

# Spam Filter with SpamAssassin

Published Sep 08, 2005 @ 9:21:31

- > *Out of my pure hatred for spam, love for qmail/vpopmail/perl, and*
- > *dislike of the complexity of all current 'spam filtering'*
- > *implementations for qmail (and/or vpopmail), I decided to write my own*
- > *perl app to do all the things I wanted (and more) when filtering email.*
- >
- > *Using procmail and the standard spamassassin methods required patches*
- > *for qmail (seekable patch) and all kinds of other ridicules setup, and*
- > *then still didn't work with vpopmail (at least I couldn't get it to). So*
- > *I thought, "Why are we hacking up source codes and changing how stuff is*
- > *processed to accomplish this? After all, qmail's mentality is chaining*
- > *programs together to complete the task, so why don't we filter this*
- > *way?"*
- *Jon Coulter (me), vpopmail mailing list, 8/2/2002*

## Introduction

Let's face it, spam is a pain. We also have to realize that it likely isn't going to go away any time soon. However, the sociological and technical problems that allow spam are beyond the scope of this article. What I want to do is provide a drop-in spam filter for those of us that use vpopmail (with qmail) using SpamAssassin.

## A Little Background

SpamAssassin is a great tool for filtering spam. It is open source, written entirely in perl (*almost*) and has an active development community that is always updating the software to better filter spam e-mail messages.

The problem with SpamAssassin's default implementation is that it is hard to insert into the qmail stream. Even harder is having per-user settings or filters in an environment like vpopmail. So a couple years ago I wrote a script that would make use of the Mail::SpamAssassin perl module (the core of SpamAssassin) and allow me to customize the configuration at runtime (per-user) and add any additional logic I wanted. For example, there were situations where I wanted to block every email from a certain domain name, or a subject line that matched a specific regular expression. Neither of this (especially the latter) is possible with SpamAssassin itself. But with this custom script, I'm able to configure many more "not necessarily spam-related" filters, and while I was at it, allow it to be configured on a per-account basis **and** have it work with vpopmail!

The script itself is written in perl, and the contents of it are too long (and at the same time not very interesting) to include pieces of it specifically in this article. Instead, this article will aim at installing, configuring and using the script. You can download the script at the bottom of the page, under the attachments, and view the source code all you want. Please provide any comments and/or improvements in the forums (link at the bottom of the page). [page] **Script Installation/Usage**

First, make sure SpamAssassin is installed. This is required so that we can load 'Mail::SpamAssassin' for spam checks.

Download and copy spam-filter.pl to your ~vpopmail/bin/ directory. (See attached files). Make sure this script can be executed by the vpopmail user (chown vpopmail spam-filter.pl && chmod +x spam-filter.pl)  
Setup 2 vpopmail accounts, one for spam mail and one for real mail.

Note: You can just ignore spam mail if you want, but I don't recommend it,

as there may be false-positives. For this example, I'll use 'userclean@domain.ext' and 'userspam@domain.ext'

Now create a .qmail- (.qmail-username in this case) file for the user that the mail actually goes to (cannot be either of the above users) in the ~vpopmail/domains/domain.ext/ directory.

In the file, put something to the effect of:

```
| /usr/local/vpopmail/bin/spam-filter.pl -C "/usr/local/vpopmail/bin/vdelivermail 1 user@domain.ext" -X "/usr/local/vpopmail/bin/vdelivermail 1 userspam@domain.ext" -x 100 -c 100 -S
```

(yes, all one line, and including the '|')

Here is a full list of the command line options (copied and pasted from the source of spam-filter.pl):

```
# opts (all optional):
# -d -- just delete anything marked spam (default)
# -S -- quietly discard message (no bounce) if spam
#     note: this still executes -X command, if specified
#         turn this off by using -K:
# -K -- if -S is specified, then also 'kill' the message all together (avoid exec of -X)
# -f /some/file.cf -- the spamassassin config file
# -r -- report the spam to any report databases
#     (not yet implemented)
# -M 'eat me' -- the spam bounce message (Defaults to 'DIE SPAM')
# -p /path/to/file -- prefs file
# -C "/path/to/program arg1 arg2" -- Pipe to given program
#     Note: this happens on success (Still exists 0)
#     If exit from this program is not '0', then this will exit 111 (tmp fail)
# -c 111 -- exit with give code if -C program fails (defaults to 111)
# -X "/path/to/program arg1 arg2..." -- pipe to this program on spam
#     Note: give a value of 'C-' to use -C's value
# -x same as -c, only used with -X
#
# Added 8/26/2002 -- use spamc/spamd to check spam
# -s -- use spamc/spamd instead of loading the stuff yourselves
#     note: spamc doesn't allow reading of SA prefs file, this is a spamd option
#         (See spamd/spamc docs for details)
#     note: the prefs file things that check headers (like subject scans)
#         will be ignored, since we're not parsing this in-process
#
# Added 9/3/2002 -- allow rewrite of emails that are going to be sent to
#     a program via -X or -C
# -w -- reWrite email message to reflect that its spam
#
# Added 9/9/2002 -- use Mail::SpamAssassin if
#     spamd appears to be dead (not running)
# -F -- fallback to Mail::SpamAssassin
```

See the source code of spam-filter.pl to see what the command line arguments are.

Another optional usage would be to use it as a 'stop processing .qmail file if spam' type of program. This was its original usage (and, of course, will still work as such). For this, you still have 2 vpopmail accounts (or however many you want), and set it up as so (this goes in the .qmail-realuser file):

## Spam Filter with SpamAssassin

```
# all mail will go to this account
| /usr/local/vpopmail/bin/vdelivermail 1 userspam@domain.ext

# this will 'stop' processing if its a spam message
| /usr/local/vpopmail/bin/spam-filter.pl -M 'die spam, die!'

# only 'clean' mail will be delivered this far
| /usr/local/vpopmail/bin/vdelivermail 1 userclean@domain.ext
```

This will allow *\*all\** mail to be delivered to userspam@domain.ext (even non-spam e-mail), and only 'clean' e-mail will go to userclean@domain.ext. This will also bounce with the error of 'die spam, die!' to the mail server it came from. You can have it quietly ignore spam message (i.e., don't bounce) by using the -S flag.

Or... you can use as many combinations of -C and -X programs as your imagination can throw at you. I've tried to make this as flexible as possible. [page] **But wait... there's more!**

First of all, it should be noted that this script can also make use of the spamd/spamc combo rather than forcing perl to read in and use Mail::SpamAssassin for every single message (which is a big overhead). You'll need to read the spam assassin documentation on how to setup spamd/spamc, but for reference, the -s option will tell the script to attempt to use spamd/spamc instead of Mail::SpamAssassin. Including a -F (for 'Fall-back') will force spam-filter.pl to fall-back and use Mail::SpamAssassin if spamd does not appear to be running.

You can do special bounce filters and white-list things too! This happens by creating a .spamprefs-<user> file in the domain's home directory (so, the same directory as the .qmail-<user> file). The '<user>' should be the same user as the .qmail-<user> file you created above. You can also specify which preferences file to read with the -p flag. The syntax and options in this file are like so:

```
# are we disabled?
# set to 1 to have the whole program exit with 0
# note: -X and -C programs are still executed
# (well, only -C of course, since it is never marked as 'spam')
disabled:          0

# white-listed users
# this doesn't go through spam assassin at all
allow_from:       jon@yahoo.com
allow_from:       @ledscripts.com
allow_from:       services@ebay.com

# include values from another file:
allow_from:       :include:/etc/email_allow_from

# bounce these people no matter what
bounce_from:      badstuff@somedomain.ext

# an entire domain:
# how often does somebody from microsoft email you anyway?
bounce_from:      @microsoft.com

# based on subject
# based on an exact match
bounce_sujbect:   blah blah blah

# perl regex
```

```
bounce_subject_regex:  .*boost your sales.*

# max message size (in bytes, of course)
# the default is 5mb
# if you set this, anything over the limit will bounce with 'too large' error
max_msg_size: 2000000
```

Notice that all directives can have multiple entries (though some, like `disabled` and `max_msg_size` obviously will only read the first). Also note that you can use the `:include:` directive (similar to `/etc/aliases`, for those of you familiar with `sendmail`). The contents of the included file should have each value on its own line. For example, here would be `/etc/mail_allow_from`:

```
user2@foo.com
@test.com
@ebay.com
myfriend@yahoo.com
```

### Conclusion

There are still more options put into this (i tend to go all out when i do something ). Look around the source code and see what other things exist. Also, please **please** feel free to make suggestions and/or comments. Please post them in the forums following the link at the bottom of this page.

- Jon Coulter