



Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Configuración de un completo servidor de correo seguro con Postfix y Cyrus

(16750 lectures)

Per **Jaume Sabater**, [Primetime](http://www.linuxsilo.net/) (<http://www.linuxsilo.net/>)

Creat el 11/03/2005 16:37 modificat el 24/05/2005 12:23

Debian GNU/Linux + Postfix + Cyrus IMAP + SASL + TLS + LMTP + Sieve + Amavisd-new + SpamAssassin + ClamAV + Mailman + SquirrelMail.

Este documento le guiará a través de los pasos a seguir para instalar y configurar el MTA Postfix y el servidor Cyrus IMAP. El objetivo es conseguir un sistema de correo electrónico totalmente funcional y de alto rendimiento que use un completo abanico de modernas tecnologías y protocolos que mejoren su eficiencia, robustez, flexibilidad y seguridad. Asimismo, se proporcionan muchas facilidades de uso para los usuarios de este sistema.

Versión 1.0.9 con numerosos bugfixes y adiciones.

Índice

1. [Introducción y objetivos.](#)
2. [Tecnologías usadas.](#)
 - 2.1. [Debian GNU/Linux.](#)
 - 2.2. [Postfix.](#)
 - 2.3. [TLS.](#)
 - 2.4. [Cyrus IMAP.](#)
 - 2.5. [Cyrus SASL.](#)
 - 2.6. [LMTP.](#)
 - 2.7. [Sieve.](#)
 - 2.8. [Amavisd-new.](#)
 - 2.9. [SpamAssassin.](#)
 - 2.10. [Clam Antivirus.](#)
 - 2.11. [Mailman.](#)
 - 2.12. [SquirrelMail.](#)
3. [Versiones usadas del software.](#)
4. [Instalación y configuración de Cyrus SASL.](#)
 - 4.1. [El fichero /etc/cyrus.conf.](#)
 - 4.2. [El fichero /etc/imapd.conf.](#)
 - 4.3. [Perspectiva general, conceptos y administración del servidor Cyrus IMAP.](#)
 - 4.3.1. [Espacio de nombres de los buzones de correo.](#)
 - 4.3.2. [Listas de control de acceso.](#)
 - 4.3.3. [Cuotas de usuario.](#)
 - 4.3.4. [Procesos de recuperación de las bases de datos.](#)
 - 4.3.5. [El fichero de buzones.](#)
 - 4.3.6. [Suscripciones.](#)
 - 4.3.7. [Logging.](#)
 - 4.3.8. [El directorio proc.](#)
 - 4.4. [Configuración de buzones de correo.](#)
5. [Instalación y configuración de Postfix.](#)
 - 5.1. [El fichero /etc/postfix/master.cf.](#)
 - 5.2. [El fichero /etc/postfix/main.cf.](#)
 - 5.3. [Autenticación y autorización.](#)



- 5.4. [Alias virtuales](#)
- 5.5. [Dominios virtuales](#)
6. [Cifrado del canal de comunicación usando TLS.](#)
- 6.1. [La generación de los certificados.](#)
- 6.2. [Modificaciones en Postfix.](#)
- 6.3. [Modificaciones en Cyrus IMAP.](#)
7. [El uso de Sieve.](#)
8. [Filtros de contenidos.](#)
- 8.1. [SpamAssassin.](#)
- 8.2. [Clam Antivirus.](#)
- 8.3. [Amavisd-new.](#)
- 8.4. [Modificaciones en Postfix.](#)
- 8.5. [Medidas anti-UCE.](#)
9. [Listas de correo con Mailman.](#)
10. [Correo a través de web con Squirrelmail.](#)
11. [Puertos en un firewall.](#)
12. [Uso de sockets TCP con LMTP.](#)
13. [RFCs.](#)
14. [Agradecimientos.](#)
15. [Bibliografía.](#)
16. [Historial de revisiones.](#)

1. Introducción y objetivos.

Este documento le guiará a través de los pasos a seguir para instalar y configurar el MTA Postfix y el servidor Cyrus IMAP. El objetivo es conseguir un sistema de correo electrónico totalmente funcional y de alto rendimiento que use un completo abanico de modernas tecnologías y protocolos que mejoren su eficiencia, robustez, flexibilidad y seguridad. Asimismo, se proporcionan muchas facilidades de uso para los usuarios de este sistema.

Al final del artículo conseguiremos tener un sistema de correo con las siguientes características:

- Independencia de los usuarios de sistema y las cuentas de correo electrónico.
- Un dominio principal donde se crean cuentas de correo.
- Múltiples dominios virtuales con redirecciones a las cuentas del dominio principal.
- Autenticación a través de SASL (Simple Authentication and Security Layer), con métodos de texto plano o login.
- Transporte seguro del tráfico mediante TLS.
- Acceso a los buzones por IMAP sobre SSL y por webmail.
- Filtrado de correo en el servidor a través de Sieve.
- Filtros antivirus y antispam.
- Listas de correo.

2. Tecnologías.

2.1. Debian GNU/Linux.

[Debian](#)⁽¹⁾ es un sistema operativo (SO) [libre](#)⁽²⁾ para ordenadores. El sistema operativo es el conjunto de programas básicos y utilidades que hacen que funcione el ordenador. Debian utiliza el núcleo [Linux](#)⁽³⁾ (el corazón del sistema operativo), pero la mayor parte de las herramientas básicas vienen del [Proyecto GNU](#)⁽⁴⁾; de ahí el nombre GNU/Linux.

Debian GNU/Linux ofrece más que un SO puro; viene con miles de [paquetes](#)⁽⁵⁾, programas precompilados distribuidos en un formato que hace más fácil la instalación en su ordenador. En este artículo se utilizará la rama Sarge de Debian.



2.2. Postfix.

El MTA (Mail Transportation Agent) [Postfix](#)⁽⁶⁾ pretende ser rápido, fácil de administrar y seguro, a la vez que suficientemente compatible con [Sendmail](#)⁽⁷⁾ como para que los usuarios existentes no se asusten. Por lo tanto, externamente mantiene el estilo de Sendmail, mientras que internamente es completamente diferente.

A diferencia de Sendmail, Postfix no es un programa monolítico, sino una [combinación de pequeños programas](#)⁽⁸⁾, cada uno de los cuales lleva a cabo una función especializada. En este documento, el lector encontrará la información necesaria para tener el sistema funcionando junto a otros componentes que completan la instalación de un sistema de correo electrónico. Puede encontrarse más información sobre Postfix en la [documentación online](#)⁽⁹⁾ de su website.

2.3. TLS.

Por defecto, toda comunicación en Internet se hace sin ningún tipo de cifrado y sin una autenticación fiable. Esto significa que cualquiera con acceso físico a la línea de datos por la que viaja un paquete puede espiar dicha comunicación. Aún peor, es posible redirigir o alterar esa comunicación para que la información que se desea mandar se pierda y nadie se dé cuenta.

De cara a solventar estos problemas de seguridad, [Netscape, Inc.](#)⁽¹⁰⁾ introdujo el protocolo [SSL](#)⁽¹¹⁾ (Secure Sockets Layer), que ha ido evolucionando en el protocolo estandarizado [TLS](#)⁽¹²⁾ (Transportation Layer Security). Ofrece tanto cifrado de la comunicación (frenando las escuchas) como autenticación fuerte (asegurando que ambas partes de una comunicación son correctamente identificadas y que la comunicación no puede ser alterada).

Postfix/TLS no implementa el protocolo TLS por sí mismo, sino que usa el paquete [OpenSSL](#)⁽¹³⁾ para esta tarea. En el website de OpenSSL pueden encontrarse enlaces a documentación que profundiza en el protocolo y sus características.

2.4. Cyrus IMAP.

[Cyrus IMAP](#)⁽¹⁴⁾ (Internet Message Access Protocol) es desarrollado y mantenido por el [Andrew Systems Group](#)⁽¹⁵⁾ de la [Carnegie Mellon University](#)⁽¹⁶⁾.

A diferencia de otros servidores IMAP, Cyrus usa su propio método para almacenar el correo de los usuarios. Cada mensaje es almacenado en su propio fichero. El beneficio de usar ficheros separados es una mayor fiabilidad ya que sólo un mensaje se pierde en caso de error del sistema de ficheros. Los metadatos, tales como el estado de un mensaje (leído, etc.) se almacenan en una base de datos. Además, los mensajes son indexados para mejorar el rendimiento de Cyrus, especialmente con muchos usuarios e ingentes cantidades de mensajes. No hay nada tan rápido como el servidor IMAP Cyrus.

Otra característica muy importante es que no son necesarias cuentas locales de Linux para cada usuario. Todos los usuarios son autenticados por el servidor IMAP. Esto lo convierte en una magnífica solución cuando se tiene una gran cantidad de usuarios.

La administración es llevada a cabo mediante comandos especiales de IMAP. Esto le permite usar tanto la interfaz de línea de comandos como los interfaces web. Este método es mucho más seguro que un interfaz web para */etc/passwd*.

Desde la versión 2.1 de Cyrus, se usa la versión 2 de la librería SASL para la autenticación. En la configuración descrita en este artículo se implementa una autenticación de tres capas. Cyrus se autentica con *saslauthd*, quien redirige la petición al mecanismo que le hayamos definido, por ejemplo *sasldb*, que buscará la información del usuario en una base de datos [Berkeley DB](#)⁽¹⁷⁾.

2.5. Cyrus SASL.

[SASL](#)⁽¹⁸⁾ son las siglas de *Simple Authentication and Security Layer*, un método para añadir soporte para la autenticación a protocolos basados en la conexión que ha sido estandarizado por la [IETF](#)⁽¹⁹⁾ (Internet Engineering Task Force). Se usa en servidores (en este caso Cyrus IMAP) para manejar las peticiones de autenticación de los clientes. Para ello, el protocolo incluye un comando para identificar y autenticar un usuario contra un servidor y para, opcionalmente, negociar la protección de las subsiguientes interacciones del protocolo. Si se negocia su uso, una capa de seguridad es añadida entre el protocolo y la conexión.



La [librería SASL de Cyrus](#)⁽²⁰⁾ también usa la librería OpenSSL para cifrar los datos. El lector encontrará más información en la página web de [Cyrus SASL](#)⁽²¹⁾.

2.6. LMTP.

SMTP (Simple Mail Transfer Protocol) y sus extensiones ESMTP (SMTP Service Extensions) proporcionan un mecanismo para transferir correo fiable y eficientemente. El diseño del protocolo SMTP requiere que el servidor maneje colas de envío de correo.

En ciertas circunstancias, fuera del área que engloba el intercambio entre hosts independientes en redes públicas, es deseable implementar un sistema donde el receptor del correo no maneje colas, como es el caso de un MDA (Mail Delivery Agent). Esto es precisamente lo que hace el protocolo [LMTP](#)⁽²²⁾ (Local Mail Transfer Protocol).

Aunque LMTP es una alternativa al protocolo ESMTP, usa (con algunos pequeños cambios) la sintaxis y la semántica de ESMTP. Este diseño permite al LMTP utilizar las extensiones definidas para el ESMTP. LMTP no debería ser nunca usado en el puerto 25.

2.7. Sieve.

[Sieve](#)⁽²³⁾ es un lenguaje que puede usarse para crear filtros de correo electrónico en el momento de la entrega final del correo (en el lado del servidor). No está ligado a ningún sistema operativo o servidor de correo en particular. Requiere el uso de la especificación de mensajes del RFC 822. El lenguaje es suficientemente potente para ser útil, pero está limitado de modo que permita la creación de sistemas de filtrado seguros en el lado del servidor. El objetivo es no permitir a los usuarios hacer nada más complejo (y peligroso) que escribir sencillos filtros de correo, además de facilitar editores basados en interfaces gráficas de usuario. El lenguaje no permite definir bucles o funciones, ni tampoco proporciona variables.

Se supone que el uso del lenguaje tiene lugar al final de la entrega, cuando el mensaje se mueve a una cuenta accesible por el usuario. En aquellos sistemas donde el MTA (Mail Transport Agent) realiza la entrega final (como es tradicional en los sistemas UNIX), es razonable clasificar cuando el MTA deposita el correo en la cuenta del usuario. Sin embargo, los filtros Sieve pueden ser usados por varios "puntos finales de entrega" del sistema de correo: por el servidor SMTP, por un servidor IMAP o POP que archive una o más cuentas de usuario, o por un cliente de correo (MUA, Mail User Agent) que actúe como gestor de las entregas (por ejemplo, un cliente POP o IMAP sin conexión).

2.8. Amavisd-new.

[Amavisd-new](#)⁽²⁴⁾ es un interfaz de alto rendimiento y fiabilidad entre el MTA y uno o más filtros de contenidos: antivirus o el módulo Mail::SpamAssassin de Perl. Está escrito en Perl, asegurando alta fiabilidad, portabilidad y facilidad de mantenimiento. Se comunica con el MTA via (E)SMTP o LMTP, o mediante el uso de otros programas. No existen problemas de sincronización en su diseño que pudieran causar pérdidas de correos.

Normalmente se posiciona dentro o cerca del gestor de correo principal, no necesariamente donde se ubiquen las cuentas de correo de los usuarios (donde tiene lugar el envío final). Si se está buscando una solución que soporte configuración por usuario y ratios de mensajes pequeñas que se ubique al final del proceso de envío (p.e. llamado desde procmail o en sustitución de un agente local de envío), posiblemente puedan encontrarse otras soluciones más apropiadas.

Cuando está habilitado el uso de Mail::SpamAssassin (SA), se llama a SA una sola vez por mensaje (independientemente del número de destinatarios). Amavisd-new se beneficia del uso del módulo de Perl Net::Server, el cuál ofrece un rápido entorno multihilo. Amavisd-new ofrece un servidor SMTP que cumple con el RFC 2821, un servidor LMTP que cumple con el RFC 2033, un cliente SMTP y genera notificaciones de estado de envío (o no) que cumplen los RFC 1892 y 1894. Esto lo hace adecuado para múltiples analizadores de virus y de correo publicitario en plataformas de correo donde la fiabilidad y el cumplimiento de los estándares son importantes.



2.9. SpamAssassin.

[SpamAssassin](#)⁽²⁵⁾ 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*, también conocido como correo electrónico comercial no solicitado. 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. También incluye soporte para informar de mensajes de *spam*, automática o manualmente, a bases de datos como [Vipul's Razor](#)⁽²⁶⁾.

2.10. Clam AntiVirus.

[ClamAV](#)⁽²⁷⁾ es una herramienta antivirus GPL para UNIX. El propósito principal de este software es la integración con los servidores de correo (escaneo de datos adjuntos). 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](#)⁽²⁸⁾ y [FreeBSD](#)⁽²⁹⁾, la detección de más de 20000 virus, gusanos y troyanos, el soporte integrado para archivos comprimidos con [Rar](#)⁽³⁰⁾, Zip, [Gzip](#)⁽³¹⁾ y [Bzip2](#)⁽³²⁾ y formatos de correo [Mbox](#)⁽³³⁾, [Maildir](#)⁽³⁴⁾ y ficheros crudos de correo.

2.11. Mailman.

[Mailman](#)⁽³⁵⁾ es un software libre que permite gestionar listas de distribución, noticias y correo electrónicos. Mailman está integrado con la web, permitiendo a sus usuarios una fácil administración de sus cuentas, así como a sus propietarios administrar las listas. Mailman incluye soporte para crear archivos de correos, procesamiento automático de correo rechazado, filtrado de contenido, envío en modo compendio o resumen, filtros de *spam*, etc.

2.12. SquirrelMail.

[SquirrelMail](#)⁽³⁶⁾ es un paquete de correo por web basado en estándares y escrito en [PHP](#)⁽³⁷⁾ 4. Incorpora soporte PHP para los protocolos IMAP y SMTP, y todas sus páginas se crean en puro [HTML 4.0](#)⁽³⁸⁾ (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.

[Avelsieve](#)⁽³⁹⁾ (SIEVE Mail Filters Plugin for SquirrelMail) es un [plugin para SquirrelMail](#)⁽⁴⁰⁾ que permite crear scripts hechos con Sieve en un servidor Cyrus IMAP que tenga habilitado el soporte para dicho lenguaje (Tim's SIEVE daemon). Avelsieve es parte de [Cyrusmaster](#)⁽⁴¹⁾, una herramienta de administración de Cyrus basada en web. Debería proporcionar una interfaz similar a la de los filtros de usuario a los administradores y personal de soporte técnico.

3. Versiones.

Las versiones del software usadas en este artículo son:

- Debian GNU/Linux Sarge
- Cyrus SASL: 2.1.18-1
- Cyrus IMAP: 2.1.18-1
- Postfix: 2.1.5-9
- SpamAssassin: 3.0.3-1



- ClamAV: 0.84-2
- Amavisd-new: 20030616p10-5
- Squirrelmail: 1.4.4-5
- Mailman: 2.1.5-8

4. Instalación y configuración de Cyrus IMAP y SASL.

La instalación de Cyrus es muy sencilla en Debian. Los paquetes necesarios para instalar Cyrus IMAP son `cyrus21-admin`, `cyrus21-common`, `cyrus21-doc` y `cyrus21-imapd`, y para Cyrus SASL son `libsasl2`, `sasl2-bin` y `libsasl2-modules`. Empezaremos por las librerías SASL y el servicio `saslauthd`. Por lo tanto, como `root`:

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

Editamos el fichero `/etc/default/saslauthd` y modificamos el parámetro `MECHANISMS`, de modo que el resultado final sea:

```
START=yes
MECHANISMS="sasldb"
```

E iniciamos el demonio `saslauthd` con el comando:

```
/etc/init.d/saslauthd start
```

Tal y como se especifica en `/usr/share/doc/sasl2-bin/README.Debian`, SASL2 usa `/dev/random`, lo cual podría bloquear una máquina con demasiadas conexiones. En caso de tener que usarlo en una máquina con mucha carga de trabajo, debería considerarse la instalación de una tarjeta de generación de entropía.

Los scripts de instalación del paquete `sasl2-bin` crean una base de datos de usuarios vacía, en formato Berkeley Database, que se encuentra en `/etc/sasldb2`. Este archivo pertenece al usuario `root` y al grupo `sasl`, por lo que todos los usuarios que quieran acceder a ella sin pasar por el servicio `saslauthd` tendrán que pertenecer a ese grupo. En la actualidad este servicio sólo soporta el método de autenticación `PLAIN`. Para poder usar `CRAM-MD5`, `DIGEST-MD5` y otros deberá usarse la librería `libsasl2` directamente. Y para que la librería SASL2 utilice el módulo de autenticación `sasldb` se han de configurar en cada programa que use esta librería las opciones `pwcheck_method=auxprop` y `auxprop_plugin=sasldb`, y añadir al usuario con que se ejecute el servicio al grupo `sasl`. Pero debido a que hay otras consideraciones que añaden excesiva complejidad a la configuración y que el método de autenticación `PLAIN` sobre canal cifrado con `TLS` es suficiente seguridad, el autor de este artículo se decanta por el uso de `saslauthd`.

El paquete `sasl2-bin` proporciona los ejecutables necesarios para la administración de la base de datos de usuarios, entre otros. Con el comando `sasldblistusers2` podemos listar los usuarios existentes, mientras que con el comando `saslpasswd2` podemos añadir y borrar usuarios. Los parámetros más utilizados son `-c` y `-d`, para añadir y borrar, respectivamente. Por ejemplo:

```
saslpasswd2 -c jsabater
```

crearía el usuario `jsabater`, tal y como podemos ver usando `sasldblistusers2` (`genma` es el hostname):

```
$ sasldblistusers2
jsabater@genma: userPassword
```

Para eliminar el usuario ejecutaríamos el comando:

```
saslpasswd2 -d jsabater
```

`saslpasswd2` ofrece asimismo la opción `-n`, de modo que podamos crear usuarios sin contraseña en texto plano. Es decir, obligamos a que se usen las claves específicas de cada mecanismo ([OTP](#)⁽⁴²⁾, [SRP](#)⁽⁴³⁾, etc), pero se va a prescindir de su uso porque es en la configuración de Cyrus IMAP donde se establecerá el nivel mínimo de autenticación requerido por todos los usuarios. Por último, la opción `-u` nos permite asociar usuarios a dominios. Por ejemplo:

```
saslpasswd2 -c jsabater -u linuxsilo.net
```



añadiría el usuario `jsabater@linuxsilo.net` a la base de datos.

```
$ sasldblistusers2
jsabater@genma: userPassword
jsabater@linuxsilo.net: userPassword
```

El siguiente paso es instalar el servidor Cyrus IMAP. Para ello ejecutaremos el siguiente comando:

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

Nótese que se omite la instalación del servidor POP3 de Cyrus. 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 a medida que vayamos avanzando en la configuración y el uso de los diversos protocolos (*TLS*, *LMTP*, mecanismos *SASL*, etc). 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:

1. `/etc/cyrus.conf`, donde se determinan los servicios que se ejecutarán al arrancar el sistema Cyrus;
2. `/etc/imapd.conf`, donde se especifican los parámetros de configuración del servicio *IMAP* de Cyrus.

Nótese también que la instalación de la plataforma Cyrus nos obliga a instalar el paquete `postfix` (en el caso de que hayamos desinstalado `exim` con anterioridad, el servidor de correo que viene por defecto en Debian), lo cual no es ningún inconveniente. Y precisamente éste es el único paquete que nos solicita configuración al instalarse, pero de momento lo pasaremos por alto con la opción "No configuration". Antes de continuar con la configuración, es muy recomendable que el lector se familiarice con las particularidades del sistema Cyrus, por lo que el autor le invita a leer, detenidamente y en varias ocasiones, la documentación disponible en `/usr/share/doc/cyrus21-doc`. En estos ficheros se encuentra la mayoría de la información necesaria para configurar tanto Cyrus IMAP como Postfix, y paulatinamente se hará uso de la misma a lo largo del artículo. Los documentos más relevantes son `README.Debian.gz`, `README.Debian.simpleinstall.gz` y `README.postfix.gz`.

Antes de continuar es conveniente asegurarse de que el grupo `sasl` tiene permisos de lectura sobre la base de datos de usuarios, `/etc/sasldb2`, así como que el usuario `cyrus` pertenece a este grupo. Los scripts del paquete `cyrus21-common` automáticamente nos dejan una configuración así, pero mejor cerciorarse al principio que provocar un tedioso proceso de depuración cuando haya múltiples elementos en juego.

4.1. El fichero `/etc/cyrus.conf`

Este fichero de configuración consta de tres partes claramente diferenciadas:

1. **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.
2. **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.
3. **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.

Es suficiente con realizar un único cambio sobre la configuración estándar que nos deja el mantenedor del paquete de GNU/Debian Linux, y es comentar la línea donde se declara la ejecución del servicio `pop3`. Asimismo, de momento no vamos a utilizar IMAP sobre SSL, así es que no tocaremos la línea que hace referencia a ese servicio. Por lo tanto, nos quedará un fichero de configuración tal que:

```
START {
    recover      cmd="/usr/sbin/ctl_cyrusdb -r"
    delprune     cmd="/usr/sbin/ctl_deliver -E 3"
    tlsprune     cmd="/usr/sbin/tls_prune"
}
SERVICES {
    imap         cmd="imapd -U 30" listen="imap" prefork=0 maxchild=100
    lmtpunix     cmd="lmtpd" listen="/var/run/cyrus/socket/lmtp" prefork=0
```



```

maxchild=20
    sieve      cmd="timsieved" listen="localhost:sieve" prefork=0 maxchild=100
    notify     cmd="notifyd" listen="/var/run/cyrus/socket/notify" proto="udp"
prefork=1
}
EVENTS {
    checkpoint cmd="/usr/sbin/ctl_cyrusdb -c" period=30
    delprune   cmd="/usr/sbin/ctl_deliver -E 3" at=0401
    tlsprune   cmd="/usr/sbin/tls_prune" at=0401
}

```

En estos momentos entra en escena una cuestión importante en la configuración: 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, muy habitual, 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. En un anexo de este documento se describe el proceso necesario para llevar a cabo este cambio de configuración.

Tras cualquier cambio en el fichero `/etc/cyrus.conf` es preciso reiniciar el servidor, tarea que podemos llevar a cabo cómodamente con el comando `/etc/init.d/cyrus21 restart`.

4.2. El fichero `/etc/imapd.conf`

`/etc/imapd.conf` es el fichero de configuración del servidor Cyrus IMAP y en él se definen los parámetros locales para IMAP. 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. Las líneas en blanco o que empiecen por `#` son ignoradas. 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*. Esta opción está documentada en el fichero `/usr/share/doc/cyrus21-doc/html/altnamespace.html`. En este artículo se usará el valor por defecto a *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). En este artículo se usará el valor a *yes* (basta con descomentar la línea).
- **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. Este usuario se va a autenticar mediante el método SASL, por lo que debe añadirse a la base de datos `/etc/sasldb2` mediante el comando `saslpasswd2`, tal que `$$saslpasswd2 -c cyrus`.
- **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.
- **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*.
- **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).
- **allowplaintext:** mediante esta opción decidimos si vamos a permitir uso del mecanismo de autenticación *sasl plain*. Es recomendable mantener el valor por defecto *yes* hasta que tengamos todo el sistema base configurado y funcionando. Más adelante en este artículo se explicará como usar otros métodos de autenticación más seguros.
- **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 pluggings existentes y para definir el orden de los mismos.



- **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 alto requiere algún tipo de cifrado. Se recomienda empezar con un valor de 0 (valor por defecto), que permite login de texto plano, en primera instancia. Una vez esté todo funcionando será más sencillo aumentar la seguridad. Más adelante en este artículo se cambiará este valor.
- **sasl_maximum_layer**: valor máximo del SSF que el servidor permitirá negociar al cliente. Es pertinente dejar el valor por defecto, 256 (no descomentar la línea).
- **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*.
- **lmtpsocket, idlesocket y notifysocket**: estas tres opciones especifican las rutas para los tres sockets UNIX que utiliza Cyrus. Los valores por defecto son correctos. Cabe destacar que el valor de *lmtpsocket* (por defecto */var/run/cyrus/socket/lmtp*) debe estar en consonancia con el especificado en el fichero */etc/cyrus.conf*.

El fichero */etc/imapd.conf* quedará como el que se presenta a continuación (al menos de momento, hasta que habilitemos el cifrado del canal, más adelante en este artículo):

```
configdirectory: /var/lib/cyrus
defaultpartition: default
partition-default: /var/spool/cyrus/mail
partition-news: /var/spool/cyrus/news
newsspool: /var/spool/news
altnamespace: no
unixhierarchysep: no
lmtp_downcase_rcpt: yes
admins: cyrus
allowanonymouslogin: no
popminpoll: 1
autocreatequota: 0
umask: 027
sieveusehomedir: false
sievedir: /var/spool/sieve
hashimapspool: true
allowplaintext: yes
sasl_mech_list: PLAIN
sasl_minimum_layer: 0
sasl_pwcheck_method: saslauthd
sasl_auxprop_plugin: sasldb
sasl_auto_transition: no
tls_ca_path: /etc/ssl/certs
tls_session_timeout: 1440
tls_cipher_list: TLSv1:SSLv3:SSLv2:!NULL:!EXPORT:!DES:!LOW:@STRENGTH
lmtpsocket: /var/run/cyrus/socket/lmtp
idlesocket: /var/run/cyrus/socket/idle
notifysocket: /var/run/cyrus/socket/notify
```

Tras cualquier cambio en el fichero */etc/imapd.conf* es necesario indicarle al servidor Cyrus que relea su contenido. Tenemos dos opciones para este cometido: */etc/init.d/cyrus21 restart* o */etc/init.d/cyrus21 reload*.

4.3. Perspectiva general, conceptos y administración del servidor Cyrus IMAP

Podemos acceder a la interfaz de comandos para la gestión del servidor Cyrus IMAP mediante una llamada a *cyradm*, tal que:

```
$cyradm --user cyrus localhost
Password:
localhost>
```

De todos modos, se recomienda la lectura completa de los apartados 4.3 y 4.4 antes de empezar a trabajar con él.



4.3.1. Espacio de nombres de los buzones de correo.

El servidor Cyrus IMAP presenta los buzones de correo usando la convención *netnews*. Si bien en los nombres de los buzones no se hace distinción entre mayúsculas y minúsculas, se mantiene el formato en el cual se creó. Un nombre de buzón no puede comenzar o terminar con punto, ni tampoco puede contener dos puntos seguidos. Todos los buzones de correo del usuario *jsabater* comienzan con la cadena *user.jsabater*. El buzón *user.jsabater* es un caso especial. Desde el punto de vista del usuario *jsabater*, el buzón *user.jsabater* es lo que se conoce como su *inbox* o *buzón de entrada*. Si la lista de control de acceso del buzón de correo *user.jsabater* permite a otros usuarios ver dicho buzón, aparecerá a esos otros usuarios como *user.jsabater*. En este documento se hace referencia al buzón de correo *user.jsabater* como el *inbox* de *jsabater*.

El administrador crea y borra usuarios al crear y borrar los *inbox* de esos usuarios. Si un usuario tiene *inbox*, entonces puede suscribirse a los buzones de correo. Sólo los usuarios sin puntos en su identificador de usuario poseen la capacidad de tener un *inbox* (un usuario con un punto en su identificador podría hacer login pero no sería capaz de recibir correo). Cuando un administrador borra el *inbox* de un usuario, todos los buzones de correo personales de ese usuario se borran también.

Con la notable excepción del *inbox*, todos los nombres de buzones de correo pertenecen al sistema, independientemente del sistema. Son las listas de control de acceso las que determinan qué usuarios tienen qué permisos en qué buzones.

4.3.2. Listas de control de acceso.

El acceso a cada buzón de correo se controla desde la lista de control de acceso de cada buzón. Las ACLs (del inglés, *Access Control List*), proporcionan un poderoso mecanismo para especificar los usuarios o grupos de usuarios que tienen permisos para acceder a los buzones de correo.

Una ACL es una lista de cero o más entradas. Cada entrada tiene un identificador y una serie de permisos. El identificador especifica el usuario o grupo de usuarios para el cuál la se aplica la entrada. El conjunto de permisos está formado por una o más letras o dígitos, cada uno de ellos otorgando un privilegio en particular.

Los permisos definidos 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.

El identificador es la parte en una ACL que especifica el usuario o grupo para el cuál se aplica. El significado del identificador habitualmente depende del mecanismo de autorización que se use. Con cualquiera que sea el mecanismo de autorización, existen siempre dos identificadores especiales. El identificador *anonymous* hace referencia al usuario anónimo, no autenticado. El identificador *anyone* hace referencia a cualquier usuario, incluyendo al usuario anónimo. En la versión 2.1.16 existen únicamente dos métodos de autorización, [Kerberos](#)⁽⁴⁴⁾ y grupos UNIX (*/etc/group*). Para una cantidad pequeña de usuarios y grupos, el uso del fichero de grupos de UNIX es una opción adecuada, aún haciendo referencia a cuentas de usuario UNIX que no existen. Para mantener grandes cantidades de usuarios, en un sistema grande y automatizado, obviamente la mejor opción es Kerberos. En cualquier caso, la versión 2.2.6 de Cyrus, no disponible a fecha de 16/07/2004 en Debian GNU/Linux, incluye nuevos mecanismos de autorización de grupos a los cuales el usuario que no use la distribución Debian GNU/Linux puede tener acceso.

Independientemente de la ACL de un buzón, los usuarios administradores (es decir, los que aparecen en la opción *admins* del fichero */etc/imapd.conf*) siempre tienen los permisos *l* y *a* sobre todos los buzones de manera implícita. Cuando se crea un buzón de correo, su ACL inicial es una copia de la ACL de su buzón padre. Cuando se crea un usuario, la ACL del *inbox* de ese usuario contiene una única entrada con todos los permisos garantizados para el propietario. Cuando se crea un buzón de



correo no asociado a un usuario que no tiene padre, su ACL se inicializa a valor de la opción *defaultacl* del fichero */etc/imapd.conf*.

La implementación de dominios virtuales de Cyrus soporta administradores de dominios así como administradores "globales" (multidominio). Los administradores de un dominio concreto se especifican con un identificador de usuario completo en la opción *admins* (p.e. *admin@linuxsilo.net*) y sólo tienen acceso a los buzones asociados al dominio. Los nombres de los buzones deben especificarse del mismo modo que en una configuración monodominio.

Los administradores globales se especifican con un identificador de usuario no cualificado en la opción *admins* y tienen acceso sobre cualquier buzón del servidor. Debido a que los administradores globales usan identificadores de usuario no cualificados, pertenecen al dominio por defecto, definido en la variable *defaultdomain* del mismo fichero */etc/imapd.conf*. Como resultado de esto, no pueden tenerse administradores globales sin especificar el *defaultdomain*. Nótese que, al tratar de hacer login como administrador global en un servidor multi-home desde una máquina remota, posiblemente sea necesario cualificar completamente el identificador de usuario con el *defaultdomain*.

Los administradores globales deben usar una sintaxis tipo *buzon@dominio.tld* al especificar buzones fuera del *defaultdomain*. Ejemplos al usar *cyradm*:

Para crear un *inbox* para el usuario *jsabater* en el *defaultdomain*:

```
cm user.jsabater
```

Para crear un *inbox* para el usuario *jsabater* en el dominio *linuxsilo.net*:

```
cm user.jsabater@linuxsilo.net
```

Para listar todos los buzones del dominio *linuxsilo.net*:

```
lm *@linuxsilo.net
```

4.3.3. Cuotas de usuario

El servidor Cyrus IMAP soporta cuotas de almacenamiento, lo que se define como el número de bytes de los mensajes RFC-822, en kilobytes. Cada mensaje se cuenta independientemente, incluso si el servidor es capaz de ahorrar espacio con el uso de enlaces duros a los ficheros de mensajes. El espacio adicional requerido para almacenar el índice del buzón y la caché de los ficheros no se tiene en cuenta para calcular la cuota.

Las cuotas se aplican a las cuotas raíz, que pueden estar en cualquier nivel de la jerarquía de buzones. Las cuotas raíz no es necesario que sean buzones. Las cuotas en una cuota raíz se aplican a la suma del uso de todo buzón que se encuentre al mismo nivel o por debajo en la jerarquía y que no esté bajo otra cuota raíz en la subjerarquía. Esto significa que cada buzón se ve limitado por una cuota raíz como máximo.

Por ejemplo, dados los buzones:

```
user.jsabater
user.jsabater.lists.bulmailing
user.jsabater.lists.postfix-users
user.jsabater.todo
user.jsabater.work
user.jsabater.work.customers
user.jsabater.work.suppliers
```

Y cuotas raíz en:

```
user.jsabater
user.jsabater.lists
user.jsabater.work
```



Entonces, la cuota raíz *user.jsabater* se aplicaría a los buzones *user.jsabater* y *user.jsabater.todo*, la cuota raíz *user.jsabater.lists* tendría efecto sobre los buzones *user.jsabater.lists*, *user.jsabater.lists.bulmailing* y *user.jsabater.lists.postfix-users* y la cuota raíz *user.jsabater.work* se aplicaría sobre los buzones *user.jsabater.work*, *user.jsabater.work.customers* y *user.jsabater.work.suppliers*.

Las cuotas raíz se crean mediante el comando *setquota* de la utilidad de consola *cyradm*. Las cuotas raíz no se pueden eliminar mediante comandos, sino que deben borrarse los ficheros pertinentes, tal y como se explica más adelante en este documento.

Normalmente, para que un mensaje pueda insertarse en un buzón, la cuota raíz de ese buzón debe tener suficiente espacio sin usar para que la inserción de dicho mensaje no provoque que la cuota se sobrepase. Pero el envío de correo es un caso especial. Para que un mensaje pueda ser entregado en un buzón, el uso de la cuota raíz del buzón no debe superior a lo permitido. Si el uso no sobrepasa el límite, entonces un mensaje puede ser entregado independientemente de su tamaño. Esto provoca que el buzón sobrepase la cuota, causando que el usuario sea informado de dicho problema y dándole la oportunidad de que lo arregle. Si no se permitiera la entrega en estos casos, el usuario no tendría manera práctica de saber que hubo correo que no pudo ser entregado.

Si el uso está por encima del límite, entonces el envío del correo fallará, retornando un error temporal. Esto hará que el sistema de envío reintente la entrega durante un par de días (permitiendo al usuario darse cuenta del problema y corregirlo) y luego devolverá el correo al remitente.

4.3.4. Procesos de recuperación de las bases de datos

Reconstrucción de los directorios de buzones. La base de datos más grande del sistema Cyrus son los directorios de buzones. Cada directorio de un buzón contiene los siguientes ficheros.

- **ficheros de mensajes.** Hay uno por mensaje, que contiene el mensaje en formato RFC 822. Las líneas del mensaje están separadas por *CRLF*, no sólo *LF*. El nombre de fichero del mensaje es el UID del mensaje seguido de un punto.
- **cyrus.header.** Este fichero contiene un número mágico e información de tamaño variable sobre el propio buzón de correo.
- **cyrus.index.** Este fichero contiene información de tamaño fijo sobre el propio buzón de correo y cada uno de los mensajes que contiene.
- **cyrus.cache.** Este fichero contiene información de tamaño variable sobre cada uno de los mensajes del buzón de correo.
- **cyrus.seen.** Este fichero contiene información de estado, de tamaño variable, sobre cada uno de los usuarios que han leído el buzón de correo y tenían el permiso *s*.

El programa *reconstruct* (*cyrreconstruct* en GNU/Debian Linux) puede usarse para reconstruir estos ficheros en caso de que estén corruptos. Siempre que puedan encontrarse ficheros de índice y de encabezados, se intentará salvaguardar la información en ellos existente y que no sea deducible de los propios ficheros de mensajes. El programa *reconstruct* intenta preservar los nombres de los flags, el estado de esos flags y la fecha interna. El resto de información se obtiene de los ficheros de mensajes. Así pues, el administrador de sistemas puede recuperar los datos de un disco dañado con tan sólo restaurar los ficheros de mensajes de una copia de seguridad y ejecutando *reconstruct* para generar de nuevo los ficheros de control. Nótese que el programa *reconstruct* no ajusta el uso de la cuota en los ficheros de cuotas raíz. Tras ejecutar *reconstruct*, es aconsejable ejecutar *quota -f* (descrito más abajo) para ajustar la cuota a partir de los ficheros de cuota raíz.

Reconstrucción de las cuotas raíz. El subdirectorío *quota* del directorio de configuración (especificado en la directiva *configdirectory*) contiene un fichero por cuota raíz, con el nombre de fichero indicando el nombre de la cuota raíz. Estos ficheros almacenan el uso de la cuota y los límites de cada cuota raíz. El programa *quota*, al ser invocado con el parámetro *-f*, recalcula la cuota raíz de cada buzón y el uso de cada cuota raíz. Para eliminar una cuota raíz, basta borrar el fichero de la cuota raíz. Luego ejecútese *quota -f* para devolver la consistencia a los ficheros de cuotas.

4.3.5. El fichero de buzones

El fichero de buzones del directorio de configuración es el fichero más crítico de todo el sistema Cyrus IMAP. Contiene una lista ordenada con cada buzón del servidor, así como los buzones cuotas raíz y las ACL. Debido a que la ACL es información crítica de seguridad, no puede ser reconstruida a partir de la información guardada en cualquier otro lado. No hay ningún programa para reconstruir un fichero de buzones dañado.



Para proteger los contenidos de los buzones, es recomendable realizar una copia de seguridad frecuentemente, incluso cada pocas horas, a otra parte del disco (u otro disco).

4.3.6. Suscripciones

El subdirectorio *user* del directorio de configuración contiene las suscripciones de los usuarios. Existe un fichero por usuario, con un nombre de fichero formado por el identificador de usuario y la extensión *.sub*. Cada fichero contiene una lista ordenada de los buzones a los que está suscrito. No existe programa alguno para recuperar ficheros de suscripciones dañados. Un sistema puede recuperar los ficheros perdidos simplemente restaurándolos de las copias de seguridad.

4.3.7. Logging

El subdirectorio *log* bajo el directorio de configuración permite a los administradores mantener logs por usuario. Si existe un subdirectorio de *log* con el nombre de un usuario, entonces los servidores IMAP y POP3 mantendrán un log de las sesiones que se han autenticado como ese usuario. El log de telemetría se guarda en el subdirectorio con el nombre de fichero del identificador de proceso del servidor y empieza con el primer comando tras la autenticación.

El servidor IMAP Cyrus envía mensajes de log al *syslog* local. Los niveles de gravedad usados son:

- **CRIT**: errores críticos que probablemente requieren acciones administrativas en la línea de comandos.
- **ERR**: errores de entrada y salida (E/S), incluyendo fallos al actualizar el uso de la cuota. El mensaje al *syslog* incluyen el error Unix y el fichero específicos.
- **WARNING**: fallos de mecanismos de protección, timeouts por inactividad del cliente.
- **NOTICE**: autenticaciones, correctas o incorrectas.
- **INFO**: aperturas de buzones de correo.

4.3.8. El directorio *proc*

El subdirectorio *proc* dentro del directorio de configuración contiene un fichero por cada proceso activo del servidor. El nombre de fichero es la representación ASCII del identificador de proceso y cada fichero contiene los siguientes campos, separados por tabuladores:

- nombre del host del cliente;
- login del usuario, si está conectado;
- buzón seleccionado, si hay alguno.

Normalmente el subdirectorio *proc* es purgado al reiniciar el servidor.

4.4. Configuración de 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, tal que */usr/bin/cyradm --user cyrus localhost*. El par usuario/contraseña será el definido en la base de datos de usuarios */etc/sasl2* mediante el programa *saslpasswd2*, tal y como se explica en el punto 4 de este artículo. Se recomienda usar el usuario *cyrus* como administrador, por lo que procederemos a darlo de alta:

```
saslpasswd2 -c cyrus
```

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

```
$sasldblistusers2
jsabater@genma: userPassword
cyrus@genma: userPassword
jsabater@linuxsilo.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 se comentan acto seguido:

Comando	Alias	Función	Sintaxis	Ejemplos
createmailbox	cm	Crear buzones de correo	cm <buzon>	cm user.jsabater
deletemailbox	dm	Borrar buzones de correo	dm <buzon>	dm user.jsabater
listacl	lam	Listar las ACL de un buzón	lam <buzon>	lam user.jsabater
setacl	sam	Establecer las ACL en un buzón	sam <buzon> <usuario> <permisos>	sam user.jsabater jsabater lrs sam user.jsabater jsabater all sam user.suppliers group:suppliers lrswipd
deleteacl	dam	Borrar las ACL de un buzón	dam <buzon> <usuario>	dam user.jsabater jsabater

Notas:

- La palabra clave *all* agrupa todos los permisos.
- Para asignar permisos a un grupo UNIX se usa la forma *group:nombre_grupo*.
- Un grupo UNIX debe existir en el fichero */etc/group*. Los usuarios que añadamos a ese grupo (editando manualmente el fichero) serán buzones de correo creados con el programa *cyradm*, no usuarios de sistema (aunque puede existir correspondencia, pero no tendrá relación alguna).
- Pese a que el usuario *cyrus* tiene permisos implícitos de administración sobre todos los buzones, es preciso hacerlos explícitos para ejecutar algunos comandos sobre ellos (por ejemplo, para borrarlos). Un comando del estilo `sam user.buzon cyrus all` solucionaría el problema, pues luego se heredarían los permisos en los subbuzones.

En este punto, podemos crear un buzón cualquiera con propósitos de prueba, tal que:

```
$cyradm --user cyrus localhost
Password:
localhost> cm user.jsabater
localhost> lam user.jsabater
jsabater lrswipcda
```

En este momento deberíamos ser capaces de dar de alta una cuenta de correo usando el protocolo IMAP en nuestro cliente de correo favorito y conectarnos al servidor, usando el usuario definido con *saslpasswd2*, al buzón del mismo nombre creado con *cyradm*. Asimismo, también podemos usar la herramienta *imtest* (*/usr/bin/imtest*), incluida en el paquete *cyrus21-clients* para comprobar su correcto funcionamiento. En el siguiente ejemplo se usa el mecanismo más básico de autenticación, que es *login*. Más adelante en este artículo se incrementará el nivel de seguridad pero, en este punto, es conveniente mantener la simplicidad al máximo para facilitar la comprensión de los pasos que se siguen.

```
$ imtest -a jsabater -w <contraseña> -m login localhost
S: * OK genma Cyrus IMAP4 v2.1.16-IPv6-Debian-2.1.16-6 server ready
C: C01 CAPABILITY
S: * CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ MAILBOX-REFERRALS NAMESPACE
UIDPLUS ID NO_ATOMIC_RENAME UNSELECT CHILDREN MULTIAPPEND SORT THREAD=ORDEREDSUBJECT
THREAD=REFERENCES IDLE AUTH=NTLM AUTH=DIGEST-MD5 AUTH=CRAM-MD5 LISTEXT
LIST-SUBSCRIBED ANNOTATEMORE
S: C01 OK Completed
C: L01 LOGIN jsabater {8}
S: + go ahead
C: <omitted>
S: L01 OK User logged in
Authenticated.
Security strength factor: 0
```

Pulsando *Ctrl+C* abandonaremos el programa de pruebas:



```
C: Q01 LOGOUT
Connection closed.
```

5. Instalación y configuración de Postfix.

Es muy posible que tengamos ya instalado el paquete *postfix*, pues el paquete *cyrus21-common* depende de él o de *mail-transport-agent*. En cualquier caso, ejecutaremos el siguiente comando como *root*:

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

Esto nos dejará instalados en el sistema todo lo necesario para la configuración posterior de Postfix. En el caso de que no tuviésemos el paquete *postfix* anteriormente, el script de postinstalación de este paquete nos mostrará unas pantallas que permiten configurar de un modo básico el servidor. Si ya estaba instalado el paquete, entonces deberemos llamar a ese asistente de modo manual mediante el comando:

```
dpkg-reconfigure postfix
```

No es imprescindible usarlo, pues en el artículo se incluyen los ficheros completos de configuración, pero es un buen punto de partida. Estas son las respuestas usadas para el servidor de este artículo:

1. **General type of configuration?** Internet Site
2. **Where should mail for root go?** jsabater
3. **Mail name?** linuxsilo.net
4. **Append .domain to simple addresses?** No
5. **Other destinations to accept mail for?** linuxsilo.net, genma.linuxsilo.net, localhost.linuxsilo.net, localhost
6. **Force synchronous updates on mail queue?** No
7. **Local networks?** 127.0.0.0/8
8. **Use procmail for local delivery?** No
9. **Mailbox size limit?** 0
10. **Local address extension character?** +

5.1. El fichero */etc/postfix/master.cf*

Usaremos el protocolo LMTP para la comunicación entre el MTA Postfix y Cyrus, pues es condición sine qua non para aprovechar las ventajas de Sieve. No es posible usar el transporte *cyrus* ni tampoco *procmail* y Sieve a la vez y, además, LMTP ofrece un rendimiento muy superior al uso directo de *cyrdeliver* para entregar el correo. 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 (un socket, a efectos de permisos, funciona igual que un 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)
# =====
lmtp           unix    -        -        n        -        -        lmtp
```

Nótese que el programa no se ejecuta en un *chroot*. En el caso de querer ejecutarlo en una jaula, el socket */var/run/cyrus/socket/lmtp* debería ser accesible desde dentro de la jaula, bien creando un enlace duro (*hard link*), bien modificando la ruta en el fichero de configuración */etc/cyrus.conf*.



5.2. El fichero `/etc/postfix/main.cf`

Para conseguir que Postfix entregue los correos a Cyrus a través de LMTP deberemos configurar un transporte en el primero. No es aconsejable usar *cyrdeliver*. La configuración del transporte en Postfix puede hacerse de diversas maneras (*default_transport*, *transport_maps* o *mailbox_transport*). En este artículo se usará *mailbox_transport*. Debido a que Postfix 2.x no pasa a minúsculas los destinatarios en las entregas por LMTP, es aconsejable usar la opción *lmtp_downcase_rcpt: yes* en el fichero `/etc/imapd.conf`. Para el uso de sockets Unix, el transporte de Postfix se especifica como *lmtp:unix:/var/run/cyrus/socket/lmtp* (en este ejemplo se usa la localización por defecto del socket de Cyrus en Debian, que se define en `/etc/cyrus.conf`).

Se necesita también un servicio *lmtpd* de Cyrus escuchando en ese socket, luego es conveniente asegurarse de que exista una línea como esta:

```
lmtpunix    cmd="lmtpd" listen="/var/run/cyrus/socket/lmtp" prefork=0 maxchild=20
```

en la sección *SERVICES* del fichero `/etc/cyrus.conf`. Asimismo, es imprescindible asegurarse de que tanto Cyrus como Postfix pueden hablarse a través de ese socket. Los sockets Unix funcionan igual que los ficheros, por lo que esto se traduce en que tanto el usuario *cyrus* como el usuario *postfix* pueden leer y escribir en ese fichero. Aviso: debido a que Cyrus preautentica cualquier cosa que proceda del socket Unix, cualquiera que pueda escribir en él será capaz de inyectar correo directamente en Cyrus.

Úsese *dpkg-statoverride* para asegurarse 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* y que si se quiere ejecutar dicho transporte en una jaula, el socket deberá encontrarse dentro de esa jaula. Para ello:

1. Cree un grupo llamado *lmtp*:
\$ `addgroup lmtp`
2. Agregue el usuario *postfix* a ese grupo:
\$ `adduser postfix lmtp`
3. Corrija los permisos del directorio del socket:
\$ `dpkg-statoverride --force --update --add cyrus lmtp 750 /var/run/cyrus/socket`
4. Reinicie Postfix y Cyrus:
\$ `/etc/init.d/postfix restart`
\$ `/etc/init.d/cyrus21 restart`

Las modificaciones mínimas necesarias en Postfix nos dejan un `/etc/postfix/main.cf` como éste:

```
setgid_group = postdrop
smtpd_banner = $myhostname ESMTP $mail_name (Debian/GNU)
biff = no
append_dot_mydomain = no
delay_warning_time = 4h
command_directory = /usr/sbin
daemon_directory = /usr/lib/postfix
program_directory = /usr/lib/postfix
myhostname = webmail.linuxsilo.net
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
mydestination = $myhostname, $mydomain, localhost.$mydomain, localhost
myorigin = $mydomain
mynetworks = 127.0.0.0/8
mailbox_size_limit = 0
recipient_delimiter = +
local_recipient_maps =

mailbox_transport = lmtp:unix:/var/run/cyrus/socket/lmtp
```

Donde `/etc/mailname` es:



```
linuxsilo.net
```

Donde */etc/hostname* es:

```
genma
```

Donde */etc/hosts* contiene:

```
127.0.0.1      localhost      localhost.localdomain
66.79.182.201 genma          genma.linuxsilo.net      webmail.linuxsilo.net
```

Y donde */etc/resolv.conf* contiene:

```
search linuxsilo.net
nameserver 66.79.182.201
```

Nótese que la directiva *local_recipient_maps* = evita que Postfix busque el nombre de usuario y su contraseña en los usuarios de sistema, pues su valor por defecto es *proxy:unix:passwd.byname \$alias_maps*.

Para comprobar el correcto funcionamiento de lo que tenemos hecho hasta el momento, primero nos aseguramos de que los servicios estén escuchando en los puertos correspondientes:

```
$ netstat -an|grep LISTEN
tcp        0      0 0.0.0.0:143          0.0.0.0:*           LISTEN
tcp        0      0 127.0.0.1:2000      0.0.0.0:*           LISTEN
tcp        0      0 0.0.0.0:25         0.0.0.0:*           LISTEN
```

El puerto 25 es de *smtp*, el 143 es de *imap* y el 2000 de *sieve*. Las correspondencias entre servicios y puertos pueden consultarse en el fichero */etc/services*. Acto seguido podemos mandar un correo electrónico a un buzón local desde de la propia máquina local o desde una exterior (en negrita los comandos que tecleamos nosotros, *genma* es el hostname):

```
$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 webmail.linuxsilo.net ESMTP Postfix (Debian/GNU)
HELO localhost
250 webmail.linuxsilo.net
MAIL FROM: <jsabater@genma>
250 Ok
RCPT TO: <jsabater@genma>
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.
```

En estos momentos podemos ver que el correo ha sido enviado al buzón *jsabater@genma* ejecutando el comando:

```
$ ls /var/spool/cyrus/mail/j/user/jsabater/
1. cyrus.cache  cyrus.header  cyrus.index
```

O, por supuesto, conectándonos mediante un cliente de correo que soporte IMAP y visualizando el correo. Nótese que si modificamos el fichero */etc/postfix/master.cf* y en la línea donde se establece el *smtp*:



```
smtp      inet  n       -       -       -       -       smtpd
```

añadimos el parámetro `-v` podremos ver un log del servidor de correo mucho más detallado (particularmente, recomiendo `tail -f /var/log/mail.log | colorize`, previa instalación del paquete `colorize`). La línea quedaría tal que:

```
smtp      inet  n       -       -       -       -       smtpd -v
```

A continuación vamos a activar el uso de SASL en Postfix. El objetivo es autenticar los clientes `smtp` para que puedan hacer relay a través del servidor de correo. Para ello se modifica la opción `smtpd_recipient_restrictions` del fichero `/etc/postfix/main.cf`:

```
smtp_sasl_auth_enable = no
smtpd_sasl_auth_enable = yes
smtpd_sasl_local_domain = genma
smtpd_recipient_restrictions =
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_unauth_destination
smtpd_sasl_security_options = noanonymous
```

De este modo, se permite hacer relay a los clientes sin autenticar que pertenezcan a las redes indicadas en `mynetworks` (habitualmente la red local) y a los clientes autenticados mediante el método SASL. Se rechazarán los que están sin autenticar. Además, podríamos requerir que fuese necesario utilizar TLS para autenticarse (`smtpd_tls_auth_only = yes`), pero eso lo haremos más adelante. Opcionalmente, mediante la opción `smtpd_sasl_local_domain=genma` se indica que compruebe el usuario entrando en `/etc/sasl2` con `usuario@genma`.

Para que Postfix sepa qué mecanismo usar a través de SASL hay que crear el fichero `/etc/postfix/sasl/smtpd.conf` con las siguientes líneas:

```
pwcheck_method: saslauthd
mech_list: plain login
```

Debido a que Postfix se ejecuta por defecto en una jaula (`chrooted`), localizada en `/var/spool/postfix`, no puede acceder al socket y demás ficheros del demonio de autenticación SASL, que se encuentran en `/var/run/saslauthd`. Por lo tanto, Postfix va a tratar de encontrarlos en `/var/spool/postfix/var/run/saslauthd/`. Entonces, tenemos dos opciones:

1. No ejecutar Postfix en un entorno chroot. Para ello, editamos el fichero `/etc/postfix/master.cf` y dejamos la línea del `smtp` tal y como esta:

```
# =====
# service type private unpriv chroot wakeup maxproc command + args
#               (yes)   (yes)   (yes)   (never) (100)
# =====
smtp      inet  n       -       n       -       -       smtpd
```

Ejecutar Postfix, o cualquier otro servicio, enjaulado es una medida añadida de seguridad, pero la programación orientada a la seguridad de Postfix lo hacen ya de por sí un servidor muy seguro. Por otra parte, esta opción simplifica el mantenimiento, y eso siempre es bueno. Por lo tanto, no es una opción descartable en absoluto.

2. Usar enlaces duros. Siempre que el fichero original y el enlazado estén en la misma partición, podemos crear un *hard link* entre `/var/run/saslauthd` y `/var/spool/postfix/var/run/saslauthd`. Pero, debido a que estos ficheros son sobrescritos cada vez que se inicia el demonio `saslauthd`, debemos asegurarnos de que se vuelven a enlazar. Para ello, modificaremos el script de arranque en `/etc/init.d/saslauthd` y lo dejaremos como el que sigue (en negrita los cambios realizados):

```
#!/bin/sh -e

NAME=saslauthd
DAEMON="/usr/sbin/${NAME}"
DESC="SASL Authentication Daemon"
DEFAULTS=/etc/default/saslauthd
```



```

PWDIR=/var/run/saslauthd
PIDFILE="/var/run/${NAME}/saslauthd.pid"

mklink() {
    sleep 1
    cd /var/spool/postfix/var/run/saslauthd/
    ln /var/run/saslauthd/* .
    echo "Links inside the Postfix jail have been created."
}

rmlink() {
    rm -f /var/spool/postfix/var/run/saslauthd/*
    echo "Links inside the Postfix jail have been removed."
}

createdir() {
    # $1 = user
    # $2 = group
    # $3 = permissions (octal)
    # $4 = path to directory
    [ -d "$4" ] || mkdir -p "$4"
    chown -c -h "$1:$2" "$4"
    chmod -c "$3" "$4"
}

test -f "${DAEMON}" || exit 0

# Source defaults file; edit that file to configure this script.
if [ -e "${DEFAULTS}" ]; then
    . "${DEFAULTS}"
fi

# If we're not to start the daemon, simply exit
if [ "${START}" != "yes" ]; then
    exit 0
fi

# If we have no mechanisms defined
if [ "x${MECHANISMS}" = "x" ]; then
    echo "You need to configure ${DEFAULTS} with mechanisms to be used"
    exit 0
fi

# Add our mechanisms with the necessary flag
PARAMS="${PARAMS} -a ${MECHANISMS}"

START="--start --quiet --pidfile ${PIDFILE} --startas ${DAEMON} --name ${NAME} --
${PARAMS}"

# Consider our options
case "${1}" in
    start)
        echo -n "Starting ${DESC}: "
        dir=`dpkg-statoverride --list $PWDIR`
        test -z "$dir" || createdir $dir
        if start-stop-daemon ${START} >/dev/null 2>&1 ; then
            echo "${NAME}."
        else
            if start-stop-daemon --test ${START} >/dev/null 2>&1; then

```



```

        echo "(failed)."
```

```

        exit 1
    else
        echo "${DAEMON} already running."
        exit 0
    fi
fi
mklinks
;;
stop)
    echo -n "Stopping ${DESC}: "
    if start-stop-daemon --stop --quiet --pidfile "${PIDFILE}" \
        --startas ${DAEMON} --retry 10 --name ${NAME} \
        >/dev/null 2>&1 ; then
        echo "${NAME}."
    else
        if start-stop-daemon --test ${START} >/dev/null 2>&1 ; then
            echo "(not running)."
```

```

            exit 0
        else
            echo "(failed)."
```

```

            exit 1
        fi
    fi
rmlinks
;;
restart|force-reload)
    $0 stop
    exec $0 start
;;
*)
    echo "Usage: /etc/init.d/${NAME} {start|stop|restart|force-reload}" >&2
    exit 1
;;
esac

exit 0
```

De este modo los enlaces se alterarán adecuadamente cada vez que se pare, inicie o reinicie el demonio. Aunque antes es preciso crear el directorio donde se harán los enlaces en el interior de la jaula de Postfix:

```

mkdir -p /var/spool/postfix/var/run/saslauthd
chown root.sasl /var/spool/postfix/var/run/saslauthd
```

Tras reiniciar el servidor de autenticación (*/etc/init.d/saslauthd restart*) y el de correo (*/etc/init.d/postfix restart*) ya podemos comprobar el buen funcionamiento de la autenticación SASL. Primero tendremos que generar la cadena alfanumérica que nos autenticará durante el proceso de comunicación. Para ello necesitaremos soporte para [Perl](#)⁽⁴⁵⁾ y nuestro nombre de usuario y contraseña definidos en la base de datos */etc/sasldb2*. Ejecutamos entonces:

```
perl -MMIME::Base64 -e 'print encode_base64("username\0username\0password");'
```

sustituyendo *username* por nuestro identificador de usuario en las dos ocasiones y *password* por nuestra contraseña. En la línea siguiente aparecerá una cadena alfanumérica que usaremos en seguida. Por ejemplo, para la combinación *jsabater/jsabater* sería *anNhYmFOZXIAanNhYmFOZXIAanNhYmFOZXI=*. A continuación se muestra la comprobación del buen funcionamiento usando autenticación *PLAIN* y la cadena obtenida como ejemplo (en negrita lo que nosotros tecleamos):

```

$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
```



```

Escape character is '^]'.
220 webmail.linuxsilo.net ESMTP Postfix (Debian/GNU)
EHLO localhost
250-webmail.linuxsilo.net
250-PIPELINING
250-SIZE 10240000
250-VERFY
250-ETRN
250-AUTH PLAIN
250 8BITMIME
AUTH PLAIN anNhYmF0ZXIAanNhYmF0ZXIAanNhYmF0ZXI=
235 Authentication successful
QUIT
221 Bye
Connection closed by foreign host.

```

5.3. Autenticación y autorización.

La autenticación es el mecanismo por el cual los sistemas pueden identificar con seguridad a los usuarios. Los sistemas de autenticación proporcionan respuestas a las siguientes preguntas:

- ¿Quién es el usuario?
- ¿Es el usuario quien realmente pretende ser?

Un método de autenticación puede ser tan sencillo (e inseguro) como una contraseña en texto plano (servidores de FTP antiguos) o tan complicados como el sistema Kerberos. En cualquier caso, sin embargo, los sistemas de autenticación dependen de alguna pequeña porción de información conocida (o disponible) sólo por el individuo que está autenticándose y el sistema de autenticación (un secreto compartido). Esa información puede ser la clásica contraseña, algún tipo de propiedad física del individuo (huellas dactilares, patrón de retina, etc), o algunos datos derivados (como es el caso de los llamados sistemas de tarjeta inteligente – smartcard). De modo que pueda verificarse la identidad de un usuario, el sistema de autenticación típicamente solicita al usuario que proporcione esa información única (su contraseña, sus huellas, etc.) y, si el sistema de autenticación puede verificar que ese secreto compartido se ha presentado correctamente, entonces el usuario se considera autenticado.

La autorización, en contraste, es el mecanismo por el cual un sistema determina qué nivel de acceso un usuario particular que ya se ha autenticado debería tener sobre los recursos controlados por el sistema. Por ejemplo, un sistema de gestión de bases de datos puede estar diseñado para proporcionar a ciertos individuos la capacidad de recuperar información de la base de datos pero no la capacidad de cambiar los datos almacenados, mientras que podría dar a otros individuos esa capacidad. Los sistemas de autorización responden a las siguientes preguntas:

- ¿Está autorizado el usuario X a acceder al recurso R?
- ¿Está autorizado el usuario X a realizar la operación P?
- ¿Está autorizado el usuario X a realizar la operación P sobre el recurso R?

5.4. Alias virtuales.

La mayoría de los servidores de correo son el destino final de unos pocos dominios. Estos incluyen los hostnames y las [direcciones IP] de la máquina en la que se ejecuta Postfix, y en ocasiones también el dominio padre del hostname. En adelante se considerarán dominios canónicos estos dominios. En este artículo, el servidor tiene como dominios canónicos *localhost*, *localhost.linuxsilo.net*, *genma.linuxsilo.net* y *linuxsilo.net*, aunque realmente sólo vayan a usarse el primero y el último (los dos de en medio son equivalentes al primero).

Además de los dominios canónicos, Postfix puede configurarse para ser el destino final de otros dominios adicionales. A estos dominios se les llama hospedados, pues no tienen relación directa con el nombre de la máquina. Los dominios hospedados se implementan habitualmente con la directiva *virtual_alias_domain* o con la directiva *virtual_mailbox_domain*. Asimismo, Postfix puede configurarse como servidor secundario o de seguridad en un registro MX de DNS. En este caso, Postfix no es el destino final de estos dominios, sino que tan sólo guarda temporalmente el correo del servidor principal cuando éste está caído,



y se lo reenvía cuando vuelve a estar disponible. Esta función se implementa con la directiva *relay_domain*.

Siguiendo la línea del artículo, interesa configurar el servidor para que acepte correo de direcciones de dominios que no son el principal y lo redirija a cuentas locales. Por ejemplo, dado otro dominio de nuestra propiedad, *bsdsilo.net*, tenemos una dirección de contacto para ese dominio que es *info@bsdsilo.net* y queremos que el correo se reciba en *jsabater@linuxsilo.net*. Esto se consigue usando la directiva *virtual_alias_maps*. Entonces, en el fichero */etc/postfix/main.cf* añadiríamos las líneas:

```
virtual_alias_maps = hash:/etc/postfix/virtual
virtual_alias_domains = /etc/postfix/vdomains
```

Y crearíamos el fichero */etc/postfix/virtual* tal que:

```
info@bsdsilo.net    jsabater
```

Y el fichero */etc/postfix/vdomains* tal que:

```
bsdsilo.net
```

De esta manera, el correo recibido para el alias virtual *info@bsdsilo.net* sería reenviado al buzón local *jsabater*, que equivale a *jsabater@linuxsilo.net*. Si dejamos la configuración tal que así, cualquier correo que vaya para una dirección diferente a *info@bsdsilo.net* será rechazado con el clásico mensaje de error *550-Mailbox unknown*. Si deseamos recoger todo el correo "perdido", podemos establecer una sentencia que en inglés se conoce como *catch-all*, tal que:

```
@bsdsilo.net    jsabater
```

5.5. Dominios virtuales.

Una opción muy común en la configuración de servidores de correo electrónico es hospedar las cuentas de correo de varios dominios. Estos dominios son, efectivamente, tratados como dominios virtuales pero, a diferencia de los alias virtuales, es preciso disponer de un buzón de correo para las cuentas de cada dominio. El uso de dominios virtuales y alias virtuales es complementario, no exclusivo. Siguiendo las pautas establecidas anteriormente en este artículo, un ejemplo de configuración del fichero *main.cf* sería el siguiente:

```
virtual_transport = lmtp:unix:/var/run/cyrus/socket/lmtp
virtual_mailbox_domains = bsdsilo.net
virtual_mailbox_maps = hash:/etc/postfix/vmailbox
virtual_alias_maps = hash:/etc/postfix/virtual
```

Donde */etc/postfix/vmailbox* sería:

```
info@bsdsilo.net    jsabater
sales@bsdsilo.net   rgalli
support@bsdsilo.net asimon
# Descoméntese la siguiente línea para obtener un efecto "catch-all"
# @bsdsilo.net      asimon
```

Y donde */etc/postfix/virtual* sería:

```
postmaster@bsdsilo.net    postmaster
```

La directiva *virtual_mailbox_domains* le dice a Postfix que *bsdsilo.net* debe procesarse a través del transporte especificado en el valor de *virtual_transport*, en este caso el socket UNIX de Cyrus. Si se omite la configuración de *virtual_mailbox_domains* tanto puede ocurrir que Postfix rechace el correo (*relay access denied*) como que no sea capaz de enviarlo (*mail for bsdsilo.net loops back to myself*).

Nunca debe listarse un dominio virtual de *virtual_mailbox_domain* en *mydestination*. Y nunca debe listarse un dominio virtual de *virtual_mailbox_domain* como alias de dominio virtual en *virtual_alias_maps*.



El parámetro *virtual_mailbox_maps* especifica una tabla de búsqueda con todos los destinatarios válidos. En el ejemplo de más arriba, *info@bsdsilo.net*, *sales@bsdsilo.net* y *support@bsdsilo.net* se listan como direcciones válidas, mientras que el correo dirigido a cualquier otro destinatario será rechazado con el mensaje de error *User unknown*. Por supuesto, tal y como puede apreciarse por el comentario en el ejemplo, puede crearse una redirección para todo el correo que no tenga como destinatario alguno de los buzones especificados anteriormente para ese dominio.

Tal y como se explica en el punto 4 de este artículo, deberá crearse una cuenta en la base de datos de usuarios para estos buzones, tal que:

```
$ saslpasswd2 -c jsabater -u bsdsilo.net
$ saslpasswd2 -c rgalli -u bsdsilo.net
$ saslpasswd2 -c asimon -u bsdsilo.net
```

Además, deberán crearse también los buzones:

```
$ cyradm --user cyrus localhost
localhost> cm user.jsabater@bsdsilo.net
localhost> cm user.rgalli@bsdsilo.net
localhost> cm user.asimon@bsdsilo.net
localhost> quit
```

Por último, tal y como puede apreciarse en el ejemplo, es posible mezclar alias virtuales y buzones virtuales. En este caso se está usando la directiva *virtual_alias_maps* para redirigir la cuenta del postmaster del dominio *bsdsilo.net* al postmaster local. Puede usarse el mismo mecanismo para redirigir cualquier dirección a un buzón local o a una dirección remota.

Nota: este ejemplo asume que, en el *main.cf*, *\$myorigin* aparece listado en los valores de *mydestination*. Si no es así, debe especificarse un valor de dominio explícito en la parte derecha de las entradas de la tabla de alias virtuales o el correo será enviado al dominio incorrecto.

Cada vez que se modifiquen los ficheros */etc/postfix/vmailbox* o */etc/postfix/virtual* debe ejecutarse el comando *postmap /etc/postfix/vmailbox* o *postmap /etc/postfix/virtual*, respectivamente. Si se cambia el */etc/postfix/main.cf* debe ejecutarse */etc/init.d/postfix reload*.

6. Cifrado del canal de comunicación usando TLS.

Cuando se está estableciendo una conexión TLS, la máquina que establece la conexión tiene que validarse a sí mismo. Esto es debido a que alguien podría interceptar la comunicación y establecer una conexión cifrada. La máquina remota posiblemente no se daría cuenta y pasaría la información con normalidad. Por ello, los certificados se usan para proporcionar información única que prueba que la máquina que está cifrando la comunicación es realmente aquella con la cuál nuestra máquina quiere hablar.

Sería correcto pensar que cualquier servidor malicioso podría emitir un certificado si no fuera por el pequeño detalle que falta mencionar: cada certificado que es emitido proporciona información sobre una autoridad que dará validez el certificado enviado al establecer una conexión TLS.

Añadir cifrado al canal de transmisión es muy sencillo una vez tenemos tanto Postfix como Cyrus configurados adecuadamente, según se ha descrito en los puntos anteriores. El primer paso será instalar el paquete *openssl*:

```
apt-get install openssl
```

6.1. La generación de los certificados.

Si se pretende ofrecer servicios de correo electrónico a terceros, es más que recomendable usar un certificado que esté firmado por una autoridad certificadora (CA, del inglés *Certificate Authority*) reconocida, como [Verisign](#)⁽⁴⁶⁾ o [Thawte](#)⁽⁴⁷⁾. Pero si sólo se va a usar en la red local de una empresa pequeña o mediana, o simplemente se quiere disfrutar de las comunicaciones cifradas a título personal, posiblemente sea suficiente crear una autoridad certificadora nosotros mismos y firmar con ella el



certificado. En cualquier caso, ambas soluciones son igual de seguras; es una cuestión de formalidad ante el cliente o usuario. El proceso básico a seguir se resume en los siguientes puntos:

1. Crear el certificado.
2. Una autoridad certificadora debe firmar el certificado.
3. Instalar el certificado.

Para crear los certificados pueden seguirse los pasos descritos en el [Lutz's very short course on being your own CA](#)⁽⁴⁸⁾ o en el [Postfix SMTP AUTH \(and TLS\) HOWTO](#)⁽⁴⁹⁾. En estos tutoriales se explica, además, como actuar como autoridad certificadora y poder firmar así los certificados. Básicamente, los pasos a seguir son tres:

1. Crear una nueva autoridad certificadora: `/usr/lib/ssl/misc/CA.pl -newca`
2. Realizar la petición de un certificado: `/usr/lib/ssl/misc/CA.pl -newreq-nodes`
3. Firmar el certificado: `/usr/lib/ssl/misc/CA.pl -sign`

Con la salvedad de que no debemos añadir una palabra de paso al certificado para que el servidor no se quede bloqueado esperándola al iniciarse. Luego deberemos copiar tres de los ficheros resultantes del proceso al subdirectorio `/etc/postfix/ssl/`:

- **cacert.pem**: el certificado de la autoridad certificadora, al cual se remitirá al cliente cuando quiera comprobar la autenticidad del certificado que le ha enviado nuestro servidor Postfix.
- **newcert.pem**: el certificado público que enviaremos al cliente para establecer la comunicación segura.
- **newreq.pem**: el certificado privado que almacenaremos en el servidor y del cual la parte realmente importante es la clave, que debe permanecer secreta.

Ejecutaremos entonces los siguientes comandos:

```
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
```

Una solución intermedia entre crear nuestra propia autoridad certificadora y el pago de una ingente cantidad de dinero a una reconocida comercialmente, es el uso de una llevada por la comunidad que emita certificados al público de manera gratuita, por ejemplo [CAcert.org](#)⁽⁵⁰⁾. Si solicitamos el alta en su página web, añadimos nuestro dominio y solicitamos que nos firme el CSR (Certificate Signing Request), tan sólo deberemos entonces instalar su [certificado raíz](#)⁽⁵¹⁾ en cada máquina para evitar los mensajes de aviso de los clientes de correo. Más información en la [documentación de CAcert.org](#)⁽⁵²⁾.

6.2. Modificaciones en Postfix.

Tan sólo es necesario editar el fichero `/etc/postfix/main.cf` y agregar las siguientes líneas:

```
smtpd_use_tls = yes
# smtpd_tls_auth_only = yes
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

smtpd_tls_loglevel = 3
smtpd_tls_received_header = yes
smtpd_tls_session_cache_timeout = 3600s
tls_random_source = dev:/dev/urandom
```

De este modo restringimos que las autenticaciones al envío se puedan hacer únicamente usando TLS (esta línea aparece comentada de momento). Además, obligamos a que todas las comunicaciones con el demonio `smtpd` se hagan a través de TLS. Tras reiniciar el servidor Postfix, ya podemos comprobar el correcto funcionamiento del mismo:



```

$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 webmail.linuxsilo.net ESMTP Postfix (Debian/GNU)
EHLO localhost
250-webmail.linuxsilo.net
250-PIPELINING
250-SIZE 10240000
250-VERFY
250-ETRN
250-STARTTLS
250-AUTH LOGIN PLAIN
250-AUTH=LOGIN PLAIN
250 8BITMIME
STARTTLS
220 Ready to start TLS
QUIT
QUIT
Connection closed by foreign host.

```

Entonces, si queremos asegurarnos de que las autenticaciones se realicen únicamente sobre un canal cifrado mediante el protocolo TLS, debemos descomentar la línea que reza `smtpd_tls_auth_only = yes`. Tras reiniciar el servidor, podemos observar las diferencias:

```

$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 webmail.linuxsilo.net ESMTP Postfix (Debian/GNU)
EHLO localhost
250-webmail.linuxsilo.net
250-PIPELINING
250-SIZE 10240000
250-VERFY
250-ETRN
250-STARTTLS
250 8BITMIME
STARTTLS
220 Ready to start TLS
QUIT
QUIT
Connection closed by foreign host.

```

Nótese que el servidor no ofrece los métodos de autenticación antes de que se haya establecido el canal seguro. Asimismo, una vez que nos hemos cerciorado de que todo funciona adecuadamente, podemos reducir el nivel de logging cambiando el valor 3 por 1 en la opción `smtpd_tls_loglevel`.

6.3. Modificaciones en Cyrus IMAP.

Para activar el cifrado del canal de comunicaciones en Cyrus IMAP deberemos modificar dos ficheros, `/etc/cyrus.conf` y `/etc/imapd.conf`. En el primero añadiremos el servicio `imaps` a la lista de los que deben iniciarse con el sistema Cyrus, mientras que en el segundo configuraremos varios parámetros relacionados con TLS y la localización de los certificados en el disco duro. Entonces, en la sección `SERVICES` del fichero `/etc/cyrus.conf` tan sólo debemos descomentar una línea, de modo que quede como la que sigue:

```
imaps          cmd="imapd -s -U 30" listen="imaps" prefork=0 maxchild=100
```



Y cambiar la que antes teníamos activa para soportar IMAP sin SSL solo a través de la interfaz *localhost* (necesario para usar *cyradm*), de modo que forcemos al usuario a usar un canal cifrado:

```
imap          cmd="imapd -U 30" listen="localhost:imap" prefork=0 maxchild=100
```

Las líneas a descomentar o modificar del fichero */etc/imapd.conf* son las siguientes:

```
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
```

Puede apreciarse que la parte privada del certificado se encuentra en un directorio específicamente creado para tal propósito, */var/imap*. Esto es debido a que el usuario *cyrus* debe tener permisos de lectura sobre él, mientras que su ubicación estándar, que sería */etc/ssl/private* no nos permite esa posibilidad. En este artículo se ha usado el mismo certificado para Postfix que para Cyrus, siendo necesario ejecutar los siguientes comandos:

```
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
```

Téngase en cuenta que, tal y como hemos configurado el servidor Postfix (*smtpd*), si al enviar un correo lo hacemos a un servidor que soporta (ofrece) TLS sobre el protocolo *smtp*, nuestro servidor tratará de usar cifrado sobre el canal. Por lo tanto, es muy conveniente, por no decir imprescindible, tener un certificado firmado por una autoridad certificadora reconocida. Tras reiniciar el servidor Cyrus (*/etc/init.d/cyrus21 restart*) podemos comprobar que los cambios han surgido efecto:

```
$ imtest -a jsabater -w <contraseña> -m login -s localhost
verify error:num=20:unable to get local issuer certificate
verify error:num=21:unable to verify the first certificate
TLS connection established: TLSv1 with cipher AES256-SHA (256/256 bits)
S: * OK genma Cyrus IMAP4 v2.1.16-IPv6-Debian-2.1.16-6 server ready
C: C01 CAPABILITY
S: * CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ MAILBOX-REFERRALS NAMESPACE
UIDPLUS ID NO_ATOMIC_RENAME UNSELECT CHILDREN MULTIAPPEND SORT THREAD=ORDEREDSUBJECT
THREAD=REFERENCES IDLE AUTH=LOGIN AUTH=PLAIN LISTEXT LIST-SUBSCRIBED ANNOTATEMORE
S: C01 OK Completed
C: L01 LOGIN jsabater {8}
S: + go ahead
C: <omitted>
S: L01 OK User logged in
Authenticated.
Security strength factor: 256
```

Pulsando Ctrl+C abandonaremos el programa de pruebas:

```
C: Q01 LOGOUT
Connection closed.
```

[CAcert](#)⁽⁵⁰⁾ es una organización certificadora sin ánimo de lucro que pretende promocionar el conocimiento y la educación en la seguridad informática a través del uso de cifrado, especialmente la familia X.509 de estándares. Esta organización puede firmar nuestro certificado digital, pero aún no es una autoridad certificadora reconocida, por lo cual será siempre necesario instalar su [certificado raíz](#)⁽⁵³⁾ en la aplicación cliente. En cualquier caso, en la actualidad no hay diferencia entre usar un certificado



firmado por esta asociación o por una autoridad certificadora creada por nosotros mismos, pero quizás en el futuro esto cambie.

7. El uso de Sieve.

Para poder sacar partido al uso de scripts en el lado del servidor es preciso utilizar LMTP. Para activar un script de Sieve en el servidor es preciso ejecutar un comando en el mismo servidor, a menos que se use algún tipo de interfaz gráfica que agilice la tarea. Cada usuario puede subir y activar los scripts generados por el mismo (siempre y cuando tenga acceso con una cuenta en el servidor). Por ejemplo, dado el script *jsabater.sieve*, ejecutaríamos los siguientes comandos desde el directorio donde se encuentre el script:

```
$ sieveshell -u jsabater localhost
connecting to localhost
Please enter your password:
> put jsabater.sieve
> activate jsabater.sieve
> ls
jsabater.sieve <- active script
> quit
```

Acto seguido se muestra un script de ejemplo que clasifica el correo entrante en varias subcarpetas, usando las cabeceras propias de algunas listas de correo, el remitente del mensaje o el asunto del mismo.

```
require "fileinto";
if header :contains "List-Id" "bulmailing.bulma.net" {
    fileinto "Inbox.Bulmailing";
}
elsif header :contains "List-Post" "postfix-users@postfix.org" {
    fileinto "Inbox.Postfix";
}
elsif header :contains "From" "sylvie@linuxsilo.net" {
    fileinto "Inbox.Sylvie";
}
elsif address :contains :all ["From"] "newsletter@linuxsilo.net" {
    fileinto "Inbox.Newsletters";
}
elsif header :contains "Subject" "firebird-support" {
    fileinto "Inbox.Firebird";
}
elsif header :contains "X-Mailing-List" "debian-powerpc@lists.debian.org" {
    fileinto "Inbox.Debianppc";
}
elsif header :is "X-Spam-Flag" "YES" {
    fileinto "Inbox.Junk";
}
else {
    fileinto "Inbox";
}
```

8. Filtros de contenidos.

SpamAssassin y ClamAV son dos filtros de contenidos que serán utilizados desde Postfix a través de Amavisd-new. Se instalarán y configurarán primero estos filtros y después el interfaz Amavisd-new entre ellos y el servidor.



8.1. SpamAssassin.

Ejecutaremos el siguiente comando como root:

```
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 tocar nada. En el fichero `/etc/default/spamassassin` podemos observar que el demonio de SpamAssassin no se ejecuta por defecto, que 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 darnos 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, y 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* (*man sa-learn* para su documentación). Con *sa-learn --spam <directorio>* lo instruimos para que recoja información de correos que sabemos con certeza que son *spam*, y con *sa-learn --ham <directorio>* lo instruimos para que recoja información de correos que sabemos con certeza que no son *spam*. 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.

Este 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 *job* 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*. De hecho, éste es el motivo por el cuál se estableció la máscara de los ficheros y subdirectorios de Cyrus a 027, de modo que, si añadimos el usuario *amavis* al grupo *mail-adduser amavis mail-*, se podrán leer esos mails considerados *ham* o *spam*. Asimismo, es posible que los directorios y ficheros que se crearon por los scripts de instalación antes de realizar el cambio en `/etc/imapd.conf` deban ver sus permisos modificados. El comando *find* nos será de gran utilidad para este propósito, por ejemplo:

```
cd /var/spool/cyrus/mail
find -type f -exec chmod g+rw '{}' ';'
find -type d -exec chmod g+rwx '{}' ';'
```

Nos actualizará los permisos de forma adecuada en una instalación exacta a la que se ha realizado en el artículo.

En cualquier caso, como administrador y por propia experiencia, no recomiendo dejar en manos de los usuarios la distinción de lo que es *spam* y lo que es *ham*.

8.2. Clam Antivirus.

Ejecutaremos los siguientes comandos como root (teniendo el repositorio *non-free* en nuestro `/etc/apt/sources.list`):

```
apt-get install unrar lha arj unzoo zip unzip bzip2 gzip cpio file lzop
apt-get install clamav clamav-base clamav-daemon clamav-freshclam libclamav1
```



Como método de actualización de la lista y los patrones de los virus del *clamav-freshclam* recomiendo seleccionar *daemon* y como mirror para la descarga el más cercano a la localización geográfica de nuestro servidor. En la pregunta *Number of freshclam updates per day*, con 12 tenemos actualizada cada dos horas nuestra lista de virus. Y contestamos que sí a la pregunta de *Should clamd be notified after updates?*. 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*:

```
genma amavis[13147]: (13147-01) Clam Antivirus-clamd FAILED - unknown status:
/var/lib/amavis/amavis-20040818T163812-13147/parts: Access denied. ERROR\n
genma 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*.

8.3. Amavisd-new.

Tan sólo es necesario instalar un paquete, como root:

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

El fichero de configuración de *Amavisd-new* es bastante largo, pero tan sólo es necesario realizar unos pocos cambios a la configuración que viene por defecto. En este artículo se propone una configuración que marca los correos que son identificados como *spam*, pero los deja llegar a su destinatario. Entonces, mediante *Sieve*, es muy fácil añadir una línea que mueve directamente los correos con cierta cabecera específica a una carpeta genérica de *spam*. Así, el usuario tan sólo tiene que revisar periódicamente los correos que hay en esa carpeta y, viendo que no hay ningún falso positivo, borrarlos. En caso de haber falsos positivos, es preciso instruir a *SpamAssassin* para que olvide (opción *forget* de *sa-learn*) ese correo como *spam*.

SpamAssassin realimenta automáticamente sus filtros bayesianos con los correos que son identificados como *spam* y que superan un cierto umbral. Pero es conveniente seguir entrenándolo con aquellos correos que son *spam* y que se le han pasado por alto, los llamados falsos negativos. Este proceso es necesario pero delicado, pues una mala instrucción puede deshacernos casi irremediablemente el trabajo hecho hasta la fecha en nuestra base de datos bayesiana.

A continuación se presentan las líneas del fichero */etc/amavis/amavisd.conf* que necesitan ser modificadas, en el formato definitivo (es decir, con las modificaciones ya realizadas):

```
$mydomain = 'linuxsilo.net';
$myhostname = 'webmail.linuxsilo.net';
# @bypass_spam_checks_acl = qw( . );
$final_spam_destiny = D_PASS;
$warnbannedsender = 1;
$warnbadhsender = 1;
# $virus_quarantine_to = 'virus-quarantine';
$virus_quarantine_to = "virus-quarantine@$mydomain";
# $sa_spam_subject_tag = '***SPAM*** '; #
$banned_filename_re = new_RE(
# qr'^UNDECIPHERABLE$',
qr'\.[^.]*(exe|vbs|pif|scr|bat|cmd|com|dll)$'i,
qr'[\{\}]',
# qr'\.(exe|vbs|pif|scr|bat|cmd|com)$'i,
qr'\.(ade|adp|bas|bat|chm|cmd|com|cpl|crt|exe|hlp|hta|inf|ins|isp|js|
jse|lnk|mdb|mde|msc|msi|mso|mst|pcd|pif|reg|scr|sct|shs|shb|vb|
vbe|vbs|wsc|wsf|wsh)$'ix,
qr'\.(mim|b64|bhx|hqx|xex|uu|uue)$'i,
# qr'^\.(zip|lha|tnef|cab)$'i,
```



```
qr'^\.exe$'i,
qr'^application/x-msdownload$'i,
qr'^application/x-msdos-program$'i,
qr'^message/partial$'i, qr'^message/external-body$'i,
);
```

Estas modificaciones realizadas al fichero dejan una configuración específica para que *amavisd-new* se comporte de la manera que el autor de este artículo requiere, pero lo más habitual es que deba personalizarse para cada caso. A continuación se resumen los porqués de los cambios realizados:

- **bypass_spam_checks_acl**: comentamos esta línea para que *Amavisd-new* use *SpamAssassin* (por defecto viene deshabilitado su uso).
- **final_spam_destiny**: dejamos pasar los correos identificados como *spam*, aunque siguen siendo marcados como tales mediante cabeceras en el correo. De este modo, los destinatarios seguirán recibiendo toda su correspondencia pero podrán filtrarla fácilmente usando *Sieve* y las cabeceras que *Amavisd-new* habrá añadido al mensaje.
- **warnbannedsender**: activamos el envío de un mensaje de aviso al remitente de un mensaje que contuviera algún fichero adjunto con una de las extensiones prohibidas que más abajo se detallan.
- **warnbadhsender**: igual que el anterior para ficheros con cabeceras malformadas.
- **virus_quarantine_to**: activamos la cuarentena de los correos con virus. De este modo, cualquier correo que contenga un virus detectado será redirigido a la cuenta especificada. Así, podremos revisarlos y decidir qué hacer con ellos.
- **sa_spam_subject_tag**: al comentar esta sentencia se desactiva la modificación del asunto del mensaje, pues con las cabeceras que se han añadido es suficiente para que *Sieve* (o nuestro cliente de correo) filtre adecuadamente.
- **banned_filename_re**: rechazamos correos que contengan ficheros adjuntos con alguna de las extensiones mencionadas en esta variable (únicamente se permiten ficheros comprimidos), principalmente ejecutables y scripts.

Tras esto ya podemos reiniciar el servicio mediante el comando `/etc/init.d/amavis restart` y observar su carga en el log `/var/log/mail.log`, donde se informa de todos los módulos cargados al iniciar.

8.4. Modificaciones en Postfix.

Las modificaciones a realizar en Postfix son muy sencillas. En primera instancia, 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*. Ahora tan sólo queda reiniciar el servidor Postfix mediante el comando `/etc/init.d/postfix restart`.

8.5. Medidas anti-UCE.

Las medidas anti-UCE (del inglés, *Unsolicited Commercial Email*) son una serie de mecanismos que se habilitarán en Postfix para filtrar el uso abusivo de nuestro servidor y tratar de denegar la entrada de una buena parte del correo no solicitado en él. En primer lugar, añadimos las siguientes líneas al fichero `/etc/postfix/main.cf`:

```
smtpd_helo_required = yes
disable_vrfy_command = yes

smtpd_recipient_restrictions =
    reject_invalid_hostname,
    reject_non_fqdn_hostname,
    reject_non_fqdn_sender,
    reject_non_fqdn_recipient,
    reject_unknown_sender_domain,
    reject_unknown_recipient_domain,
    permit_mynetworks,
    permit_sasl_authenticated,
    reject_unauth_destination,
    check_recipient_access pcre:/etc/postfix/recipient_checks.pcre,
    check_helo_access hash:/etc/postfix/helo_checks,
    # check_helo_access pcre:/etc/postfix/helo_checks.pcre,
    check_sender_access hash:/etc/postfix/sender_checks,
    check_client_access hash:/etc/postfix/client_checks,
    check_client_access pcre:/etc/postfix/client_checks.pcre,
    reject_rbl_client relays.ordb.org,
    # reject_rbl_client opm.blitzed.org,
    # reject_rbl_client list.dsbl.org,
    # reject_rbl_client sbl.spamhaus.org,
    # reject_rbl_client cbl.abuseat.org,
    # reject_rbl_client dul.dnsbl.sorbs.net,
    permit

smtpd_data_restrictions =
    reject_unauth_pipelining,
    permit
```

De este modo, a falta de especificar los ficheros referenciados, establecemos el siguiente flujo en las restricciones aplicadas al *smtp_recipient_restrictions* (nótese que el orden es relevante):

1. Las primeras sentencias nos aseguran que el proceso de *HELO/EHLO* y el envoltorio del mensaje son correctos. Y en la última deshabilitamos las verificaciones de direcciones de correo.
2. Deshabilitamos la concatenación (del inglés, *pipelining*) de comandos (generalmente sólo los *spammers* tratan de concatenarlos, particularmente durante ataques de diccionario).
3. Permitimos cualquier cosa que pase las restricciones de más arriba y que pertenezca a *mynetworks* (el destino no importa).
4. Permitimos a los clientes que se han autenticado por SASL.
5. Rechazamos clientes sin autenticar.
6. Comprobamos ciertas direcciones de destinatarios antes de aplicar cualquier lista negra local o de DNS.
7. Comprobamos las listas negras locales, las listas blancas locales y las listas negras y blancas combinadas (comprobación de HELO/EHLO, remitente (origen en el envoltorio) y cliente (servidor que envía)).
8. Comprobamos las listas negras de DNS y de hosts que permiten relay abiertamente.

De los seis servidores de listas negras de DNS y RHS cinco aparecen comentados. Esto es debido a que, particularmente, me es suficiente el primero, que además lista únicamente servidores que hacen relay abierto (técnicamente es una respuesta de sí o no con criterios totalmente objetivos). Queda a merced del lector aplicar más o menos listas negras, pero, en cualquier caso, es



muy recomendable pasarse primero por sus respectivas webs y tener bien claro qué criterios siguen para considerar a un servidor de correo como digno de aparecer en sus listas negras.

Mediante la directiva *check_recipient_access* aplicamos sobre el destinatario del mensaje las comprobaciones detalladas en el fichero */etc/postfix/recipient_checks.pcre*, que se resumen en revisar la sintaxis de las direcciones:

```
# This file requires PCRE support built into Postfix
/^\@/      550 Invalid address format.
/[!%\@].*\@/ 550 This server disallows weird address syntax.
/^postmaster\@/ OK
/^hostmaster\@/ OK
/^abuse\@/  OK
```

Nótese que para poder referenciar ficheros con expresiones regulares, por ejemplo en el */etc/postfix/main.cf* la línea *pcre:/etc/postfix/recipient_checks.pcre*, es necesario tener instalado el paquete *postfix-pcre*. Por otra parte, las referencias del tipo *dbm:/etc/postfix/helo_checks* requieren de la ejecución del comando *postmap <fichero>* para que se cree el fichero en formato Berkeley Database con extensión *.db*.

En la directiva *check_helo_access* se lleva a cabo una comprobación muy sencilla pero muy eficiente a la hora de evitar el *spam*. El contenido de */etc/postfix/helo_checks* es el que sigue:

```
# This file must be "compiled" with "postmap"
linuxsilo.net      REJECT You are not in linuxsilo.net
genma.linuxsilo.net REJECT You are not genma.linuxsilo.net
webmail.linuxsilo.net REJECT You are not webmail.linuxsilo.net
66.79.182.201     REJECT You are not 66.79.182.201
localhost          REJECT You are not me
```

Aunque parezcan algo triviales, estas comprobaciones son enormemente efectivas y sencillas. Simplemente se verifica que el comando *HELO/EHLO* enviado por el host que se conecta a nuestro servidor no use ni nuestro dominio, ni nuestra IP pública o privada, ni *localhost*. Otra comprobación que podría hacerse sobre el comando *HELO/EHLO* sería la de la sintaxis del host, que debe cumplir el estándar del RFC. Esta comprobación aparece comentada en el *main.cf* de más arriba, pues no la uso en la actualidad. En cualquier caso, éste sería el contenido del fichero referenciado:

```
# This file requires PCRE support built into Postfix
/^[0-9]+(\.[0-9]+){3}$/ REJECT Invalid hostname
```

Con la directiva *check_sender_access* podemos realizar comprobaciones sobre el remitente del mensaje. En mi caso particular, este fichero únicamente contiene el comentario inicial, pues no le doy uso alguno, pero puede que el lector quiera darle algún tipo de uso parecido al que se propone a continuación para */etc/postfix/sender_checks*:

```
# This file must be "compiled" with "postmap"
spammers.com      554 Spam not tolerated here
someuser@morespammers.com OK
morespammers.com  REJECT
```

Según este contenido, rechazaríamos todo el correo de los dominios *spammers.com* y *morespammers.com* excepto el del usuario *someuser@morespammers.com*.

Mediante la directiva *check_client_access* podemos realizar comprobaciones sobre el cliente que envía el mensaje. En el *main.cf* propuesto más arriba aparecen dos referencias a esta directiva, una usando una tabla de hashing y otra expresiones regulares. El autor del artículo tampoco las usa en estos momentos, y los ficheros no contienen más que la línea de comentario inicial pero, de todas maneras, a continuación se proponen posibles usos. Este sería un ejemplo de contenido para el fichero */etc/postfix/client_checks*:

```
# This file must be "compiled" with "postmap"
spammers.com      554 Spam not tolerated here
10                554 Go away!
myfriendsdomain.com OK
```



172.16

OK

De este modo, pese a que el dominio *myfriendsdomain.com* y la IP *172.11.0.0/16* pertenezcan a listas negras, nosotros decidimos aceptar los accesos que venga de ellos. Asimismo, rechazamos todas las conexiones que provengan del dominio *spammers.com* y del rango de IPs *10.0.0.0/8*. Con expresiones regulares podemos conseguir resultados más complejos, como los del fichero */etc/postfix/client_checks.pcre*:

```
# This file requires PCRE support built into Postfix
/10\.9\.8\.7/          OK
/10\.9\.([89]|10)\.\d+/ 554 Go away. We don't want any!
```

Esta expresión regular rechazaría las conexiones desde el rango 10.9.8.0 – 10.9.10.255 excepto la dirección 10.9.8.7.

Tras ejecutar los comandos *postmap* necesarios sobre los ficheros *hash* tan sólo nos queda reiniciar Postfix con el comando */etc/init.d/postfix restart*.

Algunas notas acerca de las restricciones y las listas de acceso sobre "hostname", "helo", "client", "sender" y "recipient"

HELO/EHLO es lo que la máquina que envía le dice que es a nuestra máquina. Puede disfrazarse fácilmente y frecuentemente se configura de manera incorrecta. *HELO/EHLO* se comprueba a través de las restricciones en el *helo* y el *hostname* del *smtpd*.

Sender es el envoltorio de la dirección remitente (*Mail From* en la comunicación *SMTP*), no la dirección IP de la máquina cliente ni su *hostname*, ni tampoco el campo *From:* de las cabeceras (aunque el envoltorio del remitente perfectamente puede ser el mismo que el *From:* de las cabeceras). *Sender* se comprueba mediante las restricciones del remitente en el *smtpd*.

Client es la dirección IP de la máquina que realiza el envío, y posiblemente el *hostname* (si puede obtenerse alguno de la resolución inversa de la dirección IP). *Client* se comprueba con las restricciones del cliente en el *smtpd*.

Recipient hace referencia a la dirección de correo pasada en el comando *RCPT TO* durante la comunicación *SMTP*, no el *To:* ni el *Ccc:* u otros campos de las cabeceras. *Recipient* se comprueba mediante las restricciones del destinatario en el *smtpd*.

Si se sitúan las listas de acceso antes de las comprobaciones de listas negras de DNS, tal y como se muestra en la configuración del *main.cf* más arriba, pueden servir tanto como listas negras que como listas blancas. Pero es muy importante ser cauteloso a la hora de usar las listas blancas pues, por ejemplo, si se le da el *OK* a algo en la directiva *smtpd_recipient_restrictions* antes del *reject_unauth_destination*, podríamos dejar el servidor haciendo relay abierto para todo aquel que tenga el *OK*, o a los que sean falsificables.

9. Listas de correo con Mailman.

Antes de instalar Mailman es muy recomendable fijarse en las dependencias del paquete. En el momento de escribir este artículo, el paquete *mailman* depende de los paquetes *apache apache-common apache-utils libkeynote0 pwgen*. Asimismo, es conveniente reflexionar sobre si se desea acceder a su gestión web a través de *http* o de *https* (será necesaria la instalación del soporte SSL para Apache) y si se desea instalar la versión 1 o 2 de Apache. En este artículo se dará por supuesto que el lector tiene instalado y configurado Apache. Por otra parte, antes de ejecutar el comando también conviene asegurarse de que tenemos un alias para *root* en el */etc/aliases* apuntando a un usuario que tenga buzón de correo (p.e. *jsabater*).

Tras la instalación del paquete, es conveniente leer la documentación disponible en */usr/share/doc/mailman*, principalmente en */usr/share/doc/mailman/README.Debian.gz* y en */usr/share/doc/mailman/README.POSTFIX.gz*. En ella se detallan las modificaciones necesarias en Apache para el correcto funcionamiento de la interfaz web, así como la forma de integrar Postfix y Mailman. Acto seguido se va a configurar mailman, integrándolo con Postfix, para el dominio local.

Primero procederemos a instalar el paquete Debian de Mailman mediante el comando:

```
apt-get install mailman
```

En el diálogo de configuración que nos presenta el script de postinstalación elegimos los idiomas que vamos a utilizar en Mailman y, de entre ellos, el que será elegido por defecto. A continuación debemos modificar ligeramente el fichero */etc/mailman/mm_cfg.py* a fin de que Mailman sepa que se está trabajando con Postfix como *Mail Transport Agent*:



```
MTA = 'Postfix'
```

La configuración por defecto de Postfix nos deja la directiva *alias_maps* apuntando a */etc/aliases*. Ya que no nos interesa estar modificando este fichero y ejecutando el comando *newaliases* de Postfix cada vez que creamos o borremos una lista, utilizaremos el fichero de alias propio de Mailman, que es automáticamente actualizado por los comandos *newlist* y *rmlist*. El primer paso será generarlo:

```
# cd /var/lib/mailman
# bin/genaliases
```

A continuación añadiremos ese fichero de alias a la directiva *alias_maps* del */etc/postfix/main.cf*, además de otras directivas necesarias, tal que:

```
alias_maps = hash:/etc/aliases, hash:/var/lib/mailman/data/aliases
mailman_destination_recipient_limit = 1
unknown_local_recipient_reject_code = 550
owner_request_special = no
recipient_delimiter = +
```

Y solicitaremos a Postfix que recargue la configuración:

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

El tercer paso de la instalación de Mailman nos avisa de que es necesario crear una *site list* llamada *mailman* y que hasta que no la creamos el demonio del Mailman no arrancará. Ahora es el momento de crearla y, para ello, ejecutamos el siguiente comando:

```
$ newlist mailman
Enter the email of the person running the list: jsabater@linuxsilo.net
Initial mailman password:
Hit enter to notify mailman owner...
```

Nótese que Mailman no nos muestra la lista de alias que nos requiere que añadamos a nuestro fichero de alias. Esto es debido a la configuración realizada más arriba, eliminándose de esta manera este tedioso paso. Podemos, por lo tanto, pulsar *enter* y pasar a iniciar el demonio de Mailman mediante el comando */etc/init.d/mailman start*. Una vez iniciado el servicio, recibiremos el correo que nos notifica la creación de la lista en la dirección de correo que hayamos especificado (*jsabater@linuxsilo.net* en este ejemplo).

Las listas que creamos en el futuro tampoco nos solicitarán que añadamos manualmente la lista de alias. Cuando se añada o quite una lista, el fichero */var/lib/mailman/data/aliases.db* será automáticamente actualizado, pero no se ejecutará automáticamente un */etc/init.d/postfix reload*. Esto es debido a que es necesario ser *root* para ejecutar este comando y los scripts *suid-root* no son seguros. El único efecto de esto es que le llevará aproximadamente un minuto a Postfix darse cuenta de los cambios y actualizar sus tablas, si bien considero esto una inconveniencia menor.

Finalmente, para poder acceder a la interfaz web de Mailman, deberemos llevar a cabo unas simples modificaciones en Apache, versión 2.0.50 en este artículo:

1. **Activar el módulo *cgi***, por ejemplo mediante el comando *a2enmod cgi*. Este paso es obligatorio.
2. **Añadir algunos alias que nos permitan acceder mediante URLs más cortas.** En el *default virtual host* o en el del dominio que queramos usar para acceder a la interfaz, por ejemplo *lists.linuxsilo.net*, añadiremos las siguientes directivas:

```
ScriptAlias /mailman/ /usr/lib/cgi-bin/mailman/
Alias /pipermail/ /var/lib/mailman/archives/public/
Alias /images/mailman/ /usr/share/images/mailman/
```

Este paso es opcional. Una manera más sencilla de realizar estas modificaciones y que consigue que afecten a todos los hosts es crear un fichero llamado *mailman* dentro del directorio */etc/apache2/conf.d/* que contenga esas tres líneas.



Tras reiniciar Apache 2 con `apache2ctl graceful` podemos acceder a la interfaz web de Mailman a través de la URL `http://webmail.linuxsilo.net/mailman/listinfo` (o con el dominio que hayamos definido).

10. Correo a través de web con SquirrelMail.

De nuevo, en Debian GNU/Linux la instalación del software es tan sencilla como ejecutar un simple comando como root:

```
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 personalmente he considerado que eran merecedoras de modificación:

- **Organization Preferences: Organization Name:** Linuxsilo.net
- **General Options: Default Charset:** iso-8859-15
- **Message of the Day (MOTD): Edit the MOTD:** Welcome to Linuxsilo.net. Unauthorized access is forbidden.

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 débil 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:

```
# users will prefer a simple URL like http://webmail.example.com
<VirtualHost *:80>
DocumentRoot /usr/share/squirrelmail
ServerName webmail.linuxsilo.net
</VirtualHost>
```

Por supuesto necesitaremos una entrada en nuestro servidor de DNS que dirija los accesos a `webmail.linuxsilo.net` a la dirección IP de nuestra máquina. Si no deseamos tener un `virtualhost webmail.example.com` específico, dejamos comentado el segundo párrafo y accedemos a través de la URL `http://linuxsilo.net/squirrelmail`. Tras estos cambios tan sólo nos queda reiniciar el servidor web con `apache2ctl graceful` y cargar la URL definida en nuestro navegador preferido. En el caso de que no queramos definir un `virtual host`, también podemos acceder a través de la URL `http://linuxsilo.net/squirrelmail`, pues para ello se define el alias al principio del fichero `/etc/squirrelmail/apache.conf`.

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 de este artículo, nuestro servidor IMAP sólo acepta conexiones sobre la interfaz local, por lo cual no deberemos modificar el fichero `/etc/cyrus.conf`. Por si ha saltado el punto 6 de configuración TLS, acto seguido se muestra la línea que debe modificarse en el fichero:

```
imap          cmd="imapd -U 30" listen="localhost:imap" prefork=0 maxchild=100
```

Recuerde el lector que debe reiniciar el servidor Cyrus tras el cambio: `/etc/init.d/cyrus21 restart`. Si se desea tener acceso por `https` será necesario instalar el módulo `ssl` en Apache 2 con el comando `a2enmod ssl`, añadir las directivas necesarias en el `default virtualhost`, o en el que se tenga definido, y luego reiniciar el servidor con `apache2ctl graceful`.

Para comprobar la correcta instalación de Squirrelmail podemos usar el script `configtest.php`, localizable en `http://webmail.linuxsilo.net/src/configtest.php`



Finalmente, si queremos instalar el plug-in [Avelsieve – SIEVE Mail Filters](#)⁽³⁹⁾, deberán seguirse los siguientes pasos:

1. Descargar la última versión estable de la [página de descargas de Avelsieve](#)⁽⁵⁴⁾.
2. Descomprimir el fichero en el subdirectorio de plug-ins de Squirrelmail, `/usr/share/squirrelmail/plugins/` y asegurarnos de que el propietario y el grupo de los ficheros es `root` (`chown -R root.root /usr/share/squirrelmail/plugins/avelsieve`).
3. Activar el plug-in usando el script de configuración de Squirrelmail, que se encuentra en `/etc/squirrelmail/conf.pl`. Las opciones de menú a seguir son *Plugins: avelsieve*, guardando luego los cambios antes de salir.
4. Copiar el fichero `/usr/share/squirrelmail/plugins/avelsieve/config_sample.php` a `/usr/share/squirrelmail/plugins/avelsieve/config.php`, cambiando luego las opciones por defecto de ese fichero que no satisfagan nuestras necesidades.
5. Aparecerá en el interfaz web de Squirrelmail un nuevo enlace llamado *Filters*, desde donde cada usuario puede configurarse sus propios scripts.

Como nota al usuario, Avelsieve crea un script llamado *phpscript* dentro del subdirectorio propio del usuario en `/var/spool/sieve/` y lo activa para que sea usado por defecto.

11. Puertos en un firewall.

La configuración de un cortafuegos no implica dificultad alguna. A modo de resumen, acto seguido se listan todos los puertos que es necesario tener abiertos a las interfaces externas (se da por supuesto que la interfaz *lo* no tiene restricción alguna):

- **smtp:** 25 TCP
- **http:** 80 TCP
- **https:** 443 TCP
- **imaps:** 993 TCP

12. Uso de sockets TCP con LMTP

En el caso de que el servidor SMTP Postfix y el servidor IMAP Cyrus deban residir en máquinas diferentes, será necesario usar sockets TCP en lugar de sockets UNIX para conectarlos. Los cambios a realizar respecto de la instalación explicada durante el artículos son los siguientes:

1. En el fichero `/etc/cyrus.conf`, comentar la línea de *lmtpunix* y descomentar la de *lmt*, es decir, dejarlo como viene por defecto con la instalación del paquete.
2. En el fichero `/etc/services`, asegurarnos de que tenemos una entrada para el servicio *lmt*. El puerto usado habitualmente es el 24.
3. Asegurarnos de que tenemos permisos para *tcpwrapper* en los ficheros `/etc/hosts.allow` y `/etc/hosts.deny` (en el caso de que hayamos modificado la configuración por defecto de Debian). Cyrus podría rechazar las conexiones si no están establecidos adecuadamente).
4. Configurar el transporte *lmt* en Postfix añadiendo las siguientes líneas al fichero `/etc/postfix/main.cf`:


```
lmt_sasl_auth_enable = yes
lmt_sasl_password_maps = hash:/etc/postfix/sasl_passwd
lmt_sasl_security_options =
    lmt_destination_concurrency_limit = 100,
    lmt_destination_recipient_limit = 0
mailbox_transport = lmt:[localhost]
```
5. Especificarle a Postfix el usuario con permisos de administrador LMTP que tendrá que usar para autenticar las conexiones. Esto se hace en el fichero `/etc/postfix/sasl_passwd`:


```
# echo "localhost postman:passwd" > /etc/postfix/sasl_passwd
# postmap sasl_passwd
```
6. Modificar el fichero `/etc/imapd.conf` y asegurarse de que existe la línea *lmt_admins: postman*.

Tras estos cambios será necesario reiniciar los servidores Postfix y Cyrus. Además, allí donde aparece *localhost* deberá especificarse el hostname donde se encuentre Cyrus IMAP. Las restricciones del cortafuegos deberán variar de acuerdo a estos cambios. Más información en `/usr/share/doc/cyrus21-doc/README.postfix.gz`.



13. Request for Comments (RFCs).

Los RFC que definen el sistema de DNS están disponibles en diversos sitios web, como [RFC Editor](#)⁽⁵⁵⁾ o [Internet RFC/STD/FYI/BCP Archives](#)⁽⁵⁶⁾. Las ideas iniciales y en desarrollo aparecen primero como borradores y son más tarde formalizadas como RFC. A continuación se listan algunos de los más relevantes que hacen referencia a los formatos y protocolos usados en este artículo:

- [821](#)⁽⁵⁷⁾: Simple Mail Transfer Protocol
- [1321](#)⁽⁵⁸⁾: The MD5 Message–Digest Algorithm
- [1651](#)⁽⁵⁹⁾: SMTP Service Extensions
- [1652](#)⁽⁶⁰⁾: SMTP Service Extension for 8bit–MIMEtransport
- [1870](#)⁽⁶¹⁾: SMTP Service Extension for Message Size Declaration
- [1939](#)⁽⁶²⁾: Post Office Protocol – Version 3
- [2033](#)⁽⁶³⁾: Local Mail Transfer Protocol
- [2104](#)⁽⁶⁴⁾: HMAC: Keyed–Hashing for Message Authentication
- [2195](#)⁽⁶⁵⁾: IMAP/POP AUTHorize Extension for Simple Challenge/Response
- [2222](#)⁽⁶⁶⁾: Simple Authentication and Security Layer (SASL)
- [2243](#)⁽⁶⁷⁾: OTP Extended Responses
- [2245](#)⁽⁶⁸⁾: Anonymous SASL Mechanism
- [2289](#)⁽⁶⁹⁾: A One–Time Password System
- [2444](#)⁽⁷⁰⁾: The One–Time–Password SASL Mechanism
- [2487](#)⁽⁷¹⁾: SMTP Service Extension for Secure SMTP over TLS
- [2554](#)⁽⁷²⁾: SMTP Service Extension for Authentication
- [2595](#)⁽⁷³⁾: Using TLS with IMAP, POP3 and ACAP
- [2821](#)⁽⁷²⁾: Simple Mail Transfer Protocol
- [2831](#)⁽⁷⁴⁾: Using Digest Authentication as a SASL Mechanism
- [2920](#)⁽⁷⁵⁾: SMTP Service Extension for Command Pipelining
- [2945](#)⁽⁷⁶⁾: The SRP Authentication and Key Exchange System
- [3028](#)⁽⁷⁷⁾: Sieve: A Mail Filtering Language
- [3174](#)⁽⁷⁸⁾: US Secure Hash Algorithm 1 (SHA1)
- [3546](#)⁽⁷⁹⁾: Transport Layer Security (TLS) Extensions

14. Agradecimientos.

Quisiera agradecer a las siguientes personas su colaboración en diversos aspectos que me han llevado a poder ofrecer este artículo a la comunidad de software libre:

- [Henrique de Moraes Holschuh](#)⁽⁸⁰⁾, mantenedor de los paquetes Debian de Cyrus, por su ayuda en la comprensión de Cyrus.
- [Kiko Piris](#)⁽⁸¹⁾, de [Bulma](#)⁽⁸²⁾, por la revisión del artículo.
- [Jesús Roncero](#)⁽⁸³⁾, de [Bulma](#)⁽⁸²⁾, por su artículo sobre Postfix con LDAP.
- [Magnus Bäck](#)⁽⁸⁴⁾, de la lista [postfix–users](#)⁽⁸⁵⁾, por su ayuda técnica.
- [Ralf Hildebrandt](#)⁽⁸⁶⁾, de la lista [postfix–users](#)⁽⁸⁵⁾, por su ayuda técnica.
- [Wietse Venema](#)⁽⁸⁷⁾, de la lista [postfix–users](#)⁽⁸⁵⁾, por su ayuda y por haber creado y seguir liderando Postfix.
- [Arнау Bria Ramírez](#)⁽⁸⁸⁾, de [Bulma](#)⁽⁸²⁾, por los errores encontrados y su ayuda para corregirlos.
- [Jorge Vas Moreno](#)⁽⁸⁹⁾, de la Universidad de Sevilla, por sus correcciones que aparecen en la versión 1.0.9 del artículo.

15. Bibliografía.

- [POP before SMTP HOWTO document](#)⁽⁹⁰⁾
- [Postfix SpamAssassin Procmail](#)⁽⁹¹⁾
- [Mail relay testing](#)⁽⁹²⁾
- [LinuxBeginner.org – Spam filtering with postfix, AMaViSd–new, and Spamassassin](#)⁽⁹³⁾
- [Archived Weblog Entry – 05/30/2003: "Ok, the last of the spam lessons"](#)⁽⁹⁴⁾
- [Postfix+Cyrus Imap+Sasl+tls](#)⁽⁹⁵⁾
- [Luc de Louw 's Homepage](#)⁽⁹⁶⁾



- [The web-cyradm Project pages: Web-cyradm english](#)⁽⁹⁷⁾
- [Postfix Documentation, Howtos and FAQs](#)⁽⁹⁸⁾
- [Welcome to High5!](#)⁽⁹⁹⁾
- [Postfix FAQ](#)⁽¹⁰⁰⁾
- [Mailgraph Homepage](#)⁽¹⁰¹⁾
- [Welcome to logreport.org](#)⁽¹⁰²⁾
- http://www.clearplastic.com/new_mail.html⁽¹⁰³⁾
- [\[plug\] Delivering from Postfix to Cyrus via LMTP over TCP](#)⁽¹⁰⁴⁾
- [\[plug\] Postfix SMTP AUTH with TLS on Debian GNU/Linux](#)⁽¹⁰⁵⁾
- [Installation HOWTO for Postfix with MySQL](#)⁽¹⁰⁶⁾
- [Andrew TWiki . Cyrus . EncryptedPasswords](#)⁽¹⁰⁷⁾
- [Managing IMAP: Chapter 9: Cyrus System Administration](#)⁽¹⁰⁸⁾
- [SPAMFILTER EMAIL RELAY SERVER GUIDE](#)⁽¹⁰⁹⁾
- http://www.allyn.com/new_mail.html⁽¹¹⁰⁾
- [Authentication vs. Authorization](#)⁽¹¹¹⁾
- [PHPMyWebHosting : HomePage](#)⁽¹¹²⁾
- <http://temp.roncero.org/~golan/sasl2.html>⁽¹¹³⁾
- [Postfix/TLS/SASL on Debian/Woody](#)⁽¹¹⁴⁾
- [The Book of Postfix](#)⁽¹¹⁵⁾
- [Disc Hosting](#)⁽¹¹⁶⁾
- [Postfix SMTP AUTH \(and TLS\) HOWTO](#)⁽¹¹⁷⁾
- [Adding TLS support to Postfix](#)⁽⁴⁹⁾
- [Configuring Cyrus IMAP](#)⁽¹¹⁸⁾
- [Open Relay Test](#)⁽¹¹⁹⁾
- [Postfix Anti-UCE Cheat Sheet](#)⁽¹²⁰⁾
- [ISP Mailserv Solution Howto](#)⁽¹²¹⁾
- [Tutorial: ISP-style Email Service with Debian-Woody and Postfix \(1.x\)](#)⁽¹²²⁾
- [ORDB.org – the Open Relay DataBase](#)⁽¹²³⁾

16. Historial de revisiones

Fecha	Versión	Cambios
17/08/2004	1.0	Documento inicial
18/08/2004	1.0.1	Encerradas las direcciones de correo del MAIL FROM y RCPT TO de la sección 5.2 (configuración de Postfix) entre símbolos de menor que y mayor que, según el RFC. Corregida la sección 8.1 (SpamAssassin) acerca del uso del demonio de SpamAssassin. Añadida a la sección 8.2 (ClamAV) una nueva pregunta del script de instalación del paquete clamav-freshclam y solución a un posible error que aparece en los logs. Rehecha la parte de la sección 8.2 (Amavisd-new) que trata de la realimentación de los filtros bayesianos del SpamAssassin con falsos negativos y del tratamiento de los falsos positivos.
30/09/2004	1.0.2	Modificada la máscara de creación de los ficheros y subdirectorios de Cyrus a 027 para dar permisos de lectura al grupo <i>mail</i> . Añadida a la sección 8.1 (SpamAssassin) el porqué de la máscara 027 en Cyrus y el uso adecuado de <i>sa-learn</i> según este cambio.
13/10/2004	1.0.3	Añadida una aclaración sobre la obligatoriedad de la instalación del paquete <i>postfix</i> durante la instalación de los paquetes de Cyrus IMAP en el punto 4. Añadida una nota en la sección 4.4 sobre los permisos del usuario administrador <i>cyrus</i> . Añadida la instalación del paquete <i>openssl</i> en el punto 6 como requisito. Modificada en la sección 6.3 la línea que dejaba comentado el servicio <i>imap</i> , que debe quedar sin comentar pero restringida al <i>localhost</i> para que se pueda usar <i>cyradm</i> . Modificada también la sección 10 en consonancia. Añadida nota aclaratorio en el punto 8.2 sobre la necesidad de tener el repositorio <i>non-free</i> en nuestro <i>/etc/apt/sources.list</i> para poder instalar los paquetes <i>unrar</i> y <i>lha</i> .



14/10/2004	1.0.4	Modificada la explicación de la sección 5.2 sobre el transporte LMTP en la configuración de Postfix, pues había una falta de correspondencia entre el párrafo explicativo (que no estaba bien) y el código fuente que venía a continuación (que era correcto).
13/01/2005	1.0.5	Añadido <code>permit_sasl_authenticated</code> debajo de <code>permit_mynetworks</code> en la sección 8.5, "Medidas anti-UCE", que posibilita que los clientes que han sido autenticados por SASL puedan enviar correos fuera de las redes establecidas en <code>mynetworks</code> . Modificadas las explicaciones al respecto de estos parámetros.
14/01/2005	1.0.6	Añadidas las secciones 5.4 y 5.5, en las cuales se explican los conceptos de dominios virtuales y alias virtuales.
27/01/2005	1.0.7	Añadidos dos nuevos enlaces en la bibliografía (punto 15).
07/02/2005	1.0.8	Añadido un ejemplo de filtrado de correos marcados como spam en el punto 7 y la instalación y configuración del plug-in Avelsieve de gestión de scripts Sieve para Squirrelmail en el punto 10.
23/05/2005	1.0.9	Añadidos los datos de acceso a <code>cyradm</code> al principio del paso 4.3. Añadido el uso de la autoridad certificadora CAcert.org ⁽⁵⁰⁾ en el punto 6.1 y corregida la llamada al script <code>CA.pl</code> . Corregida la llamada al script <code>/etc/init.d/amavis</code> en el punto 8.3. Corregido un <code>chmod</code> que debía ser un <code>chown</code> en el punto 10. Cambiado el valor del parámetro <code>\$myorigina</code> en el punto 5.2, la respuesta a la pregunta <i>Mail name?</i> del asistente de configuración del paquete Postfix en el punto 5.0 y también el contenido del fichero <code>/etc/mailname</code> al dominio <code>linuxsilo.net</code> de modo que ahora Squirrelmail no envíe correos con el subdominio tras la arroba. Añadida la directiva <code>local_recipient_maps =</code> al fichero de configuración <code>/etc/postfix/main.cf</code> de Postfix. Corregidas las versiones de software según lo disponible en la rama Sarge de Debian.

Lista de enlaces de este artículo:

1. <http://www.debian.org/>
2. <http://www.debian.org/intro/free>
3. <http://www.kernel.org/>
4. <http://www.gnu.org>
5. <http://www.debian.org/distrib/packages>
6. <http://www.postfix.org/>
7. <http://www.sendmail.org/>
8. <http://bulma.net/postfix-architecture.png>
9. <http://www.postfix.org/documentation.html>
10. <http://www.netscape.com/>
11. <http://wp.netscape.com/eng/ssl3/>
12. <http://www.ietf.org/html.charters/tls-charter.html>
13. <http://www.openssl.org/>
14. <http://asg.web.cmu.edu/cyrus/imapd/>
15. <http://asg.web.cmu.edu/>
16. <http://www.cmu.edu/>
17. <http://www.sleepycat.com/>
18. <http://www.imc.org/ietf-sasl/>
19. <http://www.ietf.org/>
20. <http://asg.web.cmu.edu/sasl/sasl-library.html>
21. <http://asg.web.cmu.edu/sasl/>
22. <http://www.postfix.org/lmtp.8.html>
23. <http://www.cyrussoft.com/sieve/>
24. <http://www.ijs.si/software/amavisd/>
25. <http://www.spamassassin.org/>
26. <http://razor.sourceforge.net/>
27. <http://www.clamav.net/>
28. <http://www.linux.org/>
29. <http://www.freebsd.org/>
30. <http://www.rarlab.com/>



31. <http://www.gzip.org/>
32. <http://sources.redhat.com/bzip2/>
33. <http://www.qmail.org/qmail-manual-html/man5/mbox.html>
34. <http://www.qmail.org/qmail-manual-html/man5/maildir.html>
35. <http://www.gnu.org/software/mailman/>
36. <http://www.squirrelmail.org/>
37. <http://www.php.net/>
38. <http://www.w3.org/TR/1998/REC-html40-19980424/>
39. <http://email.uoa.gr/projects/squirrelmail/avelsieve.php>
40. http://www.squirrelmail.org/plugin_view.php?id=73
41. <http://email.uoa.gr/projects/cyrusmaster/>
42. <http://www.ietf.org/html.charters/otp-charter.html>
43. <http://srp.stanford.edu/>
44. <http://web.mit.edu/kerberos/www/>
45. <http://www.perl.com/>
46. <http://www.verisign.com/>
47. <http://www.thawte.com/>
48. http://www.aet.tu-cottbus.de/personen/jaenicke/postfix_tls/doc/myownca.html
49. http://postfix.state-of-mind.de/patrick.koetter/smtpath/postfix_tls_support.htm
50. <http://www.cacert.org/>
51. <http://www.cacert.org/index.php?id=3>
52. <http://www.cacert.org/docs/>
53. <http://www.cacert.org/cacert.crt>
54. http://email.uoa.gr/projects/squirrelmail/avelsieve_download.php
55. <http://www.rfc-editor.org>
56. <http://www.faqs.org/rfcs/>
57. <http://www.faqs.org/rfcs/rfc821.html>
58. <http://www.faqs.org/rfcs/rfc1321.html>
59. <http://www.faqs.org/rfcs/rfc1651.html>
60. <http://www.faqs.org/rfcs/rfc1652.html>
61. <http://www.faqs.org/rfcs/rfc1870.html>
62. <http://www.faqs.org/rfcs/rfc1939.html>
63. <http://www.faqs.org/rfcs/rfc2033.html>
64. <http://www.faqs.org/rfcs/rfc2104.html>
65. <http://www.faqs.org/rfcs/rfc2195.html>
66. <http://www.faqs.org/rfcs/rfc2222.html>
67. <http://www.faqs.org/rfcs/rfc2243.html>
68. <http://www.faqs.org/rfcs/rfc2245.html>
69. <http://www.faqs.org/rfcs/rfc2289.html>
70. <http://www.faqs.org/rfcs/rfc2444.html>
71. <http://www.faqs.org/rfcs/rfc2487.html>
72. <http://www.faqs.org/rfcs/rfc2554.html>
73. <http://www.faqs.org/rfcs/rfc2595.html>
74. <http://www.faqs.org/rfcs/rfc2831.html>
75. <http://www.faqs.org/rfcs/rfc2920.html>
76. <http://www.faqs.org/rfcs/rfc2945.html>
77. <http://www.faqs.org/rfcs/rfc3028.html>
78. <http://www.faqs.org/rfcs/rfc3174.html>
79. <http://www.faqs.org/rfcs/rfc3546.html>
80. <mailto:hmh@debian.org.nospam>
81. <mailto:kiko.nospam>
82. <http://bulma.net/>
83. mailto:jesus_roncero.nospam
84. <mailto:magnus@dsek.lth.se.nospam>
85. <http://www.postfix.org/lists.html>
86. <mailto:ralf.hildebrandt@charite.de.nospam>
87. <mailto:wietse@porcupine.org.nospam>
88. <mailto:arnau.bria@analisisysimulacion.com>
89. <mailto:coquevas@gmail.com>



90. <http://www.mchenry.net/popauth.html>
91. <http://traxel.com/doc/spamassassin-setup.html>
92. <http://abuse.net/relay.html>
93. <http://www.linuxbeginner.org/modules.php?name=News&file=print&sid=263>
94. <http://www.geekly.com/entries/archives/00000155.htm>
95. <http://cernalo.escomposlinux.org/~emeteo/imap/imap+postfix/>
96. <http://www.delouw.ch/>
97. <http://www.web-cyradm.org/index.php>
98. <http://www.postfix.org/docs.html>
99. <http://high5.net/>
100. <http://www.seaglass.com/postfix/faq.html>
101. <http://people.ee.ethz.ch/~dws/software/mailgraph/>
102. <http://www.logreport.org/>
103. http://www.clearplastic.com/new_mail.html
104. <http://lists.q-linux.com/pipermail/plug/2003-July/029508.html>
105. <http://lists.q-linux.com/pipermail/plug/2003-July/029503.html>
106. http://kummefryser.dk/HOWTO/mail/postfix_mysql.html#tableex
107. <http://acs-wiki.andrew.cmu.edu/twiki/bin/view/Cyrus/EncryptedPasswords>
108. <http://www.oreilly.com/catalog/mimap/chapter/ch09.html#91630>
109. <http://www.geocities.com/scottlhenderson/spamfilter.html>
110. http://www.allyn.com/new_mail.html
111. <http://www.oit.duke.edu/~rob/kerberos/authvauth.html>
112. <http://pmwhwiki.cs-ol.de/tiki-index.php>
113. <http://temp.roncero.org/~golan/sasl2.html>
114. <http://silicon-verl.de/home/flo/software/postfixsasl.html>
115. <http://www.postfix-book.com/>
116. <http://dischosting.sourceforge.net/>
117. <http://postfix.state-of-mind.de/patrick.koetter/smtppauth/index.html>
118. <http://www.delouw.ch/linux/Postfix-Cyrus-Web-cyradm-HOWTO/html/cyrus-config.html>
119. <http://members.iinet.net.au/~remmie/relay/>
120. <http://jimsun.linxnet.com/misc/postfix-anti-UCE.txt>
121. <http://www.marlow.dk/site.php/tech/postfix>
122. <http://workaround.org/articles/ispmail/>
123. <http://www.ordb.org/>

E-mail del autor: primetime_ARROBA_linuxsilo.net

Podrás encontrar este artículo e información adicional en: <http://bulma.net/body.phtml?nIdNoticia=2163>