12

FILTERS

I know where to get it, if you want it.
—Jailer #1 in Monty Python's Life of Brian

As you just learned in Chapter 11, many everyday tasks that involve email are too complex for the MTA, such as scanning incoming and outgoing mail for viruses and worms, detecting spam, archiving mail, and sanitizing mail. You also saw that Postfix offers two slightly different approaches for filtering mail that differ in when they process incoming messages. This chapter addresses the practice of these two approaches.

In particular, you will see how to append disclaimers to messages by piping messages to a script and how to scan messages for viruses using either `content_filter` or `smtpd_proxy_filter` to send them off to amavisd-new.

Appending Disclaimers to Messages with a Script

Among the countless things that you can do with a `content_filter` script is add a disclaimer to all outgoing messages. The following example uses alterMIME, a small program that is used to alter mime-encoded mail, in a script to add the disclaimer. Figure 12-1 shows you how alterMIME will be integrated into the message transport process.

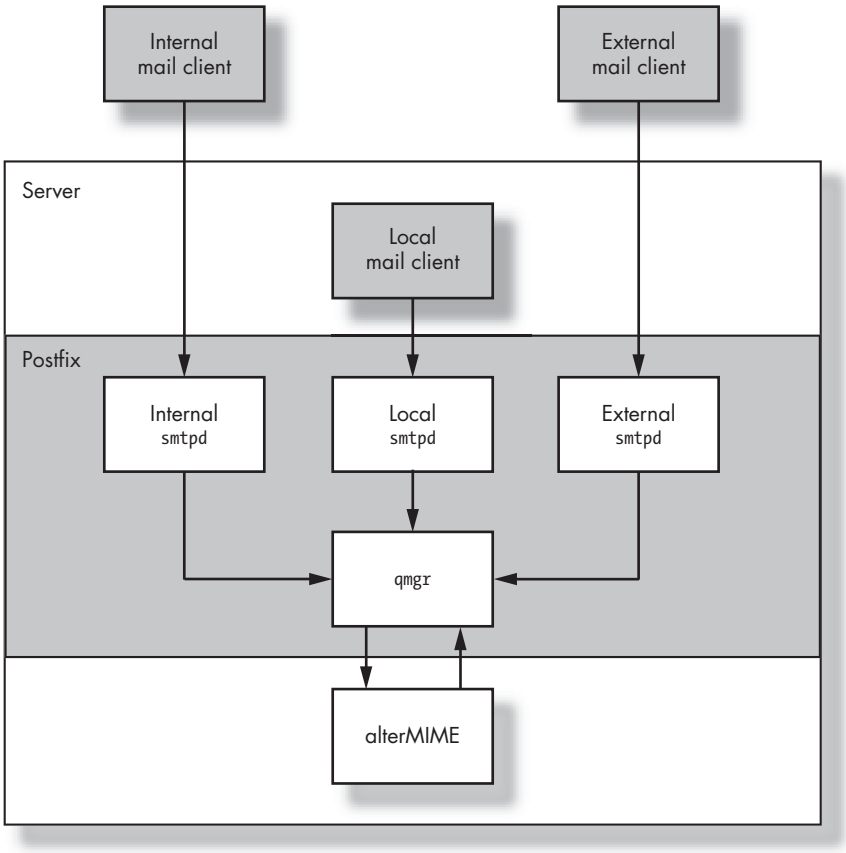

Figure 12-1: AlterMIME integration into Postfix

To add disclaimers to outbound messages without touching inbound and local messages, you need to separate the traffic for each direction. Let's say that your mail server has separate network interfaces for your internal and external networks. This means you need to create three separate instances of `smtpd` and bind them to the localhost, internal, and external network interfaces. The following example shows you how a message transport from your internal network to a remote destination would be processed if you created separate instances of `smtpd` for separate network interfaces.

1. When a message leaves your network, a mail client connects to the `smtpd` instance listening on the internal interface.
2. This internal `smtpd` accepts the message and sends it to `qmgr`.
3. `qmgr` sends the message to the `content_filter` service.
4. The `content_filter` service uses the `pipe` daemon to feed the message to the script.
5. The script adds a disclaimer.
6. The script reinjects the message to the `smtpd` instance listening on the local network interface.
7. The local `smtpd` sends the reinjected message to `qmgr`.
8. `qmgr` sends the message to `smtp` and out to the Internet.

Before you configure the transport, however, you must create the script that will invoke alterMIME from Postfix.

Installing alterMIME and Creating the Filter Script

The script will run alterMIME (`http://www.pldaniels.com/altermime`) to modify the outgoing message. If you don't have alterMIME (and you don't have a binary package for your operating system), download it, unpack it, change into your source directory, and run **`make`** and **`make install`**. This should leave you with an alterMIME executable in `/usr/local/bin/altermime`.

Creating the alterMIME Environment

You should run alterMIME as an unprivileged system user. For example, if you would like to use the `filter` username on your machine, you could run these commands to create the user:

```
# groupadd filter
# useradd -d /var/spool/altermime -G filter altermime
```

Creating a Script Directory

It's not a very good idea to clutter up your `/etc/postfix` directory with a bunch of scripts. Create a separate subdirectory as the superuser to store your scripts, and make the subdirectory accessible to `filter` and `root` only.

For example, the following command sequence creates a directory with the correct permissions and ownership:

```
# mkdir /etc/postfix/filter
# chown root /etc/postfix/filter
# chgrp filter /etc/postfix/filter
# chmod 770 /etc/postfix/filter
```

Creating the Script

The following script, named `/etc/postfix/filter/add_disclaimer.sh`, invokes alterMIME on an incoming message from Postfix (sent from the `pipe` daemon). The alterMIME program adds a disclaimer to the message and reinjects it back into the Postfix queue. AlterMIME requires a location to write a temporary file; it cannot operate on `stdin`.

```
#!/bin/sh
# System dependent settings
ALTERMIME=/usr/local/bin/altermime
ALTERMIME_DIR=/var/spool/altermime
SENDMAIL=/usr/sbin/sendmail
# Exit codes of commands invoked by Postfix are expected
# to follow the conventions defined in <sysexits.h>.
TEMPFAIL=75
UNAVAILABLE=69
# Change in to alterMIME's working directory and
# notify Postfix if 'cd' fails.
cd $ALTERMIME_DIR || { echo $ALTERMIME_DIR does not exist; exit $TEMPFAIL; }
# Clean up when done or when aborting.
trap "rm -f in.$$" 0 1 2 3 15
# Write mail to a temporary file
# Notify Postfix if this fails
cat >in.$$ || { echo Cannot write to $ALTERMIME_DIR; exit $TEMPFAIL; }
# Call alterMIME, hand over the message and
# tell alterMIME what to do with it
$ALTERMIME  --input=in.$$ \
            --disclaimer=/etc/postfix/disclaimer.txt \
            --disclaimer-html=/etc/postfix/disclaimer.txt \
            --xheader="X-Copyrighted-Material: Please visit http:// \
            www.example.com/message_disclaimer.html" || \
            { echo Message content rejected; exit $UNAVAILABLE; }
# Call sendmail to reinject the message into Postfix
$SENDMAIL "$@" <in.$$
# Use sendmail's EXIT STATUS to tell Postfix
# how things went.
exit $?
```

After creating the script, give write access only to `root`, but give execute permission to the `filter` user:

```
# chown root add_disclaimer.sh
# chgrp filter add_disclaimer.sh
# chmod 750 add_disclaimer.sh
```

Of course, now you need to create the disclaimer referenced in the script.

Creating the Disclaimer

If you already have a disclaimer, put the text in `/etc/postfix/filter/disclaimer.txt`. If you're still looking for the right disclaimer, you may want to visit emaildisclaimers.com (`http://www.emaildisclaimers.com`), a site dedicated to disclaimers and related email law. This example just uses the following dummy text (from `http://www.lipsum.com`):

```
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nam commodo
lobortis magna. Quisque neque. Etiam aliquam. Nulla tempor vestibulum.
```

With the text in place, permit only the `filter` group to read your disclaimer:

```
# chgrp filter disclaimer.txt
# chmod 640 disclaimer.txt
```

This wraps up the filter script. Now you have to configure Postfix to use the script.

Configuring Postfix for the Disclaimer Script

Configuring Postfix to invoke the script is a two-step process:

1. Define a `content_filter` parameter for the proper `smtpd` in the `master.cf` file.
2. Define the transport in the `master.cf` file.

Defining the content_filter Parameter

As explained in Chapter 11, you would now add the `content_filter` parameter to `main.cf` and specify a transport name. However, this would globally specify a `content_filter`, and the filter would apply to all processes that handle incoming mail. You don't want that to happen in this particular example, though, because you want to apply the filter only to messages that come from the internal network interface.

To assign the filter to messages coming from the internal network interface only, you will add the `content_filter` only to the single `smtpd` instance in the `master.cf` file. The address of the internal interface in the following example of `master.cf` is `172.16.0.1`:

```
127.0.0.1:smtp      inet    n       -       n       -       -       smtpd
172.16.0.1:smtp     inet    n       -       n       -       -       smtpd
    -o content_filter=disclaimer:
192.0.34.166:smtp   inet    n       -       n       -       -       smtpd
```

Notice that the name of the filter transport is `disclaimer`; this is not the script name. You'll define this transport in the next section.

Defining the Transport

You now need to define the `disclaimer` transport in the `master.cf` file. Create an instance of the `pipe` transport that runs the `add_disclaimer.sh` script. Here's how you would do it with the script shown earlier in this chapter:

```
disclaimer  unix  -       n       n       -       -       pipe
   flags=Rq user=filter argv=/etc/postfix/filter/add_disclaimer.sh -f ${sender} -- ${recipient}
```

This definition runs the `pipe` daemon as the `filter` user, calling `add_disclaimer.sh` when fed a message. It also passes the envelope sender and envelope recipient to the script. The `R` flag prepends a `Return-Path` message header with the envelope sender address, and the `q` flag quotes whitespace and other special characters in the command-line `$sender` and `$recipient` arguments.

NOTE *The pipe(8) manual page contains a full list of flags and options.*

Testing the Filter

To test the filter, you will need to perform the following steps, which are discussed in the following sections:

1. Send mail to a remote user through the internal network interface.
2. Check the mail log for filter actions.
3. Check the sent message for a disclaimer.

Sending Mail to a Remote User

To generate a message for Postfix to send to the filter, use telnet to connect to the internal network interface (where the `smtpd` instance should use the filter). Here's an example session:

```
$ telnet 172.16.0.1 25
Trying 172.16.0.1...
Connected to 172.16.0.1.
Escape character is '^]'.
220 mail.example.com ESMTP Postfix
HELO client.example.com
250 mail.example.com
MAIL FROM: <sender@example.com>
250 Ok
RCPT TO: <recipient@remote-example.com>
250 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
FROM: Sender <sender@example.com>
TO: Recipient <recipient@remote-example.com>
```

```
Subject: Testing disclaimer
This is a test. There should be text at the bottom of this message
added by a disclaimer script.
.
250 Ok: queued as 3C4D043F2F
QUIT
221 Bye
```

Checking the Mail Log

The mail log should contain evidence of filter action, as in this example:

```
Mar 12 01:59:53 mail postfix/smtpd[30206]: connect from client.example.com[172.16.0.2]
Mar 12 02:00:21 mail postfix/smtpd[30206]: 3C4D043F2F: client=client.example.com[172.16.0.2]
Mar 12 02:01:53 mail postfix/cleanup[30209]: 3C4D043F2F:
  message-id=<20040312010021.3C4D043F2F@mail.example.com>
Mar 12 02:01:53 mail postfix/nqmgr[30193]: 3C4D043F2F: from=<sender@example.com>, size=444,
  nrcpt=1 (queue active)
Mar 12 02:01:53 mail postfix/pipe[30213]: 3C4D043F2F: to=<recipient@remote-example.com>,
  relay=disclaimer, delay=92, status=sent (mail.example.com) ❶
Mar 12 02:01:53 mail postfix/pickup[30192]: 8421143F2F: uid=100 from=<sender@example.com> ❷
Mar 12 02:01:53 mail postfix/cleanup[30209]: 8421143F2F:
  message-id=<20040312010021.3C4D043F2F@mail.example.com>
Mar 12 02:01:53 mail postfix/nqmgr[30193]: 8421143F2F: from=<sender@example.com>, size=977,
  nrcpt=1 (queue active)
Mar 12 02:01:55 mail postfix/smtpd[30206]: disconnect from client.example.com[172.16.0.2]
Mar 12 02:02:03 mail postfix/smtp[30220]: 8421143F2F: to=<recipient@remote-example.com>,
  relay=mail.remote-example.com[212.14.92.89], delay=10, status=sent (250 Ok: queued as
  56851E1C65) ❸
```

❶ The `pipe` daemon uses the `disclaimer` transport to send the message to the script.

❷ The script reinjects the message with the original envelope sender.

❸ The `smtp` daemon successfully delivers the message to the envelope recipient.

Checking the Message for a Disclaimer

As a final (and somewhat obvious) test, retrieve the message and see if it contains the X-header and the disclaimer that alterMIME is supposed to add on outgoing messages. You can see both in the following example:

```
Return-Path: <sender@example.com>
X-Original-To: recipient@remote-example.com
Delivered-To: recipient@remote-example.com
Received: from mail.example.com (mail.example.com [192.0.34.166])
  by mail.remote-example.com (Postfix) with ESMTP id 56851E1C65
  for <recipient@remote-example.com>; Fri, 12 Mar 2004 02:01:25 +0100 (CET)
```

```
Received: by mail.example.com (Postfix, from userid 100)
  id 8421143F2F; Fri, 12 Mar 2004 02:01:53 +0100 (CET)
Received: from client.example.com (client.example.com [172.16.0.2])by
  mail.example.com+(Postfix) with SMTP id 3C4D043F2Ffor
  <recipient@remote-example.com>; Fri, 12 Mar 2004+02:00:21 +0100 (CET)
From: Sender <sender@example.com>
To: Recipient <recipient@remote-example.com>
Subject: Testing disclaimer
Message-Id: <20040312010021.3C4D043F2F@mail.example.com>
Date: Fri, 12 Mar 2004 02:00:21 +0100 (CET)
X-Copyrighted-Material: Please visit http://www.example.com/
message_disclaimer.html

This is a test. There should be text at the bottom of this message
added by a disclaimer script.
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nam commodo
lobortis magna. Quisque neque. Etiam aliquam. Nulla tempor vestibulum.
```

Scanning for Viruses with content_filter and amavisd-new

This section describes an advanced use of `content_filter` described in Chapter 11—how to integrate the popular program amavisd-new into Postfix. amavisd-new links an MTA and one or more virus scanners or spam-detection programs, such as SpamAssassin. It is actively developed and is recommended by many postmasters on the Postfix mailing list.

NOTE *To get virus-scanning functionality, you need to have at least one supported virus scanner installed in addition to amavisd-new; check the documentation for a survey of the supported products.*

Figure 12-2 illustrates how Postfix and amavisd-new work together with other applications such as spam detectors and virus scanners. Here's the message flow:

1. A mail client sends a message to Postfix.
2. `smtpd` accepts the message.
3. `smtpd` sends the message to `qmgr`.
4. `qmgr` sends the message to amavisd-new.
5. amavisd-new sends the message to other applications (virus scanners in this example).
6. amavisd-new reinjects the message into the local `smtpd`.
7. The local `smtpd` sends the message to `qmgr`.
8. `qmgr` either bounces or delivers the message.

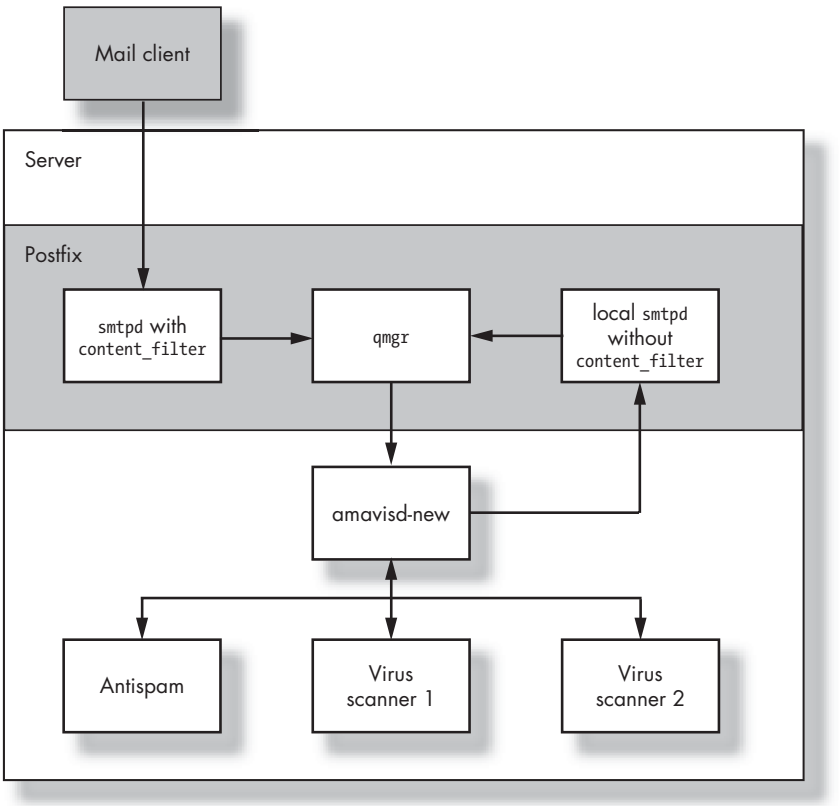

Figure 12-2: amavisd-new integration with Postfix using content_filter

Installing amavisd-new

To get amavisd-new, download it from one of the mirrors mentioned on the amavisd-new website (`http://www.ijs.si/software/amavisd`). After unpacking the archive, follow the steps in the `INSTALL` file to install amavisd-new for Postfix.

You should also read the `README.postfix` (`http://www.ijs.si/software/amavisd/README.postfix`) file for up-to-date instructions and notes specific to Postfix.

TIP *You need to build only the daemon version of amavisd-new. The helper applications, such as `amavis(.c)` and `amavisd-milter(.c)`, are not necessary for use with Postfix.*

Installing Perl Modules for amavisd-new from CPAN

amavisd-new needs a number of Perl modules to work correctly, and the `INSTALL` document in amavisd-new's `SOURCE` directory contains a full list of these modules. When installing the modules, you usually have the choice between choosing a package provided by the makers of your distribution or directly downloading the modules from CPAN (the Comprehensive Perl Archive Network, at `http://www.cpan.org`).

NOTE *CPAN is generally the best source for the most recent modules, but you may want to choose your operating system's packages for consistency instead.*

To install modules such as `Compress::Zlib` from CPAN, you need to run Perl with the CPAN module as follows:

```
# perl -MCPAN -e shell;
cpan shell -- CPAN exploration and modules installation (v1.76)
ReadLine support enabled
cpan> install Compress::Zlib
Running install for module Compress::Zlib
Running make for P/PM/PMQS/Compress-Zlib-1.33.tar.gz
Fetching with LWP:
  ftp://ftp-stud.fht-esslingen.de/pub/Mirrors/CPAN/authors/id/P/PM/PMQS/
Compress-Zlib-1.33.tar.gz
CPAN: Digest::MD5 loaded ok
Fetching with LWP:
  ftp://ftp-stud.fht-esslingen.de/pub/Mirrors/CPAN/authors/id/P/PM/PMQS/
CHECKSUMS
Checksum for /root/.cpan/sources/authors/id/P/PM/PMQS/Compress-Zlib-
1.33.tar.gz ok
Scanning cache /root/.cpan/build for sizes
Compress-Zlib-1.33/
... lots of building output ...
All tests successful, 1 test skipped.
Files=6, Tests=287,  2 wallclock secs ( 0.73 cusr +  0.11 csys =  0.84 CPU)
  /usr/bin/make test -- OK
Running make install
Installing /usr/lib/perl5/site_perl/5.6.0/i386-linux/auto/Compress/Zlib/
Zlib.so
Files found in blib/arch: installing files in blib/lib into architecture
dependent library tree
Installing /usr/lib/perl5/site_perl/5.6.0/i386-linux/Compress/Zlib.pm
Installing /usr/man/man3/Compress::Zlib.3pm
Writing /usr/lib/perl5/site_perl/5.6.0/i386-linux/auto/Compress/Zlib/.packlist
Appending installation info to /usr/lib/perl5/site_perl/5.6.0/i386-linux/
perllocal.pod
  /usr/bin/make install  -- OK
```

After getting the modules in place and installing amavisd-new, you should test it.

Testing amavisd-new

To test amavisd-new in isolation, before attempting to get it interoperating with Postfix, perform the following steps:

1. Start amavisd-new in debug mode to see if it starts up properly.
2. Perform a network test to see if it listens on a network port.

Running amavisd-new in Debug Mode

Invoking amavisd-new in debug mode gives you the answers to the following questions at once:

- Does it run? All mandatory Perl modules need to be installed for amavisd-new to start up. If a module is missing, you will get an error message that indicates the missing module.
- Can you run it as an unprivileged user? amavisd-new requires a new group (`vscan` by default) and a user account in this group (also `vscan` by default).
- Does it find optional Perl modules that implement additional functionality, such as SpamAssassin, LDAP, and SQL?
- Does it use the proper installation of Perl? If you have more than one version of Perl installed, you may not have all of the modules installed for the particular version of Perl that you're trying to use.
- Does it find auxiliary programs, such as virus scanners?
- Which configuration file is it using? Normally, it's `/etc/amavisd.conf`, but you can override this if you know exactly what you're doing.
- Can it bind to the ports specified in the configuration file?

For your first attempt, it's best to start amavisd-new interactively, keeping it attached to the terminal. To do this, switch to the user `vscan` and run amavisd-new with the `debug` option. This example session shows the output that you're looking for:

```
# su - vscan
$ /usr/local/sbin/amavisd debug
Jan 28 11:10:43 mail amavisd[29188]: starting.  amavisd at mail amavisd-new-20030616-p6
Jan 28 11:10:43 mail amavisd[29188]: Perl version              5.006 ❶
Jan 28 11:10:43 mail amavisd[29188]: Module Amavis::Conf       1.15
Jan 28 11:10:43 mail amavisd[29188]: Module Archive::Tar       1.08
Jan 28 11:10:43 mail amavisd[29188]: Module Archive::Zip       1.09
Jan 28 11:10:43 mail amavisd[29188]: Module Compress::Zlib     1.33
Jan 28 11:10:43 mail amavisd[29188]: Module Convert::TNEF      0.17
Jan 28 11:10:43 mail amavisd[29188]: Module Convert::UUlib     1.0
Jan 28 11:10:43 mail amavisd[29188]: Module MIME::Entity       5.404
Jan 28 11:10:43 mail amavisd[29188]: Module MIME::Parser       5.406
Jan 28 11:10:43 mail amavisd[29188]: Module MIME::Tools        5.411
Jan 28 11:10:43 mail amavisd[29188]: Module Mail::Header       1.60
Jan 28 11:10:43 mail amavisd[29188]: Module Mail::Internet     1.60
Jan 28 11:10:43 mail amavisd[29188]: Module Mail::SpamAssassin  2.63
Jan 28 11:10:43 mail amavisd[29188]: Module Net::Cmd           2.24
Jan 28 11:10:43 mail amavisd[29188]: Module Net::DNS           0.40
Jan 28 11:10:43 mail amavisd[29188]: Module Net::SMTP          2.26
Jan 28 11:10:43 mail amavisd[29188]: Module Net::Server        0.86
Jan 28 11:10:43 mail amavisd[29188]: Module Time::HiRes        1.55
Jan 28 11:10:43 mail amavisd[29188]: Module Unix::Syslog       0.99
```

```
Jan 28 11:10:43 mail amavisd[29188]: Found myself: /usr/sbin/amavisd -c /etc/amavisd.conf
Jan 28 11:10:43 mail amavisd[29188]: Lookup::SQL code       NOT loaded ❷
Jan 28 11:10:43 mail amavisd[29188]: Lookup::LDAP code      NOT loaded
Jan 28 11:10:43 mail amavisd[29188]: AMCL-in protocol code  loaded
Jan 28 11:10:43 mail amavisd[29188]: SMTP-in protocol code  loaded
Jan 28 11:10:43 mail amavisd[29188]: ANTI-VIRUS code        loaded
Jan 28 11:10:43 mail amavisd[29188]: ANTI-SPAM  code        loaded ❸
Jan 28 11:10:43 mail amavisd[29188]: Net::Server: 2004/01/28-11:10:43 Amavis (type \
  Net::Server::PreForkSimple) starting! pid(29188)
Jan 28 11:10:43 mail amavisd[29188]: Net::Server: Binding to UNIX socket file \
  /var/amavis/amavisd.sock using SOCK_STREAM
Jan 28 11:10:43 mail amavisd[29188]: Net::Server: Binding to TCP port 10024 on host 127.0.0.1
Jan 28 11:10:43 mail amavisd[29188]: Net::Server: Setting gid to "54322 54322"
Jan 28 11:10:43 mail amavisd[29188]: Net::Server: Setting uid to "7509"
Jan 28 11:10:43 mail amavisd[29188]: Net::Server: Setting up serialization via flock
Jan 28 11:10:43 mail amavisd[29188]: Found $file       at /usr/bin/file
Jan 28 11:10:43 mail amavisd[29188]: Found $arc        at /usr/bin/arc
Jan 28 11:10:43 mail amavisd[29188]: Found $gzip       at /usr/bin/gzip
Jan 28 11:10:43 mail amavisd[29188]: Found $bzip2      at /usr/bin/bzip2
Jan 28 11:10:43 mail amavisd[29188]: Found $lzop       at /usr/local/bin/lzop
Jan 28 11:10:43 mail amavisd[29188]: Found $lha        at /usr/bin/lha
Jan 28 11:10:43 mail amavisd[29188]: Found $unarj      at /usr/bin/unarj
Jan 28 11:10:43 mail amavisd[29188]: Found $uncompress at /usr/bin/uncompress
Jan 28 11:10:43 mail amavisd[29188]: Found $unfreeze   at /usr/local/bin/unfreeze
Jan 28 11:10:43 mail amavisd[29188]: Found $unrar      at /usr/bin/rar
Jan 28 11:10:43 mail amavisd[29188]: Found $zoo        at /usr/bin/zoo
Jan 28 11:10:43 mail amavisd[29188]: Found $cpio       at /bin/cpio ❹
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: KasperskyLab AntiViral Toolkit \
  Pro (AVP)
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: KasperskyLab AVPDaemonClient
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: H+BEDV AntiVir or \
  CentralCommand Vexira Antivirus
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: Command AntiVirus for Linux
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: Symantec CarrierScan via \
  Symantec CommandLineScanner
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: Symantec AntiVirus Scan Engine
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: Dr.Web Antivirus for \
  Linux/FreeBSD/Solaris
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: F-Secure Antivirus
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: CAI InoculateIT
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: MkS_Vir for Linux (beta)
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: MkS_Vir daemon
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: ESET Software NOD32
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: ESET Software NOD32 - \
  Client/Server Version
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: Norman Virus Control v5 / Linux
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: Panda Antivirus for Linux
```

```
Jan 28 11:10:43 mail amavisd[29188]: Found primary av scanner NAI McAfee AntiVirus (uvscan) \
  at /usr/local/bin/uvscan ❺
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: VirusBuster
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: CyberSoft VFind
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: Ikarus AntiVirus for Linux
Jan 28 11:10:43 mail amavisd[29188]: No primary av scanner: BitDefender
Jan 28 11:10:43 mail amavisd[29188]: No secondary av scanner: Clam Antivirus - clamscan
Jan 28 11:10:43 mail amavisd[29188]: No secondary av scanner: FRISK F-Prot Antivirus
Jan 28 11:10:43 mail amavisd[29188]: No secondary av scanner: Trend Micro FileScanner
Jan 28 11:10:43 mail amavisd[29188]: SpamControl: initializing Mail::SpamAssassin ❻
```

❶ This line indicates the Perl version.

❷ This line and the following one indicate that no SQL or LDAP code is present.

❸ This line indicates that the antispam code was loaded; this only succeeds when SpamAssassin or dspam are present.

❹ This line and the preceding lines indicate that various external unpackers have been found—thus enabling amavisd-new to unpack attachments compressed with these packers.

❺ This line indicates that McAfee AntiVirus is present.

❻ At this point, amavisd-new is ready.

Testing Network Connectivity

With the stand-alone amavisd-new still running, you should now see if it accepts connections. Use telnet sessions to test both of the ESMTP and LMTP alternatives.

Testing ESMTP Availability

Open a telnet connection to the local port 10024 (the default for amavisd-new). You should check that it is listening on the port and responds to ESMTP commands. amavisd-new should respond to an `EHLO` command with a set of available commands, as in the following sample session:

```
# telnet localhost 10024
220 [127.0.0.1] ESMTP amavisd-new service ready
EHLO mail.example.com
250-[127.0.0.1]
250-PIPELINING
250-SIZE
250-8BITMIME
250 ENHANCEDSTATUSCODES
QUIT
221 2.0.0 [127.0.0.1] (amavisd) closing transmission channel
```

Testing LMTP Availability

Next you need to check that amavisd-new is listening on local port 10024 (the amavisd-new default for LMTP) and responds to LMTP commands. You should be able to run an **LHLO** command, as in this session:

```
# telnet localhost 10024
220 [127.0.0.1] ESMTP amavisd-new service ready
LHLO mail.example.com
250-[127.0.0.1]
250-PIPELINING
250-SIZE
250-8BITMIME
250 ENHANCEDSTATUSCODES
QUIT
221 2.0.0 [127.0.0.1] (amavisd) closing transmission channel
```

Optimizing amavisd-new Performance

If you get a lot of mail, you might want to tweak the performance of amavisd-new. Because it makes heavy use of the filesystem to prepare messages for further inspection, amavisd-new's performance can be bound to the speed and latency of disk I/O. You can significantly optimize the read-write operation speed by moving this preparation to a RAM disk style of filesystem. The procedure described in the following sections uses the Linux temporary filesystem type (tmpfs).

Is This Safe?

You can rest assured that you won't lose any email during the filtering process due to the way that you integrate amavisd-new into Postfix. Take a look at what happens during filtering:

1. Upon receiving a new message, the Postfix queue manager daemon opens an SMTP or LMTP connection to amavisd-new.
2. amavisd-new starts working on the message (doing scanning, spam checking, and so on), but it does not immediately acknowledge that it received the message.
3. While waiting for amavisd-new, Postfix keeps the message in its queue, waiting for amavisd-new to tell it that it received the message properly.
4. After amavisd-new finishes its work, it reinjects the message back into the Postfix queue.
5. The Postfix `smtpd` that handles the reinjection accepts the message from amavisd-new.

6. Upon getting the acknowledgement from the reinjecting `smtpd`, amavisd-new acknowledges successful transport back to the originating Postfix queue manager daemon.

As you can see, amavisd-new only tells the prefilter Postfix component that it got the message after the post-filter `smtpd` accepts the processed message. This way, you can never lose mail in amavisd-new.

Sizing tmpfs

To calculate the correct size for tmpfs, consider this: If you run *n* amavisd-new instances, and each one accepts messages of at most `message_size_limit`, you need this much space:

```
n * (1 + maximum_expected_expansionfactor) * message_size_limit * 7/8
```

The `expansionfactor` is tricky, but a factor of 2 is quite okay (a compressed message—think *.zip or *.rar files here—may grow to twice the original size).

For example, if you have five amavisd-new instances and a 10MB message limit, you would get the following result for the size of tmpfs:

```
5 * (1 + 2) * 10MB * 7/8 = 131.25MB
```

NOTE *Make sure that you have enough physical memory to hold the tmpfs; otherwise, your machine will start swapping memory out to disk, and you will end up with performance that's worse than a regular filesystem.*

Configuring the Optimization

There are a few steps involved in setting amavisd-new up to use tmpfs:

1. Find amavisd-new's `$TEMPBASE` parameter.
2. Create a tmpfs filesystem.
3. Stop amavisd-new.
4. Mount the tmpfs filesystem.
5. Start amavisd-new.
6. Make sure that amavisd-new still works.

First, you need to find out where the amavisd-new `$TEMPBASE` is defined. This is the mount point for the tmpfs that you will create. The default `$TEMPBASE` is `$MYHOME`, which is `/var/amavis` by default. To find out for sure, use `grep` on your configuration file. This example shows that it is set to `$MYHOME`:

```
# grep TEMPBASE /etc/amavisd.conf
$TEMPBASE = $MYHOME;            # (must be set if other config vars use is)
$ENV{TMPDIR} = $TEMPBASE;       # wise, but usually not necessary
    "-f=$TEMPBASE {}", [0,8], [3,4,5,6], qr/infected: ([^\r\n]+)/ ],
```

```
    # adjusting /var/amavis above to match your $TEMPBASE.
    #   directory $TEMPBASE specifies) in the 'Names=' section.
```

Run another **grep** to find out what `$MYHOME` is. In the following example, you can see that the definition of `$MYHOME` is commented out, so it uses the default value:

```
# grep MYHOME /etc/amavisd.conf
# $MYHOME serves as a quick default for some other configuration settings.
# $MYHOME is not used directly by the program. No trailing slash!
#$MYHOME = '/var/lib/amavis';   # (default is '/var/amavis')
$TEMPBASE = $MYHOME;            # (must be set if other config vars use is)
#$TEMPBASE = "$MYHOME/tmp";     # prefer to keep home dir /var/amavis clean?
#$helpers_home = $MYHOME;       # (defaults to $MYHOME)
#$daemon_chroot_dir = $MYHOME;  # (default is undef, meaning: do not chroot)
#$pid_file  = "$MYHOME/amavisd.pid";  # (default is "$MYHOME/amavisd.pid")
#$lock_file = "$MYHOME/amavisd.lock"; # (default is "$MYHOME/amavisd.lock")
#$forward_method = "bsmtp:$MYHOME/out-%i-%n.bsmtp";
$unix_socketname = "$MYHOME/amavisd.sock"; # amavis helper protocol socket
                                  # (usual setting is $MYHOME/amavisd.sock)
$LOGFILE = "$MYHOME/amavis.log";  # (defaults to empty, no log)
    "{} -ss -i '*' -log=$MYHOME/vbuster.log", [0], [1],
```

Now you need to create a tmpfs entry in your `/etc/fstab` file, using the filesystem size calculated in the previous section (“Sizing tmpfs”). The following example uses a 150MB size and limits access to a particular user and group. In this case the user ID is `7509` and the GID is `54322`, which match the user and group `vscan` in the `/etc/passwd` and `/etc/group` files; keep in mind that your system almost certainly has different numbers, and you will need to look them up by yourself:

```
/dev/shm   /var/amavis   tmpfs defaults,size=150m,mode=700,uid=7509,gid=54322 0 0
```

Before you mount `/var/amavis`, make sure to stop amavisd-new with a command such as this:

```
# /etc/init.d/amavisd-new stop
```

Next, mount `/var/amavis` (remember that this is the tmpfs filesystem that you just defined in `/etc/fstab`):

```
# mount /var/amavis
```

Now start amavisd-new again:

```
# /etc/init.d/amavisd-new start
```

Check whether all is well by looking at the logs and examining `df -h` output. In the following example, `/var/amavis` is 100MB, and only 76KB are currently in use:

```
# df -h /var/amavis
Filesystem            Size  Used Avail Use% Mounted on
/dev/shm              100M   76k   99M   1% /var/amavis
```

NOTE *Sometimes amavisd-new leaves stale files in its `$TEMPBASE` directory. To prevent `$TEMPBASE` from getting filled with these files, you can stop amavisd-new daily, remove the stale files, and restart. A daily cron job script such as the following will get the job done:*

```
#!/bin/bash
/etc/init.d/amavisd stop
rm -Rf /var/amavis/amavis-200*
/etc/init.d/amavisd start
```

Configuring Postfix to Use amavisd-new

At this point, Postfix and amavisd-new should each run independently of the other. Therefore, you need to configure Postfix to send messages to amavisd-new and create another `smtpd` instance for message reinjection. The following steps (discussed in the following sections) will integrate amavisd-new into Postfix:

1. Create a transport.
2. Configure the transport.
3. Configure a reinjection path.

NOTE *Because the filtered mail needs a way of getting back into the Postfix queue system without being scanned again, you need a dedicated `smtpd` that doesn't use `content_filter`. This allows amavisd-new to reinject the mail into the system without generating loops. Port 25 is already taken, so you can make a copy of `smtpd` listen to a nonstandard port. This example uses port 10025 on `localhost`.*

amavisd-new also needs a port to listen on. The default of port 10024 on `localhost` is fine.

Creating a Transport Using content_filter in main.cf

The first step in delegating the content processing to an external program is to define the transport that sends messages to the filtering program. Postfix uses the `content_filter` parameter in the `main.cf` file. The parameter expects a notation of *`transportname:nexthop:port`*.

In the example we're working on, amavisd-new is running on the same machine as Postfix, so you can access it at port 10024 on `localhost` (`127.0.0.1`). You need to define the following `content_filter` parameter in the `main.cf` file to make Postfix connect to amavisd-new:

```
content_filter = amavisd-new:[127.0.0.1]:10024
```

Running amavisd-new on a Different Host

If you feel that the filtering load is too much for a single machine, you can run amavisd-new on one or more machines. The `nexthop` part of *`transportname`*`:`*`nexthop`*`:`*`port`* allows you to easily specify a different host for the filter. Consider `vscanners.example.com` in the following example:

```
content_filter = amavisd-new:vscanners.example.com:10024
```

The name `vscanners.example.com` could be any one of the following:

- One machine (through one A record)
- Multiple machines (through multiple round-robin A records)
- Multiple machines (one or more machines with different priority MX records)

Defining the Transport in master.cf

Next you need to define the daemon that will connect to amavisd-new and specify the environment for the daemon. Remember that the daemon can be `smtp`, `lmtp`, or `pipe`. You saw an example of `pipe` earlier in this chapter; it's time to look at the other two.

Defining an ESMTP Transport

If you want to use the ESMTP protocol to send messages to amavisd-new, add the following entries to your `master.cf` file:

```
#==========================================================================
# service type          private  unpriv  chroot   wakeup   maxproc  command
#                       (yes)    (yes)   (yes)    (never)  (100)
# ==========================================================================
...
amavisd-new    unix     -        -       n        -        2        smtp
  -o smtp_data_done_timeout=1200s
  -o disable_dns_lookups=yes
```

There are a few things to note in the preceding entries:

- The special transport `amavisd-new` is a copy of the normal `smtp` transport. Its name must match the transport name that you gave to the `content_filter` parameter that you defined in `main.cf`.

- amavisd-new is quite resource-hungry. Unless you have a fast machine, you might want to leave the maximum number of simultaneous instances at 2.
- The `smtp_data_done_timeout` parameter is the first of two additional settings that modify this daemon's behavior. amavisd-new can take a significant amount of time to process an incoming message, and increasing the timeout after `smtp` sends the message protects Postfix from giving up before amavisd-new is done.
- Because you are probably dealing only with local machine names at this point, the `disable_dns_lookups` parameter disables unnecessary DNS lookups for the `smtp` client.

NOTE *You don't necessarily need a dedicated SMTP transport, because the default* `smtp` *does the job well. However, for performance reasons (and because of the relatively long amavisd-new timeout), it can make sense to customize a transport just for amavisd-new.*

Defining an LMTP Transport

If you decide to use the LMTP protocol to transport messages to amavisd-new (instead of SMTP), add the following entry to your `master.cf` file:

```
#==========================================================================
# service type          private  unpriv  chroot   wakeup   maxproc   command
#                       (yes)    (yes)   (yes)    (never)  (100)
#==========================================================================
...
amavisd-new    unix     -        -       n        -        2         lmtp
  -o lmtp_data_done_timeout=1200s
```

The obvious difference between this entry and the one for the ESMTP transport is that `lmtp` daemon runs instead of `smtp`, and because `lmtp` performs no MX lookups, you don't need to bother with the `disable_dns_lookups=yes` parameter.

Configuring a Reinjection Path

Finally, you need to create a reinjection path that allows amavisd-new to feed messages back into the Postfix queue. It's important that this reinjection path bypass the amavisd-new transport. Otherwise the message will get caught in a loop, where Postfix sends the message to amavisd-new, the mail is reinjected into the Postfix queue, and it is sent back to amavisd-new again.

A reinjection path that bypasses any previously defined `content_filter` parameter looks like this in your `master.cf` file:

```
#==========================================================================
# service type          private  unpriv  chroot   wakeup   maxproc   command
#                       (yes)    (yes)   (yes)    (never)  (100)
```

```
# ==========================================================================
...
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
```

Of all the options in the preceding entry, the one that is *absolutely* essential is the empty `content_filter` parameter. This overrides the `content_filter` parameter in the `main.cf` file. The remaining options override other `main.cf` parameters, including options to turn off restrictions that make no sense for a transport listening only on the localhost network interface.

After putting all of the settings in place, you're ready to test the filter. Remember that changes in `master.cf` require you to reload Postfix.

Testing the Postfix amavisd-new Filter

To test that Postfix and amavisd-new work well together, you must verify that Postfix can send mail to amavisd-new, and that amavisd-new can reinject messages. Testing involves the following steps:

1. See if Postfix listens on the reinjection path.
2. Send a message to Postfix, checking that it sends the message to amavisd-new, and that the message comes back into the Postfix queue.
3. See if a virus scanner detects a test pattern.

Checking the Reinjection Path

Once you've changed the `master.cf` file, run the **`postfix reload`** operation to make Postfix read the revised file and then examine the log file for any complaints. Then, check whether the `smtpd` reinjection daemon is listening on `localhost`, port 10025, as in the following session:

```
$ telnet 127.0.0.1 10025
220 mail.example.com ESMTP Postfix
EHLO 127.0.0.1
250-mail.example.com
250-PIPELINING
250-SIZE 10240000
```

```
250-VRFY
250-ETRN
250-STARTTLS
250-AUTH LOGIN PLAIN DIGEST-MD5 CRAM-MD5
250-XVERP
250 8BITMIME
QUIT
221 Bye
```

Sending a Test Message to Postfix

Postfix should be able to let an uninfected message pass through the system. Send a message from the command line, and track it with the log file messages from Postfix and amavisd-new. For example, you could use the following command to mail your `main.cf` file to `recipient@example.com`:

```
# sendmail -f sender@example.com recipient@example.com < /etc/postfix/main.cf
```

Then take a look at the log file. The bottom of the file should have a session that starts like the following, where Postfix assigns a message ID to the message that you can use to track the message:

```
Jan 31 10:45:08 mail postfix/pickup[10096]: 2788029AB29: uid=0 from=<sender@example.com>
Jan 31 10:45:08 mail postfix/cleanup[10652]: 2788029AB29:
  message-id=<20040131094508.2788029AB29@mail.example.com>
```

The next set of messages should show Postfix handing the message to `localhost` for processing by amavisd-new (unfortunately, Postfix does not log the port number or the name of the transport):

```
Jan 31 10:45:08 mail postfix/qmgr[10097]: 2788029AB29: from=<sender@example.com>, size=1271,
  nrcpt=1 (queue active)
Jan 31 10:45:08 mail postfix/smtp[10660]: 2788029AB29: to=<recipient@example.com>,
  relay=localhost[127.0.0.1], delay=0, status=sent (250 2.6.0 Ok, id=25809-04, from MTA: 250
  Ok: queued as 377D829AB2A)
```

Now amavisd-new scans the message and logs that the message passed:

```
Jan 31 10:45:08 mail amavis[25809]: (25809-04) Passed, <sender@example.com> ->
  <recipient@example.com>, Message-ID: <20040131094508.2788029AB29@mail.example.com>, Hits: -
```

Next, the message comes back into Postfix from amavisd-new for reinjection into the queue. Notice that the second `smtpd` also logs the message ID:

```
Jan 31 10:45:08 mail postfix/smtpd[10658]: connect from localhost[127.0.0.1]
Jan 31 10:45:08 mail postfix/smtpd[10658]: 377D829AB2A: client=localhost[127.0.0.1]
```

```
Jan 31 10:45:08 mail postfix/cleanup[10652]: 377D829AB2A:
  message-id=<20040131094508.2788029AB29@mail.example.com>
Jan 31 10:45:08 mail postfix/qmgr[10097]: 377D829AB2A: from=<sender@example.com>, size=1723,
  nrcpt=1 (queue active)
Jan 31 10:45:08 mail postfix/smtpd[10658]: disconnect from localhost[127.0.0.1]
```

Finally, Postfix relays the message to another host for delivery (it could also deliver locally, if this server happened to be the final destination):

```
Jan 31 10:45:08 mail postfix/smtp[10655]: 377D829AB2A: to=<recipient@example.com>,
  relay=relayhost[10.0.0.1], delay=0, status=sent (250 OK id=1AmrgY-00073g-00)
```

Checking a Test Virus Pattern

Your last test is to simulate a message infected by a virus. You can do this by getting the EICAR test virus pattern (`http://www.eicar.org`) and sending it to Postfix. Any virus scanners that don't have this pattern specifically disabled should be able to recognize it. For example, the following command should send a virus to `recipient@example.com`:

```
# sendmail -f sender@example.com recipient@example.com < eicar.com
```

The log messages look like they did before, up to the point where amavisd-new scans the message:

```
Feb  6 15:48:54 mail postfix/pickup[30051]: 13B9E29AB29: uid=0 from=<sender@example.com>
Feb  6 15:48:54 mail postfix/cleanup[30741]: 13B9E29AB29:
  message-id=<20040206144854.13B9E29AB29@mail.example.com>
Feb  6 15:48:54 mail postfix/qmgr[19295]: 13B9E29AB29: from=<sender@example.com>, size=347,
  nrcpt=1 (queue active)
Feb  6 15:48:54 mail postfix/smtp[30744]: 13B9E29AB29: to=<recipient@example.com>,
  relay=localhost[127.0.0.1], delay=0, status=sent (250 2.5.0 Ok, id=10217-07, BOUNCE)
...
Feb  6 15:48:54 mail amavis[10217]: (10217-07) INFECTED (Eicar-Test-Signature),
  <sender@example.com> -> <recipient@example.com>, quarantine virus-20040206-154854-10217-07,
  Message-ID: <20040206144854.13B9E29AB29@mail.example.com>, Hits: -
```

Seeing that the message contains a virus, amavisd-new alerts `virusalert@example.com` and bounces the message back to the sender:

```
Feb  6 15:48:54 mail postfix/smtpd[30747]: connect from localhost[127.0.0.1]
Feb  6 15:48:54 mail postfix/smtpd[30747]: 639A729AB2A: client=localhost[127.0.0.1]
Feb  6 15:48:54 mail postfix/cleanup[30741]: 639A729AB2A: message-id=<VA10217-07@mail>
Feb  6 15:48:54 mail postfix/qmgr[19295]: 639A729AB2A: from=<>, size=1463, nrcpt=1
  (queue active)
Feb  6 15:48:54 mail postfix/local[30749]: 639A729AB2A: to=<virusalert@example.com>,
  relay=local, delay=0, status=sent (forwarded as 8484829AB2C)
```

```
Feb  6 15:48:54 mail postfix/smtpd[30747]: disconnect from localhost[127.0.0.1]
...
Feb  6 15:48:54 mail postfix/smtpd[30747]: connect from localhost[127.0.0.1]
Feb  6 15:48:54 mail postfix/smtpd[30747]: 7A2FD29AB2B: client=localhost[127.0.0.1]
Feb  6 15:48:54 mail postfix/cleanup[30741]: 7A2FD29AB2B: message-id=<VS10217-07@mail>
Feb  6 15:48:54 mail postfix/qmgr[19295]: 7A2FD29AB2B: from=<>, size=2554, nrcpt=1
  (queue active)
Feb  6 15:48:55 mail postfix/smtp[30744]: 7A2FD29AB2B: to=<sender@example.com>,
  relay=relayhost[10.0.0.1], delay=1, status=sent (250 OK id=1Ap7Ho-00014I-00)
```

Bouncing the message to the sender isn't a particularly good idea, because the sender is nearly always forged in current email viruses, but this is unfortunately the default for amavisd-new.

Scanning for Viruses with smtpd_proxy_filter and amavisd-new

A different and newer approach to content filtering in Postfix is to inspect incoming messages before queuing them. This type of filter is called `smtpd_proxy_filter`. You can use it with amavisd-new, as shown in Figure 12-3.

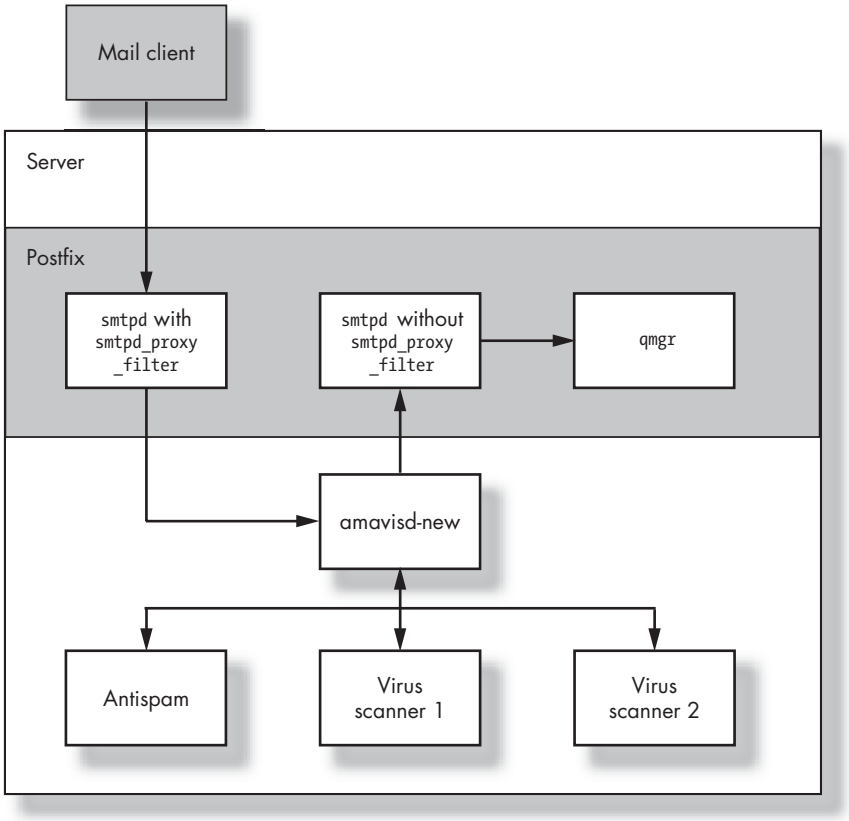

Figure 12-3: amavisd-new integration with Postfix using `smtpd_proxy_filter`

This is how the message would flow if you use smtpd_proxy_filter:

1. A mail client sends a message to a Postfix smtpd.
2. The smtpd (with smtpd_proxy_filter enabled) hands the message to amavisd-new. Notice that this is different from the case with content_filter, which uses the queue manager to send the message to amavisd-new.
3. amavisd-new sends the message to other applications (in this example, to two virus scanners).
4. amavisd-new tells smtpd whether it accepted or rejected the message. If it accepts the message, it reinjects it back into a second smtpd instance, but if it rejects the message, it acts according to your configuration.
5. The original smtpd listens to the amavisd-new replay, accepting or rejecting the message from the client.

NOTE *You can think of smtpd_proxy_filter as the smtpd daemon being broken in two parts:*

- *One part sanitizes the incoming mail with the filter.*
- *The other part does the queuing.*

This section explains how to configure amavisd-new with smtpd_proxy_filter using the general steps described in Chapter 11. You need to perform the following steps to integrate amavisd-new with the smtpd_proxy_filter parameter:

1. Install amavisd-new (described earlier in the chapter in the "Installing amavisd-new" section).
2. Test amavisd-new (described in the earlier "Testing amavisd-new" section).
3. Configure Postfix to use amavisd-new.
4. Test the configuration.

Configuring Postfix to Use amavisd-new with smtpd_proxy_filter

The first step is to define the transports the emails should take into the filtering program. You'll do these two things:

1. Modify the existing smtpd transport to proxy for amavisd-new.
2. Create an additional smtpd instance to have the mail reinjected into Postfix, circumventing any global smtpd_proxy_filter parameter.
3. Test the configuration as described in the previous section.

Modifying the Existing smtpd to Proxy

To make smtpd proxy messages to amavisd-new, append the smtpd_proxy_filter parameter to the existing smtp service in the master.cf file. For

example, the following entry makes smtpd send messages to port 10024 on localhost (remember that these are the default settings for amavisd-new):

```
#==========================================================================
# service type        private   unpriv   chroot   wakeup   maxproc  command
#                               (yes)    (yes)    (yes)    (never)  (100)
# ==========================================================================
...
smtp    inet          n         -        n        -        20       smtpd
   -o smtpd_proxy_filter=localhost:10024
   -o smtpd_client_connection_count_limit=10
```

Notice that -o smtpd_client_connection_count_limit=10 prevents one SMTP client from using up all 20 SMTP server processes defined in the maxproc column. This limit is not necessary if you receive all mail from a trusted relay host.

Also, unlike the process used earlier for the content_filter mechanism, you're not defining a global parameter in the main.cf file, so it will not be necessary to explicitly override it in the reinjection transport.

Creating an Additional smtpd Instance for Message Reinjection

To let the messages reenter the Postfix queue on a non-proxying smtpd instance, you need to add a special instance of smtpd in your master.cf file. This example creates another instance on port 10025 of localhost:

```
#==========================================================================
# service type        private  unpriv  chroot   wakeup   maxproc   command
#                              (yes)   (yes)    (yes)    (never)   (100)
# ==========================================================================
...
127.0.0.1:10025 inet  n        -       n        -        -         smtpd
   -o smtpd_authorized_xforward_hosts=127.0.0.0/8
   -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 receive_override_options=no_unknown_recipient_checks
```

The -o smtpd_authorized_xforward_hosts=127.0.0.0/8 parameter allows the after-filter smtpd to receive remote SMTP client information from the before-filter smtpd. Specifically, the after-filter smtpd will accept any XFORWARD commands sent by a host listed in smtpd_authorized_xforward_hosts. This is very useful for debugging, because the smtpd will use the original client IP address instead of localhost[127.0.0.1].

The remaining parameters lighten the load on the after-filter smtpd, because the before-filter smtpd already did this work.