How to integrate spamassassin with exim

Derrick 'dman' Hudson

November 6, 2003

Contents

1 Introduction

We all know what spam is and want a solution to the problem. This document describes some ways of using spamassassin with exim to reduce the spam in your inbox.

2 Background and Version Information

This document started as an effort by myself and several other participants on exim-users (exim-users@exim.orgexim-users@exim.org) to integrate SpamAssassin into exim to perform the scanning and tagging prior to delivery to the user. My initial goal was to continue using my filter rules (in exim's filter language) but still have the benefits of SA.

My system runs Debian GNU/Linux. In particular I use a mixture of "testing" and "unstable". In the below description exim v3.x and all spamassassin versions are from the debian packaging. Since exim v4.x was not available in the debian repository until recently the below references to exim4 refer to locally custom compiled installations.

The effort began with exim version 3.33 and spamassassin 1.5. Since then I upgraded my setup with exim version 3.34 and spamassassin versions 2.01 and 2.11.

After that point I upgraded to exim 4.01 and later to 4.04. With those versions of exim I used spamassassin 1.5, 2.01, 2.11 and 2.20.

As of July 2003 I was running exim version 4.05 and spamassassin 2.55 integrated using sa-exim 2.0. I used several releases of SA and sa-exim between "before" and "then". Now my mail infrastructure bears no resemblence to that described here, but I provide this document because some still find it useful.

Consequently:

1. The portions of this document describing the exim3 setup exists solely for historical reference. If you are using v3.x, then please upgrade

ASAP! (This urging is specifically for debian users because debian "stable" (aka woody) is stuck at version 3.36. exim 4.00 was released and declared *stable* by Philip well over a year ago. Finally "unstable" (aka sid) has an exim4 package containing version 4.20.)

2. I have not tested any newer releases of either tool. If readers provide version-specific information for these setups then I will include that information here. Otherwise it is *your* responsibility to comprehend the principles of the methodology described and determine whether or not this method and implementation is compatible with newer releases of the relevant software.

3 Scanning

There are two mechanisms for invoking SpamAssassin. The first is the `spamassassin` Perl script. The second is the `spamc` C program. With the `spamassassin` script an entire perl process is initiated, all of the rules are read, the regexes compiled, and then the scanning is performed *for each email message*. This entails a fair amount of overhead. The alternative method is to run a long-running daemon (`spamd`) so the startup costs of perl and spamassassin are incurred only once in a great while. Then the lightweight `spamc` program is used to pass the message to the daemon and return the result of the scanning. This document uses the spamc/spamd method.

First the spamd daemon needs to be started. The exact procedure for doing this will certainly vary from OS to OS. The following instructions apply to debian, because that is what I have experience with.

Secondly you will need to instruct exim to pre-process each email with spamassassin. To do this system-wide requires, at the very least, editing your exim.conf file.

3.1 Starting `spamd`

3.1.1 Old SpamAssassin (<= 2.31)

*You **really** shouldn't use any version of spamassasin older than 2.55 in the first place!*

*Note: if you are **NOT** using the *Debian* package, then you don't have this file and you don't have the init script referenced below. Check with your operating system vendor to determine the proper course of action.*

Edit /etc/default/spamassassin to start 'spamd' at boot time and to and *not* add a From_ header (a line starting with "From ") at the top of the messages. The From_ header will really break your mail because the From_ header is only for mbox mailboxes and is used to separate messages.

```
# Change to one to enable spamd
ENABLED=1
OPTIONS="-F 0"
```

Now start spamd (as root) :

```
# /etc/init.d/spamassassin start
```

3.1.2 New(er) SpamAsassin (2.40 . . . 2.55)

*Note: if you are **NOT** using the *Debian* package, then you don't have this file and you don't have the init script referenced below. Check with your operating system vendor to determine the proper course of action.*

Edit /etc/default/spamassassin to start 'spamd' at boot time. (the From_ header *mis*feature mentioned above was removed in version 2.40)

```
# Change to one to enable spamd
ENABLED=1
```

Now start spamd (as root) :

```
# /etc/init.d/spamassassin start
```

XXX - discuss report_safe=0

3.1.3 New(est) SpamAsassin (> 2.55)

Who knows? It doesn't exist at the time of this writing (August 2003). :-)

3.1.4 Verification

Once (you think) you have spamd up and running, verify it to make sure. This is very simple to do. Simply take an email message (just one!) and feed it to `spamc(1)`. For example, if the message is saved in ./test.mail, run
`$ spamc < ./test.mail | grep '^X-Spam'`
If spamassassin is up and running then you'll see some information like

```
X-Spam-Status: No, hits=3.7 required=6.0
X-Spam-Level: ***
X-Spam-Checker-Version: SpamAssassin 2.55 (1.174.2.19-2003-05-19-exp)
```

or

```
X-Spam-Status: Yes, hits=9.3 required=6.0
X-Spam-Level: *********
X-Spam-Checker-Version: SpamAssassin 2.55 (1.174.2.19-2003-05-19-exp)
X-Spam-Report:   ---- Start SpamAssassin results
X-Spam-Flag: YES
```

(for a little more fun, see Appendix A)

3.2 Integration Method 1

This method does not require any modification to the exim source code. The only action required is editing some configuration files. Once configured, an email will be handled as follows :

1. exim receives the message and stores it in the queue

2. exim *delivers* the message to itself. Specifically, the message will be transformed by the *transport_filter* then passed on to the destination of the pipe. The destination of the pipe is exim itself; communicating via the BSMTP (Batched SMTP) protocol.

3. exim receives the message again (as a new message) and queues it. This time, however, the flag $received_protocol is set to `spam-scanned` in order to avoid a mail loop that can be created here.

4. exim delivers the ("new") message a second time. This time the message is delivered as it normally would (to the user's inbox or through procmail or via expanding their .forward file or whatever).

As you can see in the handling above, exim handles each message twice, giving it a new queue id the second time. In addition, spamassassin scans each message separately for each recipient.

3.2.1 exim 3

In the *transports* section of your exim.conf create a transport for delivering the message back to exim after tagging it with SA. You should already be aware that the order transports are specified *is not* irrelevant.

```
# SpamAssassin
spamcheck:
    driver = pipe

    command = /usr/sbin/exim -oMr spam-scanned -bS
    transport_filter = /usr/bin/spamc

    bsmtp = all

    home_directory = "/tmp"
    current_directory = "/tmp"
```

```
    # must use a privileged user to set $received_protocol
    # in the second exim process!
    user = mail
    group = mail

    return_path_add = false

    log_output = true
    return_fail_output = true

    prefix =
    suffix =
```

Now add a director that will direct all messages that haven't been scanned to be delivered using the transport you defined above. You should already be aware that the order directors occur is significant. You will most likely want to place this director above all of the others, but you may have the need or desire for slightly different behavior.

```
# SpamAssassin
spamcheck_director:

    # do not use this director when verifying a local-part at SMTP-time
    no_verify

    # When to scan a message :
    #   -   it isn't already flagged as spam
    #   -   it isn't already scanned
    #   -   it didn't originate locally (as long as I don't harbor spammers :-))
    condition = "${if and { {!def:h_X-Spam-Flag:} {!eq {$received_protocol}{spam-s
    driver = smartuser
    transport = spamcheck
```

3.2.2 exim 4

In the *transports* section of your exim.conf create a transport for delivering the message back to exim after tagging it with SA. You should already be

aware that the order transports are specified is irrelevant.

```
# SpamAssassin
spamcheck:
    driver = pipe
    command = /usr/local/bin/exim4 -oMr spam-scanned -bS
    use_bsmtp = true
    transport_filter = /usr/bin/spamc
    home_directory = "/tmp"
    current_directory = "/tmp"
    # must use a privileged user to set $received_protocol on the way back in!
    user = mail
    group = mail
    log_output = true
    return_fail_output = true
    return_path_add = false
    message_prefix =
    message_suffix =
```

Now add a router that will direct messages that haven't yet been scanned to be delivered using the transport you defined above. You should already be aware that the order routers occur *is* significant. You will most likely want to place this router below the routers that handle non–local domains, but you may have the need or desire for slightly different behavior. Be sure to read the exim specification to understand what the various options mean and how to apply them to your setup.

```
# SpamAssassin
spamcheck_router:
  no_verify
  check_local_user
  # When to scan a message :
  #   -   it isn't already flagged as spam
  #   -   it isn't already scanned
  condition = "${if and { {!def:h_X-Spam-Flag:} {!eq {$received_protocol}{spam-sca
  driver = accept
  transport = spamcheck
```

3.2.3 Explanation

As the comment on the *condition* flag indicates, this router (or director) is used only messages that meet the following criteria. You are free to adjust this condition to suit your site.

- wasn't received from spamassassin (determined by $received_protocol)
- wasn't received via a pipe from a local user (the assumption is that your local users won't be generating spam, so scanning their messages is wasteful of resources)
- isn't already flagged (this is safe because the only flag that is checked is a positive flag; spammers can tag their spam for you, but they can't tag it as being clean).

3.3 Integration Method 2

Since version 4.00, exim includes an interface designed specifically to integratn an external program to make policy decisions. This interface is called "local_scan". Marc Merlin maintains sa-exim, an implementation of the local_scan API which uses Spamassassin as the decision mechanism.

sa-exim has many advantages over the technique detailed in Integration Method 1 above :

- each message is handled by exim only once
- there is less overhead to scanning (because each message is handled only once)
- tracking a message in the logs is easier (because each message is handled only once)
- statistical reporting (eg by eximstats) won't be grossly skewed (because each message is handled only once)

- spam messages can be rejected during the SMTP session — this maintains the reliability built in to the SMTP protocol while at the same time relieving your system of the burden of undeliverable bounces or delivering the spam to the user

For more details on sa-exim, installation instructions and source code see Marc's site: http://marc.merlins.org/linux/exim/sa.html.

(A note for Debian users: the (recent) exim4 package includes the "dlopen" patch Marc also maintains. The dlopen patch is a local_scan implementation that uses the dlopen(3) system call to dynamically load a separately compiled local_scan function from a .so. This eliminates the need to recompile exim when you want to change/upgrade the local_scan implementation.)

I recommend using sa-exim instead of the methods described earlier in this document.

3.4 Integration Method 3

Another way to integrate SpamAssassin with exim is to use Tom Kistner's 'exiscan' tool. I have not used exiscan so I can't provide details here. I recommend at least reading Tom's documentation and considering whether or not exiscan is the tool best suited to you (http://duncanthrax.net/exiscan-acl/).

4 Acting (Filtering/Rejecting)

Thus far the spam hasn't been dealt with; the messages have only been tagged as to whether or not it is spam. Users must now decide what they want to do with messages that have been tagged as spam. A variety of tools that can be use for filtering including procmail, maildrop, and exim.

4.1 exim filter

Here is an example of a portion of an exim filter. It takes any message tagged as spam and stores it in a mail folder. This is a good approach because no mail is lost, and no innocent person receives bounce messages as a result of their address being forged.

(exim 4) For details on what a filter file is and where it goes see section 21.10 of spec.txt[1] (specifically the 'allow_filter' option) and section 5 of filter.txt[2].

(exim 3) For details on what a filter file is and where it goes see sections 24.7 (specifically the 'filter' option) and 50.3 of spec.txt[3] and section 5 of filter.txt[4].

```
if
    $h_X-Spam-Status: contains "Yes"
        or
    "${if def:h_X-Spam-Flag {def}{undef}}" is "def"
then
    logwrite "    => junk : SPAM"
    save $home/Mail/junk/spam/
    finish
endif
```

5 Further Discussion

5.1 Managing a SpamAssassin Installation

One of the issues with spamassassin is the need for an up-to-date set of rules and scores. I found that as my copy of spammassassin gets older an increasing amount of spam slips by and ends up in my inbox. This means you should ***always*** run the latest release of spamassassin and *not* run an older release.

5.2 spambayes

This subsection is provided FYI in case you happen to care what tools I am using now :-).

Very recently (2003-08-11) I started using spambayes[6] in place of spamassassin. spambayes uses an entirely different approach to identifying spam than spamassassin uses, although I have recently been informed that spamassassin incorporates the same bayesian filter technique[7].

spambayes is working well for me, even though it doesn't provide server-side scanning and SMTP-time rejection. ***YMMV!***[1]

(Naturally, if you are trying to implement server-side content filtering for a several (or more) users then spambayes is not suitable for your deployment and you should definitely use spamassassin instead.)

5.3 The Most Effective Block

This may be a bit premature, but I've noticed lately that neither spamassassin nor spambayes are flagging a large number of messages as spam ... yet spam isn't appearing in my inbox. Statistical reports on my mail server indicated that on the order of 10% of the messages processed were rejected at the door[2]. The most effective blocks appear to be (in descending order of rejects) :

- "freemail" sender/client sanity check
- Syntactically invalid IP address literal in HELO command
- Not fully-qualified hostname in HELO command
- My own hostname presented in HELO command
- Domain of sender address doesn't exist[3]

[1] Your Mileage May Vary

[2] note that the numbers are a bit skewed due to proper retries caused by DNS-based sanity checks

[3] this is hard to measure because DNS errors are reported as "temporary" (meaning the server must retry the delivery) and occaisionally this check temporarily delays legit

- My own IP address presented in HELO command

Naturally these checks must all be done at the mail server. The freemail access check appears, by far, to be the most effective. One explanation for it can be found in http://jimsun.linxnet.com/misc/postfix-anti-UCE.txt.

A More Fun With SA

Since I laready have this written, I may as well pass it along. If you want to see only and all of spamassassin's report on a message, pipe the output of `spamc(1)` or `spamassassin(1)` through the following awk script:

```
/^ +[[:digit:].]+ points/ {print "(score=" $1 ")."}
```

Alternatively, obtain just the report with this script:

```
BEGIN {flag=0};
/^[^[:space:]]/ { ($1 ~ /X-Spam-Report/) ? flag=1 : flag=0};
{if (flag) print };
```

B Reader's Comments

(XXX; to be filled in)

References

[1] Philip Hazel
Exim Specification (v4.x)
http://www.exim.org/exim-html-4.20/doc/html/spec.html

[2] Philip Hazel
Exim Filter Specification (v4.x)
http://www.exim.org/exim-html-4.20/doc/html/filter.html

[3] Philip Hazel
Exim Specification (v3.3x)
http://www.exim.org/exim-html-3.30/doc/html/spec.html

[4] Philip Hazel
Exim Filter Specification (v3.3x)
http://www.exim.org/exim-html-3.30/doc/html/filter.html

[5] *sa-exim*
http://marc.merlins.org/linux/exim/sa.html

[6] *spambayes*
http://spambayes.sourceforge.net/

[7] *spambayes - How the Bayesian scoring works*
http://spambayes.sourceforge.net/background.html

[8] *Paul Graham's essay on Bayesian scoring*
http://www.paulgraham.com/spam.html