



Filtrar spam y virus con Sendmail

ESTRATEGIAS DE FILTRADO

Una estrategia estructurada con Sendmail nos ayuda a maximizar la protección frente a spam y virus. **POR HANNES KASPARICK**

El versátil Mail Transport Agent (MTA) Sendmail nos ofrece numerosas maneras de controlar la epidemia de virus y spam. Este artículo presenta tres escenarios de trabajo antispam y antivirus antes de finalizar con un presentación de un método radical que más o menos acabaría completamente con el spam. Suponemos que estamos mínimamente familiarizados con la configuración de Sendmail. Nos vamos a concentrar en las herramientas añadidas prácticas. Ciertos parámetros de configuración internos de Sendmail [1], así como estrategias de prevención como listas negras, también ayudan a luchar contra el malware, a pesar de que estas técnicas raramente proporcionan una solución completa. Para más información acerca de cómo configurar Sendmail, véase el resumen de fuentes en la página Web de Sendmail [2].

Interfaz de Conexión

Amavisd-new [3] es una interfaz entre el MTA y un filtro de contenidos como un escáner de virus y/o Spamassassin. Podemos acudir al artículo al respecto en el número 15 de la edición en castellano de Linux Magazine [4]. Se encuentran disponibles distintas configuraciones de

Amavis para varios MTAs. Amavis es muy seguro desde un punto de vista de seguridad, ya que el programa en Perl no es susceptible de padecer desbordamiento de búfer. Generalmente no necesita permisos de root. Puede funcionar dentro de una jaula chroot, y ofrece al administrador opciones de configuración para evitar ataques por denegación de servicio basados en mail bombs.

La interfaz se instala con facilidad. Para sistemas basados en Debian, ejecutamos `apt-get install amavisd-new`. Como Amavis también hace un chequeo en busca de virus, generalmente vamos a querer instalar un antivirus. La configuración se ubica en `/etc/amavis/amavisd.conf`.

Existen varios métodos para integrar Amavis con Sendmail. El más recomendado es la variante Dual de Sendmail. En este escenario, Amavis acepta correo electrónico en el puerto 10024 de la interfaz local de la primera instancia de Sendmail y reenvía los mensajes a la segunda instancia de Sendmail tras la verificación.

Sendmail Dual

Ejecutar Amavis significa configurar dos procesos de Sendmail que administren colas de correo por separado. Uno de los procesos controla la cola de recepción

(MTA-RX), mientras que la segunda controla la parte de emisión (MTA-TX). Amavis-new se sitúa en el medio y actúa como filtro de virus y spam. El proceso MTA-RX escucha en el puerto 25 TCP, y lee su configuración de `/etc/mail/sendmail-rx.cf` y `/etc/mail/submit.cf`, junto con el archivo de fuentes `/etc/mail/hostname-rx.mc`.

Tecleamos `mkdir /var/spool/mqueue-rx` para crear el directorio de cola de correo. La siguiente línea nos proporciona los permisos requeridos:

```
chown root:amavis ⤵
/var/spool/mqueue-rx && ⤵
chmod 700 /var/spool/mqueue-rx
```

Ahora continuamos definiendo nuestro propio socket de control para que los dos procesos de Sendmail no entren en conflicto:

```
define⤵
(`confCONTROL_SOCKET_NAME', ⤵
`/var/run/sendmail/mta/⤵
smcontrol-rx')dn1
```

El proceso MTA-TX liga el puerto 10025 de la interfaz local y usa la cola de correo y el archivo de configuración normales. Para mantener legibles las cosas, vamos a usar un archivo de fuentes compartido `/etc/mail/Hosname-tx.mc`. Definimos las configuraciones para recibir email, incluyendo los límites de recursos, en `hostname-rx.mc`. El lado que envía está

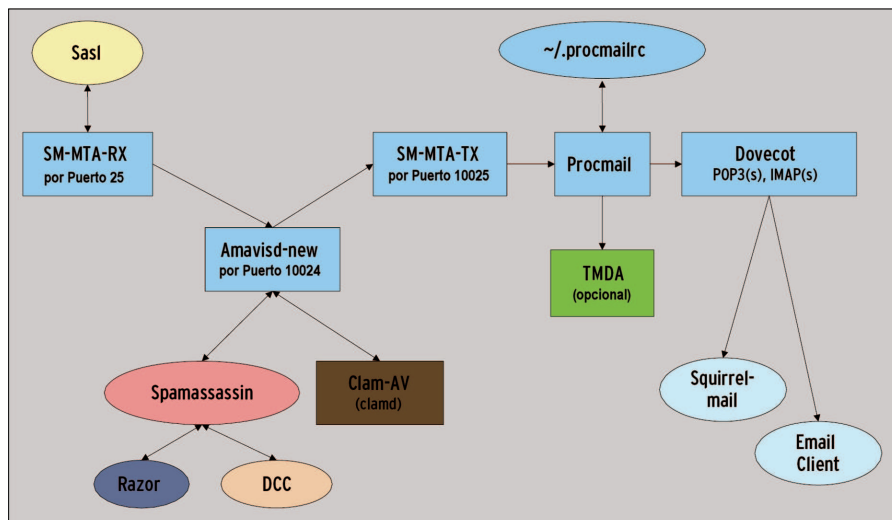


Figura 1: Sencillo, pero inflexible: Sendmail y Amavis luchando contra el spam y los virus por su cuenta.

controlado por *hostname-tx.mc*. Una ventaja de la instalación dual de Sendmail es que MTA-RX, que es accesible desde Internet, no necesita ejecutarse con permisos de root ni tampoco acceder a los directorios de los usuarios.

Solución 1: Amavis Sólo en Casa

El método más sencillo pero menos flexible para combatir el spam y los virus es usar Amavis en solitario (véase la Figura 1). La interfaz integra de manera automática Spamassassin, sus plugins y un programa antivirus. Si no somos capaces de configurar la variable *\$mydomain*, generalmente no va a funcionar nada, por lo que podemos empezar asignándole un valor. Definimos valores adecuados tanto

para *\$LOGFILE* como para *\$log_level*, para habilitar la resolución de problemas mediante el análisis de los logs.

Ahora tenemos que tener en cuenta lo que va a ocurrir con los mensajes de spam o que estén infectados con virus, y en consecuencia definimos *\$final_virus_destiny* y *\$final_spam_destiny*. Nuestras opciones son *D_PASS*, *D_DISCARD*, *D_BOUNCE* y *D_REJECT*. Un administrador responsable optará por *D_BOUNCE* antes que por *D_REJECT* para evitar problemas con el spoofing de direcciones.

Este método usa el parámetro *\$sa_** para controlar a Spamassassin. Para etiquetar cada email con una cabecera X-Spam, fijamos *\$sa_tag_level_deflt = -999*. Para permitir al usuario que identifi-

que el spam por la línea de asunto, puede ser útil un claro *\$sa_spam_subject_tag*. Para mejorar el índice de detección, puede que queramos que Amavis tenga como referencia fuentes externas (listas negras, DCC, Razor, Pyzor). Para ello, fijamos la opción *\$sa_local_tests_only = 0*.

Se ha demostrado que para *\$sa_tag2_level_deflt* son útiles los valores entre 2 y 2.5, que es la que etiqueta los mensajes bien como spam, bien como legítimos. Si tenemos demasiados falsos positivos, tendremos que subir este valor con cuidado. *\$sa_mail_body_size_limit* nos permite controlar el tamaño del correo por encima del cual Amavis no ejecuta una verificación de spam, ya que los correos grandes rara vez son spam. Con *\$sa_timeout* configuramos el número de segundos que puede esperar Amavis a Spamassassin antes de reenviar los mensajes. Las listas blancas y las listas negras, así como los receptores que prefieran tener la verificación de spam deshabilitada (*(\$spam_lovers)*) pueden configurarse con flexibilidad.

Análisis en Cascada

La solución 1 nos ofrece análisis de virus, y puede hacer uso incluso de varios programas antivirus en paralelo. Una vez que un antivirus detecte un virus, sería una pérdida de recursos mandar el correo al otro antivirus. El parámetro *\$first_infected_stops_scan = 1* evita dicha situación.

Podríamos incluso avisar al remitente de un correo infectado si especificamos el parámetro *\$warnvirussender = 1*. Sin embargo, este servicio no tiene mucho sentido, ya que los administradores no demasiado expertos van a tratar estos avisos como spam, o incluso las direcciones falsas van a impedir avisar al verdadero remitente. Si preferimos no borrar los correos infectados de manera automática, podemos configurar *\$QUARANTINEDIR*.

El escaneo antivirus puede consumir rápidamente los recursos de un servidor de correo, dando pie a ataques por denegación de servicio. Los mail bombs, en particular, se expanden hasta alcanzar un enorme tamaño cuando desempaquetamos un mail bomb para escaneralo, dejándonos sin memoria o espacio de swap. El parámetro *\$MAXLEVELS* mitiga esto al restringir los niveles de anidamiento de archivos que se han comprimido varias veces. Tras alcanzar un nivel predefinido, Amavis paraliza su

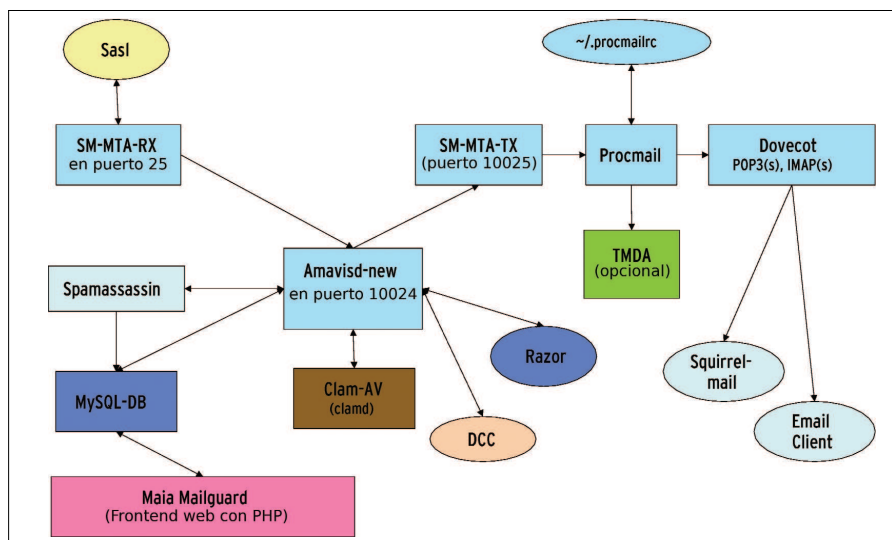


Figura 2: Un escenario más flexible, haciendo uso de bases de datos, con Sendmail, Amavis y Maia.

intento de descomprimir el archivo. `$MAXFILES` restringe el número de archivos por correo electrónico.

El parámetro `$MAX_EXPANSION*` nos permite fijar un límite de memoria en bytes que pueda ser usado en un proceso de descompresión. Cada vez que un correo electrónico exceda uno de estos tres límites, Amavisd-new versión 20030616-p8 o posterior añade una etiqueta `***UNCHECKED***` a la línea de asunto. Deberíamos filtrar de manera rutinaria los correos con esta etiqueta porque es probable que contengan virus.

Solución 2: Amavis con Maia

La solución 1 es fácil de configurar, a pesar de que no nos permite mucha configuración a nivel de usuario específico. Una manera de extender Amavis es añadir Maia Mailguard [7], como se muestra en el esquema de la Figura 2. Maia es un front-end Web sencillo, multilinguaje basado en PHP y Perl. Usa una versión parcheada de Amavisd-new, que guarda las configuraciones específicas de los usuarios en una base de datos MySQL o PostgreSQL.

El usuario puede acceder a su propia configuración individual a través del front-end Web. Y debe loguearse con su dirección de correo completa (!). Tras esto, puede administrar buzones de correo, definir listas negras individuales (véase Figura 3) o cambiar el valor umbral para el spam.

El precio por dar más servicios al usuario supone mayor riesgo para el administrador: más componentes siempre significan más riesgos de fallos. Si la base de datos se queda colgada, el servidor simplemente se queda sin distribuir correo. (Guarda los correos temporalmente, y los distribuye más tarde en caso necesario). Para mejorar la disponibili-

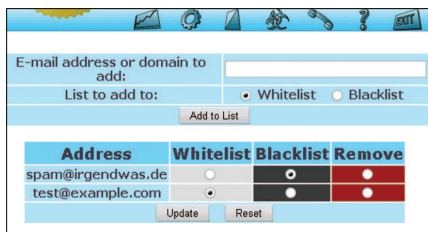


Figura 3: El front-end Maia proporciona al usuario la posibilidad de fijar configuraciones individuales, listas negras en este ejemplo.

dad, el administrador puede definir múltiples servidores SQL.

El segundo punto crítico es que el parche Maia impide que instalemos las actualizaciones de seguridad de Amavisd-new de nuestra distribución, ya que esto borraría la ruta. Para servidores de correo con un gran volumen de mail, la merma en el rendimiento debido a accesos a base de datos repetidas puede ser también muy significativa. De nuevo, el administrador puede solucionar esto asignando varios servidores.

Solución 3: Spamd basado en Milster

Para lograr una posibilidad de configuración individual máxima sin la carga gráfica, un modelo como el mostrado en la Figura 4 es seguramente nuestra mejor opción. Se usa la interfaz Milster de Sendmail para acceder a una instancia separada de Spamd (el demonio de Spamassassin). Cada usuario puede definir su propia configuración en `~/spamassassin/user_prefs`. Spamd accede al archivo `user_prefs` y detecta el spam en función de estos valores. Si no encuentra el archivo, Spamassassin aplica automáticamente las preferencias del administrador, ubicadas en `/etc/spamassassin/local.cf`.

Como Spamd necesita acceder a los directorios de usuario, se ejecuta desde la cuenta de root. Podemos configurar accesos a los directorios de usuario con permisos de nivel de usuario para que Spamd no tenga que ejecutar el proceso como root, e incrementar la seguridad de esta manera. La solución 3 es fácil de instalar: `apt-get install spamass-milter spamassassin`. Añadimos entonces las siguientes líneas:

```
INPUT_MAIL_FILTER(U
`spamassassin',
S=local:/var/run/sendmail/
spamass.sock,
F=,T=S:4m;R:4m;E:10m')dn1
```

al archivo de configuración `hostname-rx.mc` para MTA-RX, compilamos los archivos de configuración ejecutando `m4` e reiniciamos Sendmail. Definimos las preferencias globales en `/etc/spamassassin/local.cf`. Podemos verificar una lista completa de opciones en [8]. Las más importantes son las siguientes: para habilitar las configuraciones

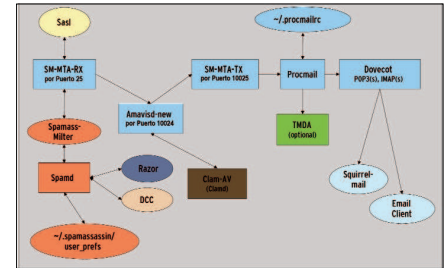


Figura 4: Una solución incluso más flexible hace uso de la interfaz Milster de Sendmail.

individuales, fijamos `allow_user_rules 1`. Para etiquetar spam, usamos `required_score 2.5`, `rewrite_subject 1` y `rewrite_header Subject ***Spam***`.

Los criterios de detección por defecto son bastante buenos. Para más detalles de una configuración más personalizada, podemos consultar [9]. Spamassassin puntúa el contenido del correo electrónico, y si la puntuación excede de un umbral, o `required_score`, etiqueta el mensaje como spam. Para incrementar la tasa de detección, es aconsejable asignar una puntuación más alta a `RAZOR2_CHECK`, `RAZOR2_CF_RANGE_51_100`, y `DCC_CHECK`. El valor `PYZOR_CHECK` es suficientemente alto.

Los Tres Cazadores: DCC, Razor y Pyzor

Los parámetros que acabamos de comentar especifican el criterio de evaluación con información de tres redes de correo globales: DCC, Razor y Pyzor, que recopilan y evalúan hashes de correo. Las funciones de hash se usan en este contexto de una manera distinta a los más conocidos MD5 y SHA1. En este caso se emplean los denominados soft hashes, que permiten que cambien algunas partes del correo, como el destinatario, sin que cambie el hash.

Cada servidor de correo que usa estas redes externas envía los hashes de todo el mail que recibe. Si 100.000 hashes alcanzan las redes (este valor es configurable) podemos suponer que este mail es spam. Después de todo, es bastante improbable que alguien envíe 100.000 correos idénticos con contenido legítimo.

Ejecutamos `apt-get` para instalar DCC, Razor y Pyzor. Spamassassin detecta automáticamente si existe. DCC requiere un servidor DCC para volúmenes superiores a los 250.000 emails y bloquea cualquier acceso por encima de

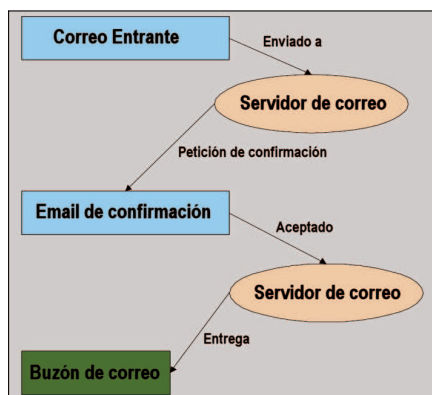


Figura 5: Un método petición-respuesta para evitar el spam gracias al Tagged Message Delivery Agent.

esa cifra. Para comprobar si estos plugins están funcionando, podemos ejecutar Tcpcdump para verificar las cabeceras de los correos spam, o bien los puertos 6277/UDP (DCC), 2703/TCP, 7/TCP (Razor) y 24441/UDP (Pyzor).

Necesitamos configurar nuestro firewall para permitir tráfico hacia afuera en estos puertos para que se establezcan las conexiones con las respectivas redes. Puede parecer un poco paranoico usar los tres programas en paralelo, pero así vamos a mejorar verdaderamente nuestra tasa de detección de spam.

Solución 4: 0% Spam

Un método menos conocido para luchar contra el spam, que sólo funciona bajo ciertas circunstancias, es el llamado Tagged Message Delivery Agent (TMDA) [10]. El TMDA adopta un método de reto-respuesta en el que cada mensaje enviado al servidor para su distribución tiene que probar su legitimidad a dicho servidor.

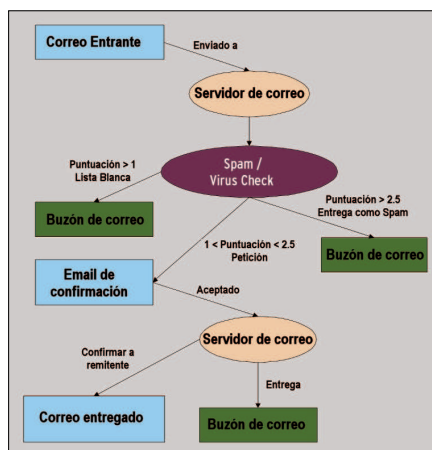


Figura 6: En un escenario TMDA más sofisticado, Spamassassin pre-selecciona los mensajes.

La Figura 5 muestra cómo funciona este principio: un servidor de correo externo está intentando enviar un mensaje a un buzón de nuestro servidor de correo. Nuestro propio servidor de correo pide al remitente que confirme y no entregue el correo original hasta que no haya recibido confirmación. Los spammers no van a enviar un mensaje de confirmación, ya que envían correos masivos, y no pueden recibir las peticiones de confirmación debido al spoofing. Esto reduce los mensajes no deseados a cero por ciento, a costa de un considerable trabajo extra. La Figura 6 muestra un método más sofisticado que TMDA.

Si optamos por este método de tolerancia cero, hemos de recordar lo siguiente:

- Los spammers normalmente usan direcciones de otra gente, por lo que nuestro servidor de correo no debería molestar a esta gente inocente pidiéndole información. En otras palabras, no deberíamos dar al TMDA los mensajes que el filtro haya identificado previamente como spam.
- A los correos electrónicos con menos nivel de spam se les puede distribuir el correo de manera directa, sin tener que pasar por el TMDA.
- Si dos sistemas TMDA se encontrasen podrían entrar en un bucle infinito. La página de TMDA [10] señala que esto no debería ocurrir, pero no se garantiza nada. El FAQ de la misma página comenta otros problemas y las posibles soluciones.
- Se recomienda configurar una lista blanca personal, independientemente de Spamassassin. El mejor método es añadir nuestra libreta de direcciones. Las listas blancas son particularmente importantes para registrarse y usar listas de correo.

Un efecto positivo es que cualquiera que nos envíe correo, y no esté en nuestra lista blanca, recibe confirmación de que el correo ha llegado. Al mismo tiempo, debería mencionarse que el TMDA no es la solución perfecta en ciertos escenarios debido a su complejidad.

Rendimiento

Un servidor de correo que intente proporcionar una solución perfecta para cualquier escenario siempre corre el peligro de afectar a su rendimiento a medida que el volumen de correo comience a incrementarse. Esto nos lleva a poner las

soluciones propuestas ante un test de rendimiento.

Hemos usado Postal [11] para hacer un test de stress a un Pentium 4 con una CPU de 2.8 GHz y 512MB de RAM. El programa de benchmark envía correos electrónicos con un tamaño máximo de 15KB al servidor en un periodo de cinco minutos. La solución 1 (Amavis en solitario) se mostró capaz de procesar 600 emails por minuto. Le llevó otros cinco minutos entregar todos los mensajes. Con ClamAV habilitado el proceso no se ralentizó notablemente. Si usamos Spamassassin como test externo, como describe la solución 3, el rendimiento depende en gran medida del ancho de banda y la latencia de la conexión a Internet que tengamos. Con TMDA (solución 4) no se ha notado un efecto significativo en el rendimiento.

Bien Está lo que Bien Acaba

El primero de los tres diseños que hemos estudiado debería funcionar bien al primer intento en la mayoría de los entornos. Antes de habilitar la solución 4, deberíamos considerar con cuidado los pros y los contras. Lo que tienen los cuatro en común es que escalan bien debido al diseño extremadamente modular. ■

RECURSOS

- [1] Funcionalidades anti-spam de Sendmail: http://www.sendmail.org/m4/anti_spam.html
- [2] Sendmail: <http://www.sendmail.org>
- [3] Amavis-new: <http://www.ijs.si/software/amavisd/>
- [4] "Filtrando", por Larkin Cunningham; Linux Magazine Edición en Castellano, Número 15, p. 25.
- [5] Securityfocus y Frsirt: <http://www.securityfocus.com> <http://www.frsirt.com>
- [6] "Antivirus", por James Mohr; Linux Magazine Edición en Castellano, Número 15, p. 16.
- [7] Maia Mailguard: <http://www.renaissance.com/maia/>
- [8] Configuración de Spamassassin para correo entrante: http://spamassassin.apache.org/full/3.0.x/dist/doc/Mail_SpamAssassin_Conf.html
- [9] Default score values for Spamassassin: http://spamassassin.apache.org/tests_3_0_x.html
- [10] TMDA: <http://www.tmda.net>
- [11] Postal: <http://www.cockler.com/postal/>