

Filtering E-Mail with Postfix and Procmail, Part One

Brian Hatch 2002-06-17

Introduction: An Overview of Server Solutions for Spam Reduction

Most folks dislike spam in their e-mail. Spam takes up our network, disk, and cpu resources. It requires that we weed through unwanted messages to find the ones that we requested. (I'm not going to try to convince you that spam is not good, you can check out some of the anti-spam resources listed in the relevant links section below, if you're interested.) Fortunately, we have many points along the network at which we can implement spam filtering, particularly:

- SMTP daemon;
- local mail delivery agent; and,
- the e-mail client

SMTP Daemon

As the e-mail message is received by our SMTP daemon (Postfix, Qmail, Sendmail, etc) we have our earliest chance to reject the offending message. If it is possible to determine that the message is spam at this stage, we are able to keep the message off our systems with the least amount of our own resources. For example if we know that the user 'yet-more-spy-software@example.com' is a spammer, we can reject the message as soon as the 'mail from:' SMTP command is sent. This has the benefit of being able to send back an immediate bounce message, and also means that we do not waste our bandwidth receiving a mail we would just be deleting anyway.

Local Mail Delivery Agent

Once the SMTP server determines that a mail is destined for a local user, it hands off the e-mail to a mail delivery agent to stick it in that user's mailbox. Sometimes this program is the SMTP daemon itself, but often it is not. It is common to use Procmail as a local delivery agent with Postfix, for example, even though Postfix has a delivery agent built in.

The local delivery agent can use its own spam determinations to plop the offending messages into dedicated spam folders or just dump them to /dev/null. Many local delivery agents are able to call external commands to filter messages directly or help make additional friend-or-foe determinations.

The E-Mail Client

The last place you can guard against spam is in your e-mail client. Some mailers can scan the message

headers or content to route spam to separate boxes much like a local delivery agent could. However, this usually requires that the e-mail client retrieves the mail in the same manner as POP/IMAP rather than operating on a local spool.

Filtering Spam with Postfix and Procmail

This article is the first of a three-part series that will help systems administrators to implement the first two methods to filter out unwanted e-mails before they arrive in the end-user's in-box. Server configurations are more powerful and can filter for all users on the system at once, while e-mail client configurations can only protect an individual's right to spam-free e-mail. As systems administrators, most SecurityFocus readers will prefer that the unwanted e-mail doesn't even reach the client at all. Although mail filtering is an option for many STMP servers and delivery agents, this series will only focus on two: Postfix and Procmail, respectively.

An Overview of Postfix

The Postfix system is broken into many different programs each of which run with minimal permissions. The master process camps on port 25 as root and hands over connections to the smtpd daemon, which runs without root privileges. This program deposits e-mails in a queue directory where they are processed by another Postfix program and, eventually, delivered to a local mailbox. For a more detailed view of Postfix' inner workings, see [Postfix-the big picture](#).

The entire Postfix configuration is contained in the /etc/postfix directory. The main configuration file is main.cf, which controls the variables used by all the pieces. The master.cf file controls which processes are run. (For example, you turn off inbound SMTP simply by commenting out the 'smtpd' line in master.cf.) After any change, you simply run 'postfix reload' to reload the configuration. As you might anticipate, 'postfix stop' and 'postfix start' stop and start the whole Postfix system.

Rejecting Spam with Postfix

The Postfix smtpd server has simplistic but effective spam detection abilities. It has a small set of data that it can look at to make a friend-or-foe determination, including:

- Remote machine ip address/hostname
- SMTP session based options (HELO/EHLO/etc)
- Envelope sender
- Envelope recipient
- Message Header/Body

When Postfix determines that a message is spam, it will pause the SMTP conversation for a bit and then

send an error. The delay is to slow down the sending system, partially in an effort to annoy dedicated spamming systems.

If Postfix can determine that a message is spam, it will respond to the sending machine with an error code, which consists of three digit numbers followed by informational text. There are two main kinds of error codes: 4xx codes indicate temporary failure, which means that the sending machine should try the mail again, while 5xx codes indicate a permanent failure. The most commonly used codes are represented in the following table:

Code	Meaning
554	Transaction failed (This is the default for messages using Postfix' REJECT syntax.)
550	Requested action not taken: mailbox unavailable. (Good for sending permanent errors.)
450	Requested mail action not taken: mailbox unavailable. (Good for sending temporary errors.)

For a full list of error codes, see [RFC-821](#). We'll see how we can select error codes in the next section.

When debugging your spam rejection rules it may be convenient to supply 4xx error codes so that the client retries messages again later. This will keep you from losing legitimate e-mail as you tweak your spam blocking rules.

Postfix Map Files

Many of the Postfix configurations we will discuss in this two-part series can employ external map files. These files contain e-mail addresses, host names, or other data that you may wish to accept or reject.

As an example, we'll take a look at a map that could be used by the various smtpd restriction commands:

```
# OK means to accept the message. Useful to make sure that certain mail
# that would satisfy later restrictions gets accepted anyway. Make sure
# you list these rules first.
#
trusted_friend@example.com      OK
grandpa_george@some_domain.org  OK

# REJECT drops the message with a default error message.
#
hostile_domain.com              REJECT

# Any 5xx code means a fatal error, and the mail client should not try again.
# Here we'll make a specific rejection message to send as part of the
```

```
# bounce message, rather than the default REJECT version.
#
ex-girlfriend@some_host.net      554 Get a life and move on.

# Code 4xx means that the client should retry later, it's a non-fatal error.
#
busy-mailinglist@example.org     450 Sorry, we're performing an upgrade of
our mailinglist software, be back on Thursday.
```

The first argument on the line lists some data that will be used by a spam-determination rule, usually a hostname or e-mail address. (For regular expression maps, the first argument is everything between a set of slash (/) characters. Those are covered later.) The remainder of the line is the SMTP result that will be returned if the first part matches. Each Postfix restriction (based on message sender, HELO response, etc) can have different return values. However, they all include the "OK", and "REJECT" options, which mean to accept or reject permanently the e-mail, respectively.

These flat files need to be converted to a Postfix lookup table, generally a hash, dbm, or btree file. Lookup tables are indexed versions of flat files, which means finding entries is extremely fast. Hash and btree files are named with a .db extension, while dbm files are actually two files, one with a .pag and one with a .dir extension. Unless you specifically prefer one form over another, you can stick with your system's default, which you can learn by running the following command:

```
$ postconf |grep database_type
default_database_type = hash
```

In the remainder of the article I'll explicitly use hash files, just to be specific - the choice is up to you. Thus, if I were to include the 'check_client_access' restriction (which we'll get to later) into some rule, I'd be using:

```
check_client_access hash:/etc/postfix/access
```

In the case shown above, our Postfix installation prefers hash (.db) files. In our /etc/postfix/main.cf we will be pointing to such indexes, and will need to name both the file name and the indexing method. Thus, if I were to tell the 'check_client_access' restriction (which we'll get to later) that it should perform lookups in the access file, I'd say the following in /etc/postfix/main.cf:

```
some_configuration_rule = check_client_access
hash:/etc/postfix/access
```

The check_client_access determination will look at the /etc/postfix/access file hash (technically access.db) to lookup keys.

Any time you make a change to files that are being used in this way, you'll want to rerun the postmap

command, and restart Postfix. Rather than manually run `postmap` for each file, you can save some time by creating a Makefile in `/etc/postfix`, such as this one:

```
$ cat /etc/postfix/Makefile
MAPS: map1.db
      map2.db
      map3.db

%.db: %
      postmap $<
```

List all the maps you need (`access.db`, `virtual.db`, etc) in the MAPS section. The bit at the end just tells make to run `postmap` on the plain text file to create the db (or dbm, etc) file. So each time you make a change to a map, simply do the following:

```
# cd /etc/postfix
# make
# postfix reload
```

Most Postfix installations chroot the various daemon programs, which means they cannot see that you've changed the files in `/etc/postfix`. So remember to reload the Postfix system each time you make a change to these files.

Blocking E-Mail Based on the SMTP Client

Postfix can block e-mail based on the machine that is attempting to send the e-mail. All Postfix has available is the machine's IP address and host name (derived from a reverse DNS lookup). There are three main methods available: rejecting machines with no reverse DNS entries, explicit entries in map files, and DNS blackhole lists.

Blocking Machines With No Reverse DNS

Most administrators make sure that there is a valid reverse DNS entry for all their machines. This is useful in many respects, such as allowing the machine to access systems protected by TCP Wrappers. You can have Postfix deny access from machines that do not have reverse DNS entries by creating the following entry in `/etc/postfix/main.cf`:

```
smtpd_client_restrictions = reject_unknown_client
```

Every IP address can have a reverse DNS entry (PTR record in DNS-speak) that maps that IP address to a host name.

Some host-based authorization services use reverse DNS to determine if a machine is allowed to access certain resources. TCP Wrappers, for example, can be used to allow only machines in some_domain.com to access the SSH server. For this reason, most hosts configured by competent administrators will have a valid reverse DNS entry.

Spammers, on the other hand, often use machines with no reverse DNS entry, because then we'd be able to block e-mail based on that information. However, we can block hosts based on the fact that they do not have a reverse DNS entry by using `smtpd_client_restrictions = reject_unknown_client`.

Standard practice encourages PTR records for all IP addresses. However, there is no rule or RFC that says that this is required. There are many machines out there that do not have PTR records, so using this rule will block legitimate e-mail. If you do enable this restriction, be prepared to manually maintain a map file that lists IP addresses of machines that are allowed to send e-mail to you.

SMTP Client Map Restrictions

You can create a map file that lists any combination valid or invalid machines, such as this one:

```
# Whoops, we need to talk to these machines
# but they has no reverse DNS set up:
10.0.10.1      OK
10.0.10.5      OK
# Reject these guys, they keep sending us junk mail
# and won't take us off their lists
spam_central.com  REJECT
```

You tell Postfix to look at this file (let's say it's a hash named `/etc/postfix/client_restrictions`) by adding the following line to `/etc/postfix/main.cf`:

```
smtpd_client_restrictions = hash:/etc/postfix/client_restrictions
```

DNS Blackhole lists

Many Internet users have created lists of machines (IP address) that regularly send spam. Originally, these lists were published and those who wished to block those hosts would import the data into their spam blocking mechanisms. Unfortunately, this wasn't easily automated, and you needed to perform the updates periodically. Paul Vixie came up with an ingenious solution to this problem. He created the Real Time Blackhole List (RBL), which is available via DNS rather than flat files. The RBL is now part of the MAPS (Mail Abuse Prevention System) at mail-abuse.org.

There are now many different sites that offer RBL-style lookups, often referred to as DNSBL (DNS-based Blackhole Lists). You simply configure the DNSBL domains you wish Postfix to handle, and it will check

the client IP address against these lists. If the IP address is registered at any of the DNSBL domains, then the message will be rejected.

If you wanted to use the bl.spamcop.net list you would add the following entry to your main.cf:

```
smtpd_client_restrictions = reject_maps_rbl
maps_rbl_domains = bl.spamcop.net
```

You can list as many DNSBL domains as you wish and all will be queried. To use the MAPS+ RBL as well, your entry would read

```
smtpd_client_restrictions = reject_maps_rbl
maps_rbl_domains = rbl-plus.mail-abuse.org
                  bl.spamcop.net
```

The most difficult part of implementing DNSBL blocking is determining which DNSBL you wish to use. You will need to read about the procedures used by the DNSBL you're considering in order to determine if you believe that their tests and motives are correct - both for adding and removing IP addresses from their databases. This is left as an exercise for the reader.

Personally, I use the DNSBL (which is currently free for 'hobbyists' although you must register) and bl.spamcop.net (which is free, but you should really donate to help support it.) Together these block about thirty e-mails to my server a day.

Bundling Postfix Restriction Options

We've seen three separate smtpd_client_restrictions we could implement. As you might expect, we can enable all of them by listing them together as:

```
smtpd_client_restrictions = hash:/etc/postfix/client_restrictions,
                          reject_unknown_client, reject_maps_rbl
```

The order is important. If you want to be able to allow mail from a domain that is listed in a DNSBL database, then you need to make sure your map file (which contains the whitelisted machine with an OK) is checked before you check the DNSBL listings.

Blocking Spam Based on SMTP Compliance

The next method to block spam involves checking the remote system's compliance with the SMTP protocol, and verifying the data it provides. The rule of the Internet is to 'be strict about what you send and liberal about what you accept'; however, we will break that rule by dropping connections that aren't up to snuff.

SMTP HELO

Every SMTP session is supposed to begin with the client declaring who it is by sending a HELO or EHLO (extended helo) command like this:

```
$ nc mailserver 25
220 mailserver.example.com ESMTPE ReegenMail
EHLO my.host.name.some_where.com
250-mailserver.example.com
250-PIPELINING
250-SIZE 10240000
250-ETRN
250-XVERP
250 8BITMIME
```

Postfix doesn't normally require a HELO/ELHO to receive e-mail, but by including `smtpd_helo_required = yes` in `/etc/postfix/main.cf`, every client must provide a HELO before beginning the rest of the SMTP session. Some spam software does not send a HELO, and by using this option you may successfully block such bone-headed software.

This option alone may not do too much good. The host name sent by the client may be totally bogus, and this option alone doesn't care. However, we can place restrictions on the hostname provided, as seen here:

```
smtpd_helo_restrictions = reject_invalid_hostname
```

There are several restrictions you may place, which I will list here briefly:

- `reject_invalid_hostname` - reject unless the hostname has valid syntax.
- `reject_unknown_hostname` - reject unless the host has a valid MX or A record in DNS.
- `reject_non_fqdn_hostname` - reject unless the host is fully qualified.
- `permit_naked_ip_address` - Postfix will allow dotted quads that are not wrapped in square brackets (*à la* `[127.28.29.1]`) even though it violates the RFC.
- `check_helo_access maptype:mapname` - look up the hostname in the file `mapname` and reject or accept as appropriate.

These settings will block e-mail from hosts that provide invalid information intentionally (to hide the true source of the spam), but will also deny machines that are simply misconfigured. This is often the case with machines that are inside a firewall and have IP addresses or host names that cannot be resolved outside the firewall.

Valid Envelope Format

After the SMTP helo is sent, the client needs to tell Postfix who the e-mail is from (MAIL FROM) and where to send it to (RCPT TO). This communication is supposed to follow RFC-821. Some spam software isn't strict about it's conformance, and we can block spam based on this fact. You'll need to add the following to /etc/postfix/main.cf:

```
strict_rfc821_envelopes = yes
```

Since most SMTP servers are forgiving about RFC-821 compliance, lots of software doesn't follow the spec correctly. This means that enabling RFC-821 compliance may cause you to reject legitimate e-mail.

Conclusion

That concludes the first part of our three-part look at filtering e-mail with PostFix. In the next installment, we will look at sender/recipient restrictions, restriction ordering, and a map file naming conventions before moving on to Procmail in the final article.

To read Part Two of this series, click [here](#).

Brian Hatch is an obsessive security freak and lead author of [Hacking Linux Exposed](#) and co-author of [Building Linux VPNs](#). While he frequently stays up late to write or hack code, he thinks it's much more fun to go to the park and push his daughter in the swing as he delivers horrible puns to his fiancée.

Relevant Links

Unsolicited Bulk Email: Definitions and Problems

Paul Hoffman, Internet Mail Consortium

Fight Spam on the Internet!

spam.abuse.net

Mail Abuse Prevention System

Mail-Abuse.org

Death to Spam: A Guide to Dealing with Unwanted E-Mail

Alchemy Mindworks

Spam and the Internet

Hormel's SPAM Corporate Web Site

Stopspam.org

Coalition Against Unsolicited Commercial E-Mail

[Privacy Statement](#)

Copyright 2006, SecurityFocus