In this chapter:

- *Why Filter on the Server?*
- *Procmail*
- *Sieve*
- *To Filter or Not to Filter…*

15

Server-Side Mail Filtering

Why Filter on the Server?

An old saying that rings true to email is "less is more." If you subscribe to mailing lists, post to Usenet, or frequent web sites that require your email address as part of the registration process, the character and volume of your incoming email can easily become unmanageable. You may wish to eliminate certain kinds of mail, such as SPAM (unsolicited commercial email), before you ever see it. In the case of quasi-unsolicited mail, or some mailing list traffic, you may wish to file it in a specific mailbox to keep it organized.

Many users manage their email using client-side filters. With client-side filtering, filtering can only take place while the client is running on your workstation. The mail must be downloaded before it can be filtered. Server-side filtering, however, makes downloading the mail unnecessary. The filtering occurs on the server, leaving scarce client resources free for other tasks. If you turn your workstation off and go on vacation for a week, filtering continues. All your incoming mail is organized for your return.

Procmail and Sieve are a couple of popular approaches to user-initiated server-side filtering. Procmail is the granddaddy of mail filtering software. It's the more powerful of the two, probably the most difficult to use, and is the best supported by ISPs, and it has the largest following. Procmail has a larger following primarily because it's been around longer. It's also more flexible, because it permits piping of messages into any arbitrary command or script you can dream up. Sieve is the alternative; it embraces prototypical standards being defined through various Internet drafts, listed at *http://www.cyrusoft.com/sieve/*. Sieve is the new kid on the block, is undoubtedly the most standards-based and the most secure, and has the

potential to be the most appealing to end users. Sieve is more secure than Procmail, because it's more narrowly focused on specific filtering tasks.

Let's take a closer look at each approach.

Procmail

Every type of software, whether it's client software, server software, or software for personal productivity, has implementations that are available free or nearly so. Interestingly, the free implementation is sometimes the best and most popular implementation. Take sendmail, for example—it's the most well known MTA out there. Another good example is Procmail. For nearly a decade, Procmail has been used by email wizards and by the users who depend on their skills. A straightforward but powerful mechanism for flexible mail handling, Procmail does more than just filter incoming email. Depending on a variety of attributes, it can file, forward, respond to, delete, or perform any action that can be described in software, such as shell or Perl scripts.

If sendmail is the universal tool for getting mail from MTA to MTA, Procmail is the universal tool of MDAs. For the most part, anyone who's been exposed to Procmail fits into one of two classes: those who are wildly enthusiastic about Procmail and those who will be once they find the time to learn it.

Procmail was developed by Stephen R. van den Berg in 1990. In recent years, a complement of volunteers have joined forces to help Procmail evolve from a set of useful utilities written by one person to an ongoing effort that could very well outlast any given person working on it.

To use Procmail, you define, refine, and enshrine a series of Procmail filtering rules, which typically use regular expressions to describe common factors in the headers of email messages to which you want to apply some action.* For those who are so inclined, Procmail is a spectacular way to automate the tasks associated with culling through your email, filing the mail you want to keep in folders, and discarding the junk mail you don't want to keep.

One of the appeals of Procmail is that you can pipe the message header, the body, or the entire message into an external process. Once you can pipe email into a script, the actions you can take on the email are limited only by your imagination.

---

* The Unix equivalent of speaking in tongues, regular expressions (or *regex*) are the art of tersely describing text patterns using a variety of tokens and substitution rules. *[aeiou]{2,3}$*, for example, becomes any two- or three-letter word consisting of all vowels. More germane to the topic, however, would be a regex like "*^Subject:\ .*(M|m)ake\ (M|m)oney\ (F|f)ast*", which would catch many of the MMF messages bound for your mailbox. For more information, see *Mastering Regular Expressions* by Jeffrey E. F. Friedl (O'Reilly) or the *egrep*(1) or *regexp*(5) manpages.

Procmail comes with special-purpose facilities that will handle 99% of everything you might want to do with your email, so it's not always necessary to pipe email into a script. If, however, you want to set up a rule that auto-replies to mail by returning the day's C-SPAN television schedule to the sender, Procmail's just the ticket.

How Do You Install It?

There are two ways to run Procmail: either as the system-wide delivery agent or on a per-user basis. Let's assume here that, for whatever reason, it's impractical to use Procmail as your default MDA, so users are left to their own devices to use it.

There are a variety of mirrors for the current version of Procmail. Two sources of information are the Procmail Home Page (*http://www.procmail.org/*) and Infinite Ink's Processing Mail with Procmail site (*http://www.ii.com/internet/robots/procmail/*).

The *procmail.org* site maintains a pointer to the latest stable version of Procmail at *ftp://ftp.procmail.org/pub/procmail/procmail.tar.gz*. Once you download and unpack it, you'll find an INSTALL file in the top-level directory that encourages you to edit the *config.h* and *Makefile*, then do a conventional build.

You can probably get away without editing anything in *config.h* and *Makefile*. On one of our test systems, a modestly loaded Sun SPARC 10 running *gcc* under Solaris 2.5.1, we ran *make* on the unadulterated source and got an error-free build in just under two minutes, forty seconds. The Procmail developers have integrated many of the tasks found in a *configure* script into the Makefile, so building Procmail leaves you with precious little to do but twiddle your thumbs and contemplate the fact that the spammers will soon be washed right out of your hair.

Building Procmail gets you the following programs:

Formail
: Formail is very nearly as useful as Procmail. Not only can you use it within your Procmail recipes, but you can use it in scripts that "munge" incoming mail, or reformat into a different mail storage format. Formail can take a message on standard input and swap the *To:* and *From:* addresses around to produce a return header; it can be used to escape old headers with an "X-" or to add entirely new header fields. It is frequently used at the top of a Procmail recipe file to load the important fields of the current message into variables that might be used in a later recipe.

Lockfile
: Internet messaging can sometimes be a mess of multiple processes all trying to read and write the same files at the same time. File-locking problems have

become the bane of anyone with a vested interest in Internet email. The Lockfile utility is a bullet-resistant* utility for use in mail processing scripts. It's not likely to be needed inside Procmail recipes themselves. However, the odd script you might find yourself writing to convert someone's 10-year-old VMS mail file into a Unix mail file on an active NFS-mounted filesystem could benefit from a good application of Lockfile.

Procmail

The Leatherman Tool of email processing, Procmail takes mail messages one at a time from standard input and processes them according to a file of "recipes." These recipes can, based on a variety of factors, delete messages, file them in flat files or MH-style folders, bounce them to other addresses, or pipe them into other software. The factors on which the actions are based are nearly endless and include such things as header content and message size or external factors like the date, system load, or other random thing,

Mailstats

Mailstats is a nifty little shell script that reduces your Procmail log file down to statistics regarding how many messages and number of bytes (total and average) went to various destinations.

How Does It Work?

Procmail, when run in "per-user" mode, is invoked from the *.forward* file. Procmail first sets environment variables to their default values, reads the incoming mail message to the EOF marker, separates the body from the header, and then, if no command-line arguments are present, looks for a file named *.procmailrc* in the user's home directory ($HOME). The filtering rules in *.procmailrc* determine where to distribute the mail message—usually a folder, but sometimes the message is bounced elsewhere or banished to */dev/null*.

If no *.procmailrc* file is found, or if there is no rule in the *.procmailrc* file that applies to the message, then Procmail stores the message in the user's incoming mail folder.

Simple Examples

Let's cut to the chase. We could fill a book with good ways to put Procmail to work. Our goal here, though, is just to give you some initial suggestions and point you in a useful direction. Let's jump into a sample *.procmailrc* file.

---

* It's always good to be dubious when it comes to file-locking schemes, no matter how good they are.

In Example 15-1, we give three recipes: one effectively deletes messages, one files them in a folder, and one bounces them to another email address.

Example 15-1. Example Procmail Rules File

```
MAILDIR=$HOME/mail              # This directory must exist!
PATH=$HOME/bin:/usr/local/bin:/usr/bin:/usr/sbin
MYADDR=johndoe@imap.unt.edu
LOGFILE=$MAILDIR/procmail.log   # Keep a logfile to aid in debugging

#  RECIPE 1: Filter out annoying SPAM and drop it in the bit bucket
:0 w:
* ^Subject: .*((M|m)(AKE|ake)) ((M|m)(oney|ONEY)) ((F|f)(AST|ast)).*
/dev/null

#  RECIPE 2: Filter mailing list messages into a folder
#
:0 w:                           # Use a lockfile with personal mail folders!
* ^(To|Cc): .*info-cyrus.*
info-cyrus

#  RECIPE 3: Bounce urgent messages to my text pager
:0 c
* ^Priority: Urgent
! mypager@myhost.mydomain
```

Note that the syntax for mailbox and folder names on Cyrus servers is slightly different. Examples of Procmail rules files specific to Cyrus servers are shown in Chapter 9, *Cyrus System Administration*. For brevity, we're assuming the system here uses standard Unix format mail folders in *$HOME/mail* and that the user's incoming mail goes into */var/spool/mail/$LOGNAME*.

Example 15-2 shows a handful of mail messages and descriptions of how our procmail rules would act on them.

Example 15-2. Spam Mail

```
Date: Mon, 23 Mar 1998 05:48:11 +0100
From: Spamford and Son <spamford@junk.com>
To: <kwm@kwm.net>
Reply-To: spamford@junk.com
Subject: MAKE MONEY FAST!!!!!

Our new printing presses let YOU make money three times faster
than our previous printing presses. And NOW, a special offer! Buy
three printing presses, and get a FREE, yes FREE, bail bond.
Hurry, don't wait! Limited time offer.
```

Rule 1 in the procmail rules file (Example 15-1) looks for messages with a subject line that contains the phrase "MAKE MONEY FAST." Because Example 15-2 has such a subject line, Procmail catches it and pipes it into */dev/null*, and it is not delivered.

Example 15-3 is a message posted to a mailing list. Rule 2 looks for all mail either sent directly to or carbon copied to the *info-cyrus* mailing list and files it into a folder called *~/mail/info-cyrus*. Notice that the first line of the rule contains the two flags: a "w" and a ":". The *w* tells Procmail to wait for the filter to finish and check its exit code before filtering the message. The *:* tells procmail to lock the folder until delivery of the message is complete.

Example 15-3. Mailing List Message

```
Return-Path: <owner-info-cyrus@lists.andrew.cmu.edu>
Date: Sat, 25 Sep 1999 14:13:21 +0200 (CEST)
To: info-cyrus@lists.andrew.cmu.edu
Subject: Sieve
MIME-Version: 1.0
Content-Type: TEXT/PLAIN; charset=US-ASCII
Sender: owner-info-cyrus@lists.andrew.cmu.edu
Precedence: bulk

Could somebody post a sample Sieve script?
```

Example 15-4 is an urgent warning generated by a system watcher program. Procmail can be used to bounce incoming messages to another address, including the email address of a text pager. Rule 3 in the Procmail rules looks for messages that are flagged "Urgent." Because the message in Example 15-3 is such a message, Procmail catches it and forwards it to *mypager@myhost.mydomain*, which is the email address of a text pager.

Example 15-4. Urgent Message

```
Return-Path: <root@localhost>
Date: Sun, 26 Sep 1999 18:10:21 +0200 (CEST)
To: operator@localhost
Subject:
Priority: Urgent

Warning: /var is out of bounds (usage 95%).
```

Risks

If you receive a large amount of email, test your Procmail rules carefully. If Procmail is repeatedly unable to deliver mail due to an incorrect rules file, the system mail queue could fill up and cause a system hang.* Trying out an entirely new *.procmailrc* file is a bit like the scene in the movie where the guy has to defuse the bomb by clipping the red or the green wire, only in this case, there's a built-in

* The authors have seen a user bring a Sun Enterprise–class machine to a grinding halt when an error in his *.procmailrc* caused a storm of redelivery attempts. You can add some insurance against this happening on your system by limiting the number of processes per user to some fixed number.

delay of a few minutes before you hear the roar of hundreds or thousands (depending on how big the mailstream or file is that you're trying to process) of messages that are misdelivered. It's par for the course that anyone who pushes the envelope with Procmail has to spend a few days explaining to everyone why they suddenly had copies of old mail redelivered to them at the rate of 20 an hour. Take our word for it, it's always best to try out new rules on two or three test messages.

This is also a good place to mention indiscriminate Procmail user empowerment. If you've got a black box mail server and you give your users unfettered access to change their *.procmailrc* files, it's the same as giving each of them a shell account. In fact, depending on how you've implemented Procmail, it could be worse. If you've got a shared-mission mail server with shell accounts and associated safeguards already in place, then it's no big deal to let users just edit the *.procmailrc* file willy-nilly. If, however, you've got a black box mail server, we strongly advise you to wrap a CGI script around your editing functions—better yet, give users a restricted set of choices regarding what they can have Procmail do, and have the script generate a *.procmailrc* from scratch.

It should be apparent at this point that, as a user-configurable filtering agent, Procmail is wizard-friendly but may not be every end user's cup of tea. That's where Sieve comes in. Granted, Sieve has greatly reduced functionality compared to Procmail, but most users are far less demanding than your average wizard.

Sieve

Sieve, described in a draft RFC, is a language that is used to filter RFC 822 mail messages at the time of final delivery. Sieve is not intended to filter or process content other than RFC 822 messages. Sieve is not dependent on any particular platform, protocol, or mail architecture.

Background

Current filtering schemes abound, each with its own syntax and functionality, none of which interoperate with one another. Much to the dismay of both users and system administrators, mail filters must be translated or ported when a user moves from one client or server to another. If you've ever moved users from a Unix mail system to a Cyrus system, chances are that you experienced problems dealing with your Procmail users. Procmail filters don't port directly from Unix mail systems to Cyrus. There are syntax differences tied to the differences in mailbox format between the two systems. If there were a standard filtering language in place, then software developers and vendors could write filtering interfaces that could use common scripts. Sieve is intended to be just that standard.

Scope

Sieve is not a complete programming language. Not much can be done with Sieve, other than writing mail filters. Sieve was designed to make filtering simple and easy to use for the end user, while protecting the server it runs on by preventing users from doing things that could be destructive, such as making shell escapes or writing loops. Sieve is also designed to be easy to incorporate into GUI email clients, and thus promotes it as a standard across vendors and platforms. In spite of its simplicity, Sieve has all the features necessary to provide the fundamental tools needed to filter RFC 822 mail messages.

To further promote interoperability, Sieve has an extension mechanism that allows "beyond basic" functionality to be added while working within an open standards framework. Several extensions have already been suggested, including:

- Regular expression matching.
- Ability to handle detailed addressing. Detailed addressing permits the sender to bypass the INBOX and send email directly to a folder in a given mailbox. For example, mail addressed to *kwm+sample@themullets.net* will be placed, if the ACLs permit, in the folder *sample* belonging to the user *kwm@themullets.net*.
- A vacation command.
- Ability to set IMAP flags on delivery (e.g., \Deleted, \Answered, or \Seen).
- An *include* command to include an external Sieve program.

Sieve Implementations

Currently, the only freely available Sieve implementation is Carnegie Mellon University Sieve (*ftp://ftp.andrew.cmu.edu/pub/cmu-sieve/*). Sieve is integrated in Versions 1.6 and later of the Cyrus IMAP server.

The current implementation looks for the user's Sieve script in his home directory. Obviously, for sites that run the Cyrus IMAP server on a black box, code would need to look somewhere else for the scripts. Cyrus 1.6 and higher look in a configurable location defined by the *sievedir* setting in the *imapd.conf* file. Developers at CMU do plan to make that change in the next release of Sieve. If you can't wait until the next release, you can do it yourself—the code modification is trivial.

Other changes in the works include support for storing Sieve scripts on a server using ACAP.

Sieve Examples

A variety of good Sieve script examples is provided in the Sieve draft. To give you an idea of what the Sieve syntax is like, here are a sample mail message and some Sieve scripts. For each script, we tell you what happens to the message when the Sieve script processes it. The sample mail message is shown in Example 15-5.

Example 15-5. Sample Mail Message

```
Date: Tue, 8 Jul 1999 19:21:56 -0800 (PST)
From: niceguy@no.shame.org
To: johndoe@work.com
Subject: Make Money Fast!!!!!!!!!!

Wanted, 30 people earn $28 to lose 29 pounds in 31 days!!!
Guaranteed 100%! Don't delay!!! Go to http://www.over-18.com
immediately!!! Limited offer!!! Act now!!!
```

The *require* action declares an extension so that it can be used in the Sieve script. One of the extensions we've already mentioned is the *vacation* extension. Another, *fileinto*, delivers a message into a specified folder. A declaration is required to use an extension. Example 15-6 delivers the message in Example 15-5 into the user's *Spam* folder.

Example 15-6. The require and fileinto Actions

```
require "fileinto";
if header :contains ["from"] "niceguy" {
    fileinto "INBOX.Spam";
}
```

The *reject* action allows you to refuse delivery of a message and return it to the sender. When the message is returned, it's enclosed in a reject form that prints an informative message indicating why the message was rejected. In Example 15-7, our sample message is returned to the sender.

Example 15-7. The reject Action

```
if header :contains "subject" "Make Money Fast" {
    reject "I do not accept spam mail. Sorry.";
}
```

discard quietly throws a message away, as shown in the script in Example 15-8.

Example 15-8. The discard Action

```
if header :contains ["from"] ["niceguy@no.shame.org"] {
    discard;
}
```

Sieve Documentation

Here are some pointers to information on Sieve, which will be useful if you're interested in learning more and watching it progress from draft to standard.

Sieve web site

The Sieve web site, maintained by Cyrusoft International, Inc. is found at *http://www.cyrusoft.com/sieve/*.

IETF drafts

The Sieve draft is available at *http://search.ietf.org/search/brokers/internet-drafts/query.html*. Search for the string "sieve" to find the latest version.

Two proposed drafts for Sieve extensions are also available at the same site. If you search on the string "sieve", you will also find drafts for the Sieve Vacation Extension and an extension to allow specification of IMAP message flags on an IMAP server.

Mailing list and archives

Sieve discussion takes place on the MTA Filters mailing list. To subscribe, send a note to *ietf-mta-filters-request@imc.org* with the word "subscribe" in the body. An archive of the MTA Filters list is available at *http://www.imc.org/ietf-mta-filters*.

To Filter or Not to Filter…

It's clear that Procmail and Sieve have two slightly different aims. Procmail is flexible enough to be used for everything from a self-modifying reactive SPAM filter to a file server, while Sieve would offer the power of server-side filtration with the mechanical ease of client-side filtering. One issue should be covered, though. That issue is whether filtering should be done at all.

There is a handful of ethical and legal gotchas involved in doing server-side filtering, especially if the filtering is managed by the service provider and not the end user. We'll leave you to sort out the various legal issues, but here are a few issues that merit consideration before you launch into providing filtering services.

Silencing the Bullhorn

Most enterprise-wide email systems have a mechanism for sending out broadcast messages to everyone in the organization. Unfortunately, these mechanisms are frequently used for trite "anyone want to buy my Beanie Baby" messages and, likewise, for firestorms of responses about such inane messages (and counter-firestorms to those responses). Ultimately, such threads serve to introduce many of

the recipients to the capabilities of their MUA's client-side filtering. The obvious problem here is that broadcast messages like "There's a large noxious cloud of Benzene descending on campus" won't reach the entire audience.

Actually, for installations that can't or won't control access to such broadcast mechanisms, this could be a good argument for using server-side filtering. Most users don't want to experiment with local filtering rules. If users knew that they could just submit a web form or send an email message to their local mail administrator requesting that they only get "official" broadcast messages, those noxious Benzene clouds would be a lot easier to avoid.

With Friends Like That…

Proactive attempts to rid users of their SPAM could backfire. If you take the narrow definition of SPAM as unsolicited commercial email, there's no way to tell without contacting the user if any given piece of mail was truly unsolicited or not. True, there's a wide number of telltale signs, like message distributions in multiples of 50, message-ID tags that fit certain patterns, or lists of recipients in alphabetical order spanning only 2 or 3 letters of the alphabet. All that needs to happen, though, is to have one message administratively discarded that the user wanted to receive and all your efforts to improve the users' messaging experience will have been for naught.

Some systems are starting to gravitate toward a "spam on the side" approach, where messages that meet certain strict suspicious criteria are filed in a special folder owned by the user. Messages are purged from that folder if they're over a week old. A system like this would be of particular value on servers that implement delivery of messages to numerous local users as links to a message that is only stored once in the physical mailstore.

Loose Cannons and Processes

This issue is more a factor with Procmail than with Sieve and is actually just a warning against loosely restricted end user shell accounts. A user could quite easily cause a process to go spinning out of control and consume an unhealthy amount of system resources by fumblefingering his *.procmailrc* file or by autofiling his mailing list messages so efficiently that he completely forgets them, and they consume great amounts of server storage.

There's no easy way, short of filtering abstinence, to ensure that these problems don't crop up. The best approach for administrators who do want to engage in server-side filtering is to keep an eye on their servers and place as many reasonable restrictions as possible on what a user can cause to happen with their filtering rules.