BetterDocumentation SpamdReadme

spamd/README

The following is the text of spamd/README, based off of revision 233282.

Please feel free to edit it as much as you like to make it more useful. Periodically the version in Subversion will be updated to incorporate some of the changes.

To see the latest version in Subversion, click here

Feel free to write comments about your changes in the Comments section (at the bottom).

```
SpamAssassin Daemon
===================

The purpose of this program is to provide a daemonized version of the
spamassassin executable.  The goal is improving throughput performance for
automated mail checking.  This document is a brief synopsis of how spamc/spamd
work, and how to use them effectively.


The Server: spamd
-----------------

spamd is the workhorse of the spamc/spamd pair -- it loads an instance of the
spamassassin filters, and then listens as a daemon for incoming requests to
process messages.  By default, spamd listens on port 783, but this is
specifiable on the command line.

/*
 * FIXME: The following paragraph(s) have to be updated as childs are now
 *        pre-spawned
 */
When spamd receives a connection, it spawns a child to handle the request.
The child will expect to read an email message from the network socket, which
should then be closed for writing on the other end (so spamd receives an EOF).
spamd will then use SA to rewrite the message, and dump the processed message
back to the socket before closing the connection.  The child process then dies.

In theory, this child-forking should be quite efficient, since on most OSes
the fork will not actually copy any memory until the child attempts to write
to a memory page, and then only the dirty page(s) will be copied.  This means
the entire perl engine and the SA regular expressions, etc. will only be
loaded once and then be reused by all the children, saving a lot of overhead.


The Client: spamc
-----------------

spamc is the client half of the pair.  It should be used in place of
'spamassassin' in scripts to process mail.  It will read the mail from
stdin, and spool it to its connection to spamd, then read the result back and
print it to stdout.  spamc has extremely low overhead in loading, so it should
be much faster to load than the whole spamassassin program (and a perl VM).


Installation
------------

/*
 * FIXME: This chapter has to be updated, 'make install' works, more init
 *        scripts
 */

Simply copy the two executables to where you want them.  Then, configure your
system to run spamd in the background, and where your mailer invokes
'spamassassin' instead invoke 'spamc'.  It's that easy!

There's a Red Hat/Mandrake-style startup script called 'spamassassin'
```

in this directory, suitable for installation in /etc/rc.d/init.d .

Security

Since spamd effectively has both read and write access on all of the mail
which passes through it , you may want to keep security in mind. Depending
on the nature of your set-up. If you are installing it on a site-wide
basis at least some caution is advisable.

System-Level Security

spamd has the facility to run as a non-root user, this has potential security
payoffs. If a fault is found in spamd or spamassassin code, any third party
linked-libraries or imported perl modules there is the potential for abuse of
both the running uid of spamd, and the uid of the username supplied by spamc
(and this could be any user).

When run as root, spamd will change uid's to the user invoking spamc in order
to read and write to their configurations. This functionality is not possible
if spamd does not run as root and is a disadvantage if you rely on this. If you
use mysql or LDAP for per-user configuration there is no reason in the world
to run as root, and this remains fully functional.

If you do not need to let your users define their own rules, maintain
their own whitelists, or have non-world-readable home and ~/.spamassassin
directories, then just set spamd up to run with the "-u username" option.
Since spamd can use auto-whitelisting, which requires it maintain a
database of email addresses on-disk, you should use a non-"root" but
non-"nobody" user: "mailnull" or "mail" are good choices, or even create a
"spamd" user.

If you plan to use Razor or Pyzor, please note that they both rely on
their external configuration files in ~/.razor and ~/.pyzor being
readable, and Razor will try to write to a log file in
~/.razor/razor-agent.log that must be writable (Razor will complain about
'unblessed references' in this case). You may find the -H switch to spamd
to be useful; it allows you to set a 'helper home directory' that will be
used as $HOME when external helpers like Razor, Pyzor and DCC are run.

The Bayesian Classifier

If you plan to use Bayesian classification (the BAYES rules) with spamd,
you will need to either

  1. modify /etc/mail/spamassassin/local.cf to use a shared database of
  tokens, by setting the 'bayes_path' setting to a path all users can read
  and write to. You will also need to set the 'bayes_file_mode' setting
  to 0666 so that created files are shared, too.

  2. Alternatively, let the users train their individual Bayes database.

http://wiki.apache.org/spamassassin/SiteWideBayesFeedback can be very
helpful here.

We have implemented an auto-learning algorithm (option 'bayes_auto_learn', on
by default) which can use high-scoring and low-scoring (options
'bayes_auto_learn_threshold_spam' and 'bayes_auto_learn_threshold_nonspam')
mails to improve classification efficiency.

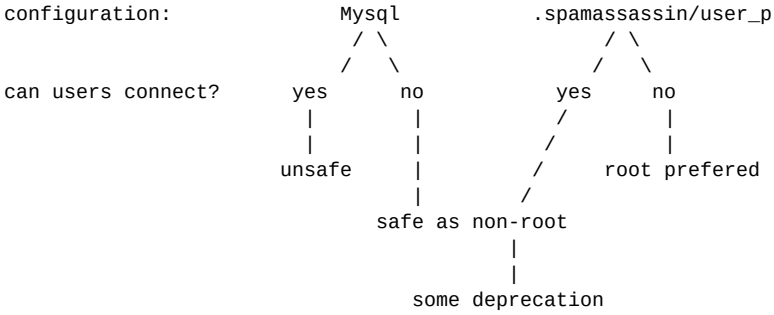
Security And User-Spoofing Clients

Since spamd makes no effort to authenticate the username supplied by
spamc, it is easily possible for malicious users invoking modified

spamc clients to make spamd:

 (1.) read (and hence determine) the contents of other users configurations
 (2.) change the contents of other users configurations (whitelisting)
 (3.) grab CPU time as that user -- this is an issue on ulimit'd systems

If users do not have the opportunity to invoke spamc themselves, and
the network is secure, running spamd as root is the preferred option,
Be clear that the issues above dont affect you. Note: if you use mysql
or LDAP for per-user configuration on systems, you will remain vulnerable
to (1.) and (2.).



If you use spamd across a network and spamc connects from other hosts, you
should ensure (as with all services) the security of your network segments.
Mail is sent as plaintext, and is prone to packet sniffing and spoofing
techniques if you are on an insecure network. If you cannot avoid this consider
using an encrypted transport layer, such as a VPN, ssh tunnel or similar, or
using an SSL-enabled spamc (see 'SSL Support' below).

Performance

So how much faster is this than just using 'spamassassin'? Well, on my 400MHz
K6-2 mail server, spamassassin process a 11689 byte message in about 3.36
seconds, spamc/spamd processes the same message in about 0.86 seconds, or about
4 times faster. With bigger messages, the difference is less pronounced; a
115855 byte message takes about 5 seconds with spamassassin, and 2.5 seconds
with spamc/spamd, or about 2 times faster. However, if many messages are being
processed in parallel, the spamc/spamd combination will likely be much more
efficient, since spamassassin has much higher overhead starting up, and will
consume more non-shared memory than will spamc/spamd. For example, on the
115855 byte message, spamc consumes *no* heap memory (and very little on the
stack), where spamassassin uses over 15MB of heap space and a peak of 3.5M.
In processing the 115855 byte message 10 times in parallel, spamd uses just
22M of heap, with a peak of only 2.5M spamassassin would have used 150M
total, and a peak of up to 35M to do this same job.

Regarding how much resources to allocate for spamd, Francesco Potorti reports
'On a Sun Ultra60 with 512MB memory, I found that 20 is a reasonable number
(for --max-children), and maybe it could be increased. In fact, the memory
footprint of a single Perl interpreter for spamd is about 20MB, but the total
memory occupied by several concurrent spamd processes is not much higher. In
peak activity periods, with load average around 15, more than 13 spamd
processes running or sleeping, and many other amavis and sendmail processes
active, the total memory used was around 350MB, plus about 200MB on swap.'

Shared Library

spamc can now be used as libspamc.so; simply run "make spamd/libspamc.so" at
the top level. Thanks to Liam Widdowson <liam@inodes.org> for this patch.

SSL Support

```
If you have the OpenSSL headers and libraries installed, you can build an
SSL-enabled version of spamc or the spamc shared library by running "make
spamd/sslspamc" or "make spamd/libsslspamc.so".  Thanks to Nate
<nate@cs.wisc.edu> for this patch.


Bugs
----

There are no known bugs with this setup.  Several reasonable sized sites
are now running it on their production mail systems.  However, you should
still test it completely in *your environment* before trusting all your
mail to it.  If you discover compilation, runtime, or load-performance
bugs, please open a ticket at http://bugzilla.spamassassin.org/

NEW: Presumably the following bug has been fixed, since I'm running OS X Server v.10.4
 with SpamAssassin and it appears to support DCC (use_dcc is 1 in local.cf).

There is/was an issue if you run spamd using the standard perl installation
on Mac OS X and certain *BSD-flavored UNIX platforms.  spamd will change
effective uid to the user calling spamd for security reasons.  Before
calling out to any external programs (DCC and Pyzor, as of 3.0.0,) spamd
will fork() and change the real uid to the same as the effective uid.
Unfortunately, the default perl in at least Mac OS X, does not allow perl
programs to change the real uid so for security reasons the spamd child
will die.  To fix this issue, either disable the DCC and Pyzor rules,
or install a different version of perl which supports setuid() calls.

The default perl binary in FreeBSD had a similar issue when attempting
to change the real uid.  This has been worked around, but there could
be an issue such as the one in Mac OS X that we have not yet heard about.
```

Comments

Please enter comments here. You can type @"SIG@ to insert your signature. – DuncanFindlay <<DateTime(2005-08-22T02:45:30Z)>>