seen)**: mantiene el estado leído por usuario. Se preservan los flags Seen y Recent para cada usuario.
- **w (write)**: el usuario puede modificar los flags excepto Seen y Deleted (los cuales son controlados por otros permisos).
- **i (insert)**: el usuario puede insertar mensajes en el buzón de correo.
- **p (post)**: el usuario puede mandar correo a la dirección de envío del buzón. Este permiso difiere del permiso *i* en que el sistema de envío inserta información de seguimiento en los mensajes enviados.
- **c (create)**: el usuario puede crear subcarpetas en el buzón.
- **d (delete)**: el usuario puede alterar el flag Deleted, expirar correos y borrar o renombrar el buzón.

- **a (administer)**: el usuario puede cambiar la ACL del buzón.

- **dam**, borra ACL de un buzón de correo (*dam user.usuarioDemo*).

Llegados a este punto ya podemos crear un buzón para el *usuariodemo2*, que vamos a crear a continuación:

```
saslpasswd2 -c usuariodemo2
```

Ahora, le creamos un buzón de correo:

```
cyradm --user cyrus localhost
Password:*****
localhost> cm user.usuariodemo2
localhost> lam user.usuariodemo2
usuariodemo2 lrswipcda
localhost> quit
```

Vamos a instalar el paquete *cyrus21-clients* para probar si todo funciona correctamente:

```
apt-get install cyrus21-clients
```

Lanzamos la herramienta *imtest* tal y como se indica a continuación, si obtenemos el mensaje *Authenticated*, todo irá bien:

```
imtest -a usuariodemo2 -w <contraseña> -m login -s localhost
verify error:num=19:self signed certificate in certificate chain
TLS connection established: TLSv1 with cipher AES256-SHA (256/256 bits)
S: * OK gmfccentral.glomedia.net Cyrus IMAP4 v2.1.18-IPv6-Debian-2.1.18-1 server ready
C: C01 CAPABILITY
S: * CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ MAILBOX-REFERRALS NAMESPACE UIDPLUS ID
S: C01 OK Completed
C: L01 LOGIN usuariodemo2 {11}
S: + go ahead
C: <omitted>
S: L01 OK User logged in
Authenticated.
Security strength factor: 256
```

#### 4.1.4. Prueba

En este momento ya tenemos un sistema de correo cuyos usuarios son independientes de los usuarios de la plataforma, lo que redundará en un mayor grado de seguridad. Para poder leer el correo necesitamos un cliente con soporte IMAPS y con autenticación *DIGEST-MD5*, por ejemplo, KMail. Para enviar los mensajes, utilizaremos como máquina de correo saliente un SMTP sobre *TLS* y para recibirlos *IMAP* sobre *SSL*, todo un lujo.

Recuerde, que en caso de haber fijado direcciones administrativas (por ejemplo, donde se envían los avisos de virus), ha de reajustarlas ahora creando usuarios dentro de SASL con sus correspondientes buzones.

## 4.2. Instalación segura de Apache

El proceso de instalación de Apache2 junto con SSL en Debian Sarge se compone de los pasos siguientes:

- Instalar el paquete Apache2:

```
apt-get install apache2
```

- Crear un certificado *autofirmado*:

```
apache2-ssl-certificate
```

- Activar el módulo SSL en Apache:

```
a2enmod ssl
```

- Hemos de editar el fichero */etc/apache2/sites-available/default* y activar para el host virtual principal SSL añadiendo las líneas:

```
SSLEngine On
```

```
SSLCertificateFile /etc/apache2/ssl/apache.pem
```

- En el mismo fichero, antes del primer host virtual añadimos:

```
NameVirtualHost *:443
```

- Si fuera necesario se crean los dominios virtuales necesarios en este fichero, por ejemplo:

```
#####
Servidores seguros
#####
<VirtualHost *:443>
 ServerAdmin info@glomedia.net
 DocumentRoot /var/www/apache2-default
 ServerName hermes.glomedia.net
 SSLEngine On
 SSLCertificateFile /etc/apache2/ssl/apache.pem
</VirtualHost>
```

- Reiniciamos Apache:

```
/etc/init.d/apache2 restart
```

Ya podremos acceder a nuestro servidor HTTPS mediante *https://localhost*

## 4.3. Instalación del webmail

### 4.3.1. Sistema base

Vamos a usar como sistema de *WebMail*, el muy popular *SquirrelMail*. *SquirrelMail* está escrito en PHP 4 (en sus últimas betas ya soporta PHP5). Incorpora soporte PHP para los protocolos IMAP y SMTP, y todas sus páginas se crean en puro HTML 4.0 (sin requerir el uso de JavaScript), de modo que se garantice la máxima compatibilidad entre navegadores. Tiene muy pocos requerimientos y es muy fácil de instalar y configurar.

*SquirrelMail* tiene toda la funcionalidad que se espera de un cliente de correo electrónico, incluyendo soporte de MIME, agendas de contactos y gestión de carpetas. En Debian GNU/Linux la instalación es tan sencilla como ejecutar:

```
apt-get install squirrelmail
```

Tal y como se nos indica al final de la instalación, para configurar *SquirrelMail* disponemos de la herramienta */usr/sbin/squirrelmail-configure*. De las opciones que nos presenta a lo largo y ancho de los menús, estas son las que consideramos que eran merecedoras de modificación (es vital activar la autenticación DIGEST-MD5):

```
Organization Preferences:
 Organization Name: Glomedia.net
Server Settings:
 Update IMAP Settings
 Authentication type: digest-md5
 Languages:
 Default Language: es_ES
 Default Charset: iso-8859-15
Message of the Day (MOTD):
 Edit the MOTD: Bienvenido al sistema de correo de Glomedia.
```

Grabamos los cambios con la opción *Save data* y ya tan sólo nos queda modificar la configuración de Apache 2 para tener acceso a la interfaz web.

Para ello, el camino más corto y sencillo es, tal y como se documenta en */usr/share/doc/squirrelmail/README.Debian*, modificar el fichero de configuración de Apache para SquirrelMail, que hallaremos en */etc/squirrelmail/apache.conf*, adaptándolo a nuestras necesidades. Finalmente, tan sólo deberemos crear un enlace simbólico en */etc/apache2/conf.d/*, tal que:

```
ln -s /etc/squirrelmail/apache.conf /etc/apache2/conf.d/squirrelmail.conf
```

En el fichero */etc/squirrelmail/apache.conf* tan sólo deberemos descomentar el último apartado, que activa el acceso por https de manera automática (en el caso de encontrarse todos los plugins necesarios) y descomentar el apartado central para definir nuestro virtual host, tal que:

```
<VirtualHost *:80>
DocumentRoot /usr/share/squirrelmail
```

```
ServerName webmail.glomedia.net
</VirtualHost>
...
redirect to https when available (thanks omen@descolada.dartmouth.edu)
#
Note: There are multiple ways to do this, and which one is suitable for
your site's configuration depends. Consult the apache documentation if
you're unsure, as this example might not work everywhere.
#
<IfModule mod_rewrite.c>
 <IfModule mod_ssl.c>
 <Location /squirrelmail>
 RewriteEngine on
 RewriteCond %{HTTPS} !^on$ [NC]
 RewriteRule . https://%{HTTP_HOST}%{REQUEST_URI} [L]
 </Location>
 </IfModule>
</IfModule>
```

Ahora reiniciamos *Apache* mediante:

```
/etc/init.d/apache2 restart
```

Podremos acceder al sistema de correo mediante:

```
https://localhost/squirrelmail
```

Otra opción será el añadir una entrada al DNS para *webmail.glomedia.net*, ya que en la configuración se crea un host virtual.

Tenga en cuenta el lector que la configuración por defecto de SquirrelMail trata de acceder al servidor IMAP a través del puerto 143 del localhost, por lo que deberemos tener activado el servidor en ese puerto. Si se han seguido todos los pasos descritos, nuestro servidor IMAP sólo acepta conexiones sobre la interfaz local, por lo cual no deberemos modificar el fichero */etc/cyrus.conf*. Para comprobar la correcta instalación de Squirrelmail podemos usar el script *configtest.php*, localizable en:

```
https://localhost/squirrelmail/src/configtest.php
```

## **Parte II**

# **Proxy seguro con Squid**

# Capítulo 5

## Squid

### 5.1. Instalación de Squid

La instalación básica consiste en instalar el proxy de forma normal:

```
apt-get install squid
```

Tras esto, se nos configurará el proxy y se activará en el puerto 3128. Es el momento de abrir el navegador y especificar el uso del proxy. Tras la configuración hemos de ver un mensaje de **permiso denegado**, eso es así ya que la configuración por defecto en Sarge deniega todas las conexiones, para cambiar este comportamiento, hemos de editar el fichero */etc/squid/squid.conf*:

- Hemos de crear una ACL que permita el acceso a las máquinas de nuestra red, para ello, buscamos la sección *ACCESS CONTROLS* y añadimos la IP de nuestra red:

```
INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
acl miRed src 10.95.71.0/25
```

- Posteriormente, justo debajo añadimos la línea siguiente, de forma que se permita el acceso desde la red local:

```
http_access allow miRed
```

Ya podemos reiniciar *squid*:

```
/etc/init.d/squid restart
```

## 5.2. Instalación de SquidClamAV

SquidClamAV nos va a permitir filtrar el tráfico que fluye por el proxy, de forma que lo enviará al antivirus que ya tenemos integrado en nuestro sistema y en caso de localizar algún virus, evitará la comunicación. En primer lugar hemos de instalar las dependencias necesarias, luego, el propio software:

- Python necesita ser instalado, ya que el programa redirector está escrito en este lenguaje:

```
apt-get install python python-dev
```

- Necesitamos instalar las librerías para el desarrollador de *Clamav*:

```
apt-get install libclamav-dev
```

- Descargamos y extraemos la última versión de *Python-clamav*:

```
wget http://xael.org/norman/python/pyclamav/pyclamav-0.3.2.tar.gz
tar xzf pyclamav-0.3.2.tar.gz
```

- Desde el directorio que se nos cree procedemos con la construcción e instalación:

```
cd pyclamav-0.3.2
python setup.py build
python setup.py install
```

Ya tenemos el sistema listo para integrar SCAVR (SquidClamAV Redirector), para ello podemos seguir los pasos siguientes:

- Descargamos SCAVR desde [http://www.jackal-net.at/tiki-list\\_file\\_gallery.php?galleryId=3](http://www.jackal-net.at/tiki-list_file_gallery.php?galleryId=3)

.

- Procedemos a su extracción:

```
tar xvfz SCAVR.tar.gz
```

- Copiamos el redirector a */usr/bin*:

```
cp SquidClamAV_Redirector.py /usr/bin
```

- Fijamos permisos de ejecución:

```
chmod +x /usr/bin/SquidClamAV_Redirector.py
```

- Hemos de copiar el fichero de configuración al directorio de Squid:

```
cp SquidClamAV_Redirector.conf /etc/squid
```

Ahora ya podemos adaptar el fichero de configuración `/etc/squid/SquidClamAV_Redirector.conf` a nuestras necesidades; de momento, no cambiamos nada.

Ahora, hemos de integrar *Squid* con SCAVR, para ello, editamos `/etc/squid/squid.conf` y fijamos las siguientes directivas:

```
redirect_program /usr/bin/SquidClamAV_Redirector.py -c
/etc/squid/SquidClamAV_Redirector.conf
redirector_access deny localhost
```

Simplemente, intente descargar algunos de los virus de [http://www.eicar.org/anti\\_virus\\_test\\_file.htm](http://www.eicar.org/anti_virus_test_file.htm) y observe el resultado.