Basic Setup

Introduction

The conversation was short, and unsuccessful.

"*It's really a simple request; each of the employees needs an additional paid half week each year.*", I pointed out.

"*For what?*", my boss asked. "*Vacation? Community service? An office retreat?*"

I shuffled uncomfortably. "*Well, no. Filtering spam out of their email.*"

It sounds ridiculous, right? It isn't. Handling spam is a major expense for corporations, even if it doesn't show up as a line item in the budget.

I get about 120 spams/day (would anyone care to bet that number will drop? Didn't think so. :−)) That's 43,800 spams a year. Let's assume I do nothing but identify and discard the spam, and can do so in 3 seconds per message − pretty fast, but I'm trying to come up with a conservative estimate − I'm spending 36.5 hours a year doing nothing but filtering spam.

sigh

I consider it possible that since my email address is visible on a lot of web sites and articles that I get more spam than most; lets assume I get twice as much as the average email address holder. That means that the average email recipient is spending a half week a year filtering spam. If that's done on work hours, that means the equivalent of 1 salary for every hundred employees is wasted on filtering spam.

This is why my blood pressure goes up when I see Alyx Sachs quoted in the New York Times online (purchase required) saying, "*These antispammers should get a life. Do their fingers hurt too much from pressing the delete key? How much time does that really take from their day?*" Enough time that my wife stopped using email because of the over 19,000 spams in her account.

In addition to the time spent handling it, we also have to account for the time spent reporting illegal spams, the steady interruptions, and the time and equipment spent on filtering it out. I'll try to minimize this latter cost by leading you through setting up a very powerful spam filter − Spamassassin − with the worlds most common mail server − Sendmail.

Overview

Your current mail setup probably acts something like this. Incoming mail arrives at port 25 of your mail server, where Sendmail is listening for incoming connections. Sendmail verifies that the mail is destined for someone with a local account. Sendmail hands the mail off to the procmail program. If you don't have a .procmailrc in your home directory on the mail server, the message is placed directly in /var/spool/mail/{your_user_id}. If you do have a .procmailrc, procmail consults the rules (such as "place all mail from mywife@hercompany.com in the family folder") in that file to decide what to do with the message.

We're going to add a few rules to .procmailrc that first run a program that scores this message on how likely it's spam, and then based on this probability, filter it into a almost−certainly−spam or probably−spam folder. Mail that doesn't hit either probability cutoff gets sent through to the original email folder.

Spamassassin

Why Spamassassin? I've used 6 different spam filtering approaches over the past 6 years (junkfilter, spambouncer, a home−grown filter, razor1, razor2, and spamassassin). All of them helped, to some degree, but none of the others have as comprehensive a list of tests as Spamassassin. Here are the tests Spamassassin can use:

Header fields
: The bulk mail software tools used by spammers have certain signatures; their message−id might have a certain form, they might always screw up some portion of writing mime headers.

Body phrase identification
: Yup, you guessed it. Body parts and what you can do with them, African banks and how you get 1/5 of the take for being someone's uncle, "This is not spam", etc. Spamassassin has an amazingly good list that does a good job filtering even if you don't want to use the other features.

Bayesian filtering
: No matter how good the header and body checks are, they will always fall behind the spammers and fail to take into account the characteristics of what you consider spam and what you consider legitimate mail (called "ham"). Bayesian filtering takes folders of known spam and known ham and identifies words or phrases ("tokens") that only show up in spam and tokens that only show up in ham. When a new message is being scored in the future, if it contains a lot of spammy tokens, it spam score goes up. If it contains a lot of hammy tokens, its spam score goes down. This is a much better approach than static phrase identification as it handles the case where "Nigeria" might be a legitimate word in an email to a travel agent's office, or "breast" would be legitimate in an email to a women's clinic.

Automatic white/blacklist
: In much the same way, Spamassassin keeps an Automatic WhiteList, or AWL, of sender email addresses. When a new message comes in, Spamassassin goes back to the AWL database and asks "What was the average spam score for messages from this email address and IP address?" If that number for previous messages was high (likely spam), Spamassassin assumes this new message will be as well and raises the spam score for the new one. Likewise, if the number was low or negative (likely ham), Spamassassin lowers the spam likelihood score for this message.

Manual white/blacklist
: And for the times when Spamassassin just doesn't know you want commercial mail from a given vendor, you can manually say "I want all mail from someone@company.com or *@company.com" and they'll arrive at your mailbox even if the spam score is very high. It's also possible to say "all mail from *@fantasy−mail.com is spam, even if the score would otherwise be borderline." See down in the

advanced topics section for more information on a prebuilt manual blacklist.

DCC, Pyzor, Razor2

Spams, by and large, get distributed to lots of people with little or no modification. The DCC, Pyzor, and Razor projects attempt to cash in on this fact by asking people to submit a message to a central database once it has been identified as spam. If I identify a message as spam at 8:45am, I'll submit it to one of these databases. When you read the same message sent to you at 9:10am, Spamassassin asks that database, "Has anyone submitted this message as spam?". The database responds, "I'm 70% sure it is because someone reported it", and now its spam likelihood goes up.

RBL (A number are consulted; see tests.)

Real−time Blackhole Lists focus on the IP addresses of the mail servers that passed the message along to you. When Spamassassin asks them about a particular mail server IP address, their reponse may drive the spam likelihood up because that mail server is run by a known spammer, is an open relay (a misconfigured mail server that unwittingly agreed to do the major work in sending thousands of spams), or is dial−up modem IP address (modem connected users generally don't send mail directly; they usually hand off the message to their ISP's mail server to send).

Character set and locales

This one's easy − I have no legitimate senders that would send me mail in the GB2312 or BIG5 character sets (Chinese, I believe). I've told spamassassin that mail in non−english character sets should be marked as spam.

Positive and negative scoring

As I've mentioned, the individual spamassassin rules can either rate a message's spam likelihood up (because the rule triggers on spam) or *down* because it triggers on ham. For example, few spammmers use the Pine mail program on Linux to send their messages; emails with a signature that they were created with Pine on Linux can have their spam likelihood lowered a bit.

Note that *none* of the above criteria, by themselves, is enough to say a message is definitely spam; I can come up with examples for any of the above where a given rule will misfire and incorrectly increase or decrease the spam score. However, when taken together, the collection is marvelously strong and accurate at identifying spam and ham.

Software Setup

There are a number of steps to take, but many of them only need to be done once by a mail server administrator.

First off, make sure that your mail server is working correctly, accepting and delivering mail. I'm assuming you're using Sendmail on an rpm−based distribution. This latter is not a problem if you're not; for debian users the install may be as simple as "apt−get {packagename}", and other non−rpm distribution users are probably comfortable installing these programs from source.

Spamassassin install

Instructions and hyperlinks for a number of distributions are at the download page. I'm going off the RPM approach, so I pull down the perl−Mail−Spamassassin, spamassassin, and spamassassin−tools i386 rpms from Theo Van Dinter's site.

Most of the commands in this section should be performed by the root user. The following packages are the ones for Redhat 7.x; if you're using another distribution, see the spamassassin rpm site.

```
cd ~
```

```
mkdir spamassassin
cd spamassassin
wget http://spamassassin.kluge.net/RPMS/perl-Mail-SpamAssassin-2.53-1.7.3.i386.rpm
wget http://spamassassin.kluge.net/RPMS/spamassassin-2.53-1.7.3.i386.rpm
wget http://spamassassin.kluge.net/RPMS/spamassassin-tools-2.53-1.7.3.i386.rpm
wget ftp://ftp.kluge.net/pub/felicity/RPMS/perl-Net-DNS-0.33-0tvd.noarch.rpm
```

Before we can install these, we need to get some perl modules. Many of these will be right on your vendor's CD; the remainder should be at Theo's supplementary RPM site.

```
#For Redhat 7.2
rsync -av zaphod.stearns.org::redhatmirror/pub/redhat/linux/7.2/en/os/i386/RedHat/RPMS/perl-HTML-
rsync -av zaphod.stearns.org::redhatmirror/pub/redhat/linux/7.2/en/os/i386/RedHat/RPMS/perl-HTML-
#For Redhat 7.3
rsync -av zaphod.stearns.org::redhatmirror/pub/redhat/linux/7.3/en/os/i386/RedHat/RPMS/perl-HTML-
rsync -av zaphod.stearns.org::redhatmirror/pub/redhat/linux/7.3/en/os/i386/RedHat/RPMS/perl-HTML-
```

Now we install the Spamassassin RPMs:

```
rpm -Uvh perl-Mail-SpamAssassin-*.i386.rpm spamassassin-*.i386.rpm perl-HTML-Parser-*.i386.rpm pe
```

On RedHat 7.2, you may need to add --nodeps if rpm complains of a missing perl(HTML::Parser); the perl-HTML-Parser obviously provides this resource but doesn't appear to correctly declare so.

To avoid the overhead of starting a fresh copy of perl each time a new mail message comes in, there's a background daemon called spamd that holds most of the spam scoring code. Let's start that up:

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

To check that it's running, try:

```
[root@slartibartfast spamassassin]# netstat -anp | grep spamd
tcp   0    0 127.0.0.1:783   0.0.0.0:*  LISTEN  4753/spamd -d -c -a
unix  2    [ ]         DGRAM            5988777 4753/spamd -d -c -a
```

This says that spamd is running under PID 4753 - your PID will differ. It's listening on a Unix socket and TCP port 783 but only for connections coming from localhost.

Note that we don't have to do anything about making it start on next boot as the RPM has done that for us. If you're not using rpms, use whatever approach is appropriate for starting a given service in your default runlevel; you may need to run tools like ntsysv or chkconfig or may need to rename a file in /etc/rc3.d or /etc/rc5.d .

Spamassasin Client Install

To demonstrate, I'll do all the following with a bogus user called "spamtest", which I'll add now as root.

```
adduser spamtest
```

The following steps will need to be taken for each person that would like their mail filtered; substitute the correct username everywhere you see spamtest. Everything from this point on is done as the user for whom we're filtering mail.

```
su - spamtest
```

```
cd ~
mkdir .spamassassin
cd .spamassassin
cp −p /usr/share/spamassassin/user_prefs.template user_prefs
cat <<EOF >>user_prefs
rewrite_subject 1
report_header 1
use_terse_report 1
defang_mime 0
report_safe 0
use_bayes 1
auto_learn 1
ok_locales en
EOF
```

All of the configuration options between cat and EOF will be added to user_prefs by the cat command. See "perldoc Mail::SpamAssassin::Conf" for more info on these settings.

Let's see if spamassassin's working before we go on:

```
spamc −R </usr/share/doc/spamassassin−*/sample−nonspam.txt
−6.3/5.0
* −6.3 −− Contains a PGP−signed message

spamc −R </usr/share/doc/spamassassin−*/sample−spam.txt
8.4/5.0
*  0.7 −− From: does not include a real name
*  0.6 −− Invalid Date: header (not RFC 2822)
*  1.4 −− Valid−looking To "undisclosed−recipients"
*  1.5 −− BODY: Information on how to work at home (2)
*  1.5 −− BODY: Drastically Reduced
*  0.8 −− BODY: List removal information
*  0.7 −− BODY: Once in a lifetime, apparently
*  0.2 −− Date: is 12 to 24 hours before Received: date
*  0.6 −− RBL: Received via a relay in relays.osirusoft.com
          [RBL check: found 142.249.10.63.relays.osirusoft.com., type: 127.0.0.3]
*  0.4 −− Message−Id is not valid, according to RFC 2822
```

When the nonspam message is fed in to spamc, spamc hands the text off to spamd which calculates the actual spam score. Because it has no spam characteristics it has no plus points, but a −6.3 because it's a PGP signed message. The final score is a −6.3, far below the needed 5.0 to categorize it as spam.

The second message has a bunch of spam characteristics. None, by themselves, are enough to categorize it as spam, but together they give it a score of 8.4.

If you don't something like this output (format and exact score may vary, that's OK) for the two test messages, you should take the time to figure out why before going on.

Procmail setup

Now that spamc is correctly identifying mail, lets set up the spamtest user to actually use it and filter mail.

```
cd ~
mkdir .procmail
touch .procmail/proclog
mkdir mail
touch mail/mbox
```

We need to create a .procmailrc file in /home/spamtest . If the user doesn't already have one, here are some suggested starting points. If the user does have one, add the lines between "#Spamasssassin start" and "#Spamassassin end" from the appropriate example to their existing file.

If the user gets their mail from this machine via IMAP or POP:

```
SHELL=/bin/sh
PATH=/bin:/usr/bin
PMDIR=$HOME/.procmail
LOGABSTRACT=all
MAILDIR=$HOME/mail       #you'd better make sure it exists
LOGFILE=$PMDIR/proclog   #recommended
VERBOSE=off

#Spamassassin start
:0fw: spamassassin.lock
| /usr/bin/spamc
#Spamassassin end
```

In the above example, procmail sends the message through spamc for scoring and changing the headers like Subject if it is a spam, but the message is allowed to pass straight through to its original destination (usually /var/spool/mail/spamtest). This one folder is the IMAP INBOX.

The spam messages can now be moved from folder to folder inside the user's mailreader; this job is made easier by the modified Subject line and the X−Spam−Status and X−Spam−Level headers − read on for more detail.

If the user reads their mail right on this machine (say, with pine) use this .procmailrc instead:

```
SHELL=/bin/sh
PATH=/bin:/usr/bin
PMDIR=$HOME/.procmail
LOGABSTRACT=all
MAILDIR=$HOME/mail       #you'd better make sure it exists
LOGFILE=$PMDIR/proclog   #recommended
VERBOSE=off
DEFAULT=$MAILDIR/mbox


#Mailing list start
#If you subscribe to any mailing lists, you might want to filter them off first:
:0:
* ^X−BeenThere: dshield@dshield.org
dshield

:0:
* ^X−BeenThere: user−mode−linux−devel@lists.sourceforge.net
uml−devel

#Mailing list end


#Spamassassin start
:0fw: spamassassin.lock
| /usr/bin/spamc

:0:
* ^X−Spam−Level: \*\*\*\*\*\*\*\*\*\*\*
```

```
spam10

:0:
* ^X-Spam-Status: Yes
spamassassin-spam

#Spamassassin end
```

A little explanation is needed. The first block sets up environment variables for use in the rest of the file.

The "mailing list" block is optional. If you're subscribed to mailing lists and want to have them sent off to different folders automatically, find a line in the header that identifies the list. "X-BeenThere:" is most commonly used, but "Errors-To:", "List-Id:", "Mailing-List:", "Reply-To:", "Sender:", and "X-Loop:" are other good ones to look for.

The spamassassin block is where we finally get some useful spam filtering. The first rule (the ":0fw: spamassassin.lock" and "| /usr/bin/spamc" lines) feed the entire message off to spamc, which hands it off to spamd for spam score calculation. spamd also adds and/or modifies headers to indicate that the message is spam; in particular, it adds a "X-Spam-Status: Yes" header for any messages with a spam score 5.0 or above, and it also adds a "X-Spam-Level: *******" with the number of asterisks equal to the integer part of the score. In other words, a spam with a score of 8.3 would get a "X-Spam-Level: ********" header.

We use the X-Spam-Level header to in effect give us *two* cutoffs; 5.0 and 10.0. Those with a score 10.0 and higher get relegated to a different folder, one that we don't need to check as often, or depending in your needs, at all. These are the ones that are almost certainly spam with very little chance of false positives. With those out of the way, we can send all the remaining spams (with scores between 5.0 and 9.9) to the spamassassin-spam folder. This one should probably be checked from time to time as some messages may be incorrectly identified as spam.

As a side note, if the message doesn't match either test, we have no more procmail rules in this file to specify what to do with it. In that case, the message is sent to the mail folder specified in the DEFAULT variable; in this case, /home/spamtest/mail/mbox .

With this file in place, send a test message to spamtest@mydomain.com. The messages should show up in /home/spamtest/mail/mbox with minimal headers saying that the message score is less than 5 and should not include the "X-Spam-Status: Yes" or "X-Spam-Level: **..." headers.

Now bounce a spam message you've received to spamtest@mydomain.com . If the spam score is high it should show up in /home/spamtest/mail/spam10 . If the score is between 5 and 9.9, it should show up in /home/spamtest/mail/spamassassin-spam .

If this didn't work, go back and find out why. That's why we do this with a test user that won't get angry if mail is lost. :-) Some files that may help in the query will be /var/log/maillog and /home/spamtest/.procmail/proclog ; these may tell you where the mail was sent and possibly even why. With the three test messages I sent, proclog shows where they went:

```
From wstearns@pobox.com  Sun Mar 23 21:55:00 2003
 Subject: quick check
   Folder: /home/spamtest/mail/mbox                                    1569
From wstearns@pobox.com  Sun Mar 23 21:56:02 2003
 Subject: *****SPAM***** wow em this summer...start now
   Folder: spamassassin-spam                                           3299
From wstearns@pobox.com  Sun Mar 23 21:58:07 2003
 Subject: *****SPAM***** Earn great money from home!           DVCMXNI
```

```
Folder: spam10                                                  10863
```

While you're investigating any problems, you may wish to temporarily restore the user's original .procmailrc or simply rename this new one to something else so that more incoming mail is not misdirected.

The result

At this point, I'm going to leave you to set up your users using this approach. Many of the checks are automatically working, including the Auto−Whitelist and Bayesian filtering, although both could be helped if you explicitly feed them known ham and spam (in fact, the bayesian filtering will hold off actively influencing the score until it has a thousand hams and spams; see the autoreporting section for more info on how to train it). Razor2 is an excellent addition, and will probably show up in a future version of this article. The RBL checks should be working as well; some of your messages should include lines like the following in their headers:

```
*  0.6 −− RBL: Received via a relay in relays.osirusoft.com
          [RBL check: found 142.249.10.63.relays.osirusoft.com., type: 127.0.0.3]
```

With spamassassin in place, your users' mail should now be automatically filtered into folders. They still get just as much, but it's not a constant interruption. By filtering into "almost certainly spam" and "probably spam" folders, you actually *have* cut down on the number of messages to which your users have to actively pay attention. I've spent quite a bit of time teaching my spamassassin installation about known spam and ham, so I feel comfortable making the second cutoff at 8.0 (spams with a score higher than this I won't look at at all, but they're still available if I later find out that something was misclassified). This means I never look at 78% of the incoming spam, changing my yearly spam week into a yearly spam day.

And I figure I can spend my 4 free days this year writing a spam filtering article for you. :−)

Advanced topics

Manual blacklist

By adding lines such as:

```
blacklist_from sde@spledee.com
blacklist_from *@bonanzaoffers.com
blacklist_from *@deal−seeker.com
blacklist_from *@hispeedmediaoffers.com
blacklist_from *@jumpjive.com
blacklist_from *@*.ew01.com
```

to /etc/mail/spamassassin/local.cf or ~/.spamassassin/user_prefs , you tell spamassassin that mail from any of these domains gets a +100 spam score, effectively blocking them. I'm compiling a list of spammer domains in a format you can cut and paste right into the spamassassin config files. The list, and a script that can be run from cron to automatically pull down and install the latest version, can be found at http://www.stearns.org/sa−blacklist/ . The current version of the list is at http://www.stearns.org/sa−blacklist/sa−blacklist.current . This list works for me, but you may wish to at least briefly look it over to see if it works for you.

Many thanks to everyone that has contributed blacklist entries.

Global settings

If you want to make any user configuration options (like the above lines) apply to all spamassassin users, place them in /etc/mail/spamassassin/local.cf .

Sitewide filtering

The obvious next question is, "Can I just get spamassassin to process *everyone's* mail without having to mess with everyone's .procmailrc?" Obviously the answer's yes, or I wouldn't have asked the question. *grin*

Please make sure you have spamassassin working correctly for at least a test user before going for the Big Kahuna. Screwing up for all your mail users tends to hurt your chances for long−term employment. Also, you may want to test this on a secondary mailserver in high volume environments to make sure the additional load won't be a problem.

With the warnings out of the way, here's what we'll do. We're going to run spamc out of the system−wide procmail configuration file, /etc/procmailrc . Requests made in here are performed for all locally delivered messages.

Set up a conservative set of configuration choices in /etc/mail/spamassassin/local.cf . In particular, it might be a good idea to raise the default cutoff for spam to 8.0, at least initially. If everything works and you're not getting any false positives in a week, lower it to 7, and then 6.5 or 6 a week later. This gives the AWL and bayes databases a chance to learn a bit before they're really crucial.

```
required_hits 8
rewrite_subject 1
report_header 1
use_terse_report 1
defang_mime 0
report_safe 0
use_bayes 1
auto_learn 1
ok_locales en
```

With those settings in place, restart spamd with:

```
/etc/init.d/spamassassin restart
```

Now tell procmail to run spamc on everyone's mail. Add these to /etc/procmailrc :

```
DROPPRIVS=yes

:0fw
| /usr/bin/spamc
```

Finally, remove the ":0fw" and "| /usr/bin/spamc" lines from everyone's individual /home/{user}/.procmailrc files. The scoring and header changes were done when the message first passed through /etc/procmailrc; the custom requests like "filter all messages with a score of 10 or higher into this folder" still need to get done locally in /home/{user}/.procmailrc .

If you have a small number of users that specifically *don't* want their spam filtered, add a line like the following for each user or domain to /etc/mail/spamassassin/local.cf :

```
all_spam_to masochist1@mydomain.org
all_spam_to *@masochists.org
```

Shared whitelist and bayes databases and autoreporting addresses

For a mail setup like the above, you may want to provide an easy way for users to report spam and ham messages, especially ones that spamassassin misidentified. What we'll do is provide one email account that accepts mail from your users that is definitely spam, and another email account that accepts verified ham; both register the sender email address and tokens in the automatic whitelist and bayes databases.

This is *not* as effective as having everyone maintain their own databases, but it requires less effort; having a few people send off their spam and ham provides some benefit for all.

First we need to start using shared awl and whitelist databases so that what we submit to the system is used by everyone's mail filters.

```
adduser sharedspam
mkdir /home/sharedspam/.spamassassin/
chmod 755 /home/sharedspam
chmod 777 /home/sharedspam/.spamassassin/  #There's probably a safer way to do this.
touch /home/sharedspam/.spamassassin/user_prefs
chmod 644 /home/sharedspam/.spamassassin/user_prefs
touch /home/sharedspam/.spamassassin/auto-whitelist
chmod 666 /home/sharedspam/.spamassassin/auto-whitelist
```

Add the following to /etc/mail/spamassassin/local.cf . The "bayes_ignore_header" lines tell the bayesian filtering code to ignore the headers your users' mail apps will add in bouncing the message.

```
bayes_path /home/sharedspam/.spamassassin/bayes
auto_whitelist_path /home/sharedspam/.spamassassin/auto-whitelist
bayes_file_mode 777
auto_whitelist_file_mode 777
use_bayes 1
auto_learn 1
bayes_ignore_header ReSent-Date
bayes_ignore_header ReSent-From
bayes_ignore_header ReSent-Message-ID
bayes_ignore_header ReSent-Subject
bayes_ignore_header ReSent-To
bayes_ignore_header Resent-Date
bayes_ignore_header Resent-From
bayes_ignore_header Resent-Message-ID
bayes_ignore_header Resent-Subject
bayes_ignore_header Resent-To
```

and restart spamd:

```
/etc/init.d/spamassassin restart
```

Now let's add the accounts to which your users will send the spam and ham:

```
adduser spamtrap
cd /home/spamtrap
ln -sf ../sharedspam/.spamassassin .spamassassin
mkdir mail .procmail
chown spamtrap.spamtrap mail .procmail
```

```
adduser hamtrap
cd /home/hamtrap
ln −sf ../sharedspam/.spamassassin .spamassassin
mkdir mail .procmail
chown hamtrap.hamtrap mail .procmail
```

Here's what we put in /home/**spam**trap/.procmailrc :

```
SHELL=/bin/sh
PATH=/bin:/usr/bin
PMDIR=$HOME/.procmail
LOGABSTRACT=all
MAILDIR=$HOME/mail      #you'd better make sure it exists
LOGFILE=$PMDIR/proclog   #recommended
VERBOSE=off

#Spamassassin start

:0
* > 256000
toobigtoreport

:0c: spamassassin.lock
| /usr/bin/spamassassin −r −d −a

:0:
learned−spam

#Spamassassin end
```

And here's what we put in /home/**ham**trap/.procmailrc :

```
SHELL=/bin/sh
PATH=/bin:/usr/bin
PMDIR=$HOME/.procmail
LOGABSTRACT=all
MAILDIR=$HOME/mail      #you'd better make sure it exists
LOGFILE=$PMDIR/proclog   #recommended
VERBOSE=off

#Spamassassin start

:0
* > 256000
toobigtoreport

:0c: spamassassin.lock
| /usr/bin/sa−learn −−ham −−single −−local

#:0:
#learned−ham

#Spamassassin end
```

There are two major differences in the above files:

| /usr/bin/spamassassin −r −d −a
versus
| /usr/bin/sa−learn −−ham −−single −−local...

When one of our users submits spams, the `spamassassin -r` command will do all we'd like it to: submit the signatures of the messages to the online Razor, Pyzor, and DCC databases if we have those configured, and update both the local AWL and Bayesian databases. However, when a user submits ham, I personally don't want any chance that the body of that message will leave my network. For that reason, I suggest using `sa-learn --local`, which will only update local databases.

learned-spam

versus

#learned-ham

Once we've finished submitting the messages to local and remote databases, I'd like to save a copy of the spams; at some point in the future, we may want to submit the entire folder verbatim to Spamarchive as a resource for the spam filter developers. I'm really concerned about keeping any legitimate mail, however. Users might, in the interest of being helpful, sumbit messages that really should never leave the building. Having our company financials show up on an ftp server is a bad way to start a monday.

Once this is set up, ask some of your users to bounce (not forward, bounce; see the next section) some messages they are *sure* are spam to spamtrap@*mydomain.com* and some messages they are *sure* are ham to hamtrap@*mydomain.com*. The source email addresses will be registered in the auto-whitelist database as known spammers or legitimate senders, depending on to which address the mail was sent. Likewise, the tokens in the mail will be registered as spammy or hammy, again, depending on which trap was used.

Try to get at least 1000 spams and 1000 hams into the system. Get messages of different types from different users. Encourage your users to submit, at the very least, messages that spamassassin has misclassified.

Note that if they *forward* messages to spamtrap@, sa-learn will remember *the user's* email address - not the original spammer's - in the blacklist, so remind them to use the bounce feature.

As one last bit of harmless fun, try putting the spamtrap@*mydomain.com* address up on your web site, with warnings around it like "Please do not send any mail to this address; it's part of an email harvesting study.". Automated tools that search web pages for email addresses may find the spamtrap address, add it to their list of recipients, and start sending spam to it - instantly reporting themselves to the spam databases.

Restricting who can report

One concern is that someone could poison our AWL and Bayes databases if they sent spams to the hamtrap or hams to the spamtrap. Lets put in some simple access controls to only allow certain people to train Spamassassin.

Here's the new version of /home/**ham**trap/.procmailrc :

```
SHELL=/bin/sh
PATH=/bin:/usr/bin
PMDIR=$HOME/.procmail
LOGABSTRACT=all
MAILDIR=$HOME/mail      #you'd better make sure it exists
LOGFILE=$PMDIR/proclog   #recommended
VERBOSE=off

#Spamassassin start

:0
* > 256000
toobigtoreport
```

```
:0c: spamassassin.lock
* ^ReSent−From: William Stearns <wstearns@pobox.com>
* ^ReSent−Message−ID: <Mutt.*@homepc.stearns.org>
| /usr/bin/sa−learn −−ham −−single −−local
#:0:
#* ^ReSent−From: William Stearns <wstearns@pobox.com>
#* ^ReSent−Message−ID: <Mutt.*@homepc.stearns.org>
#learned−ham

:0c: spamassassin.lock
* ^ReSent−From: Fred Parker <fparker@spleen.com>
* ^ReSent−Message−ID: <Mutt.*@devel.spleen.com>
| /usr/bin/sa−learn −−ham −−single −−local
#:0:
#* ^ReSent−From: Fred Parker <fparker@spleen.com>
#* ^ReSent−Message−ID: <Mutt.*@devel.spleen.com>
#learned−ham

:0:
unauthorized

#Spamassassin end
```

We've added "ReSent" lines to the sa−learn and learned−ham stanzas, added a second copy of these to allow Fred to submit as well, and added a new unauthorized stanza. When I send in mail, procmail sees that it came from my email address and has a message ID my mail program would legitimately have added, so this message is learned as ham. Likewise, mail from Fred gets learned as ham.

When someone other than Fred or I tries to send mail to this address, procmail simply files it in the "unauthorized" folder. You should check this folder every once in a while to see if someone is − legitimately or not − trying to teach your Spamassassin install.

The easiest way to figure out what headers to use (the "ReSent" lines, above), simply start off with no stanzas at all except for the ":0: unauthorized" one. Bounce a message into the system and look at the headers of the message. Now create the sa−learn and learned−ham stanzas for this user, pasting in the ReSent−From header as is. You'll need to modify the ReSent−Message−ID line as this is a unique string for every new message. Replace the ID portion of:

```
ReSent−Message−ID: <Mutt.0304211722060.327@homepc.stearns.org>
```

with ".*", ending up with:

```
ReSent−Message−ID: <Mutt.*@homepc.stearns.org>
```

Don't forget to modify the .procmailrc for both the hamtrap and spamtrap.

This is *not* perfect security; someone could forge your headers and sneak messages into your traps. However, this is sufficient to stop dictionary attacks. Also, if you're trying the "email harvesting study" fun in the last section, you can't use the above filtering on the spamtrap address.

How to bounce/redirect mail

As I've mentioned, *forwarding* mail into a spamtrap registers *your* email address as being a spam source; not good. Instead, we want to use the **redirect** or **bounce** feature available in a number of mail programs.

Many thanks to all the contributors, listed in parentheses.

AOL's integrated email client
: Redirecting mail is not available. *(Dave Goldsmith)*

Eudora
: Select the message, go to the "Message" menu, choose redirect, fill in the address, and choose send. *(Brian Corcoran and Erik Wheeler)*

Evolution
: Select the message. In the "Actions" menu, choose the "Forward" submenu (not "Forward message", the "Forward" submenu). Pick "Redirect", fill in the "To" field, and press "Send". *(Johannes Ullrich)*

OS/X Mail.app
: With the email message open or selected, go to Mail's 'Message' menu and select 'Bounce to sender' or 'Redirect'. If you use this frequently, go to the "View" menu, choose "Customize toolbar", and add a button for "Redirect". *(Marion Bates)*

Microsoft Outlook 97
: Double−click on the message so it opens in a new window. Click on Tools−>Resend This Message. A warning will appear about you not being the original sender of the message. Click Yes. A message window appears. Update the To: field and click on 'Send'. *(Dave Goldsmith)*

Microsoft Outlook 2000
: Double−click on the message so it opens in a new window. Click on Actions−>Resend This Message. A warning will appear about you not being the original sender of the message. Click Yes. A message window appears. Update the To: field and click on 'Send'. *(Dave Goldsmith)*

Microsoft Outlook Express
: It does not appear to have a redirect option. *(Dave Goldsmith and Alex Bates)*

Netscape Communicator 4.x and 7.x
: They don't appear to have a redirect option.

Pine
: For a single message, highlight the message and press "b" to bounce it. Enter the target address and press enter.

  For multiple messages, select all the messages you'd like to bounce with either ":" to select them one at a time, or ";" to select multiple messages by message number, subject, body text, etc. Once selected, press "a", then "b" to Apply the Bounce command to all of them. Enter the target email address. Once done, press ";", then "a" to Unselect All selected messages.

  More info can be found at: http://www.itc.virginia.edu/desktop/email/pine/bounce.html

Sylpheed
: Click on the message, go to the "Message" menu, choose "Redirect", fill in the "To:" address, and press send. Alternately, right click in the message and choose "Redirect" from the popup menu, fill in the "To:" address, and press send. *(Dave Goldsmith)*

Other places to report spam

Once you've identified a spam, you may wish to take the time to report it to agencies that have the authority to investigate. Here are some suggestions of where to bounce those messages.

Nigerian scams
: The 419 coalition: 419.fcd@usss.treas.gov

Stock scams and investment advice
: Securities and Exchange Commission: enforcement@sec.gov

Spamarchive
: Spamarchive: Forward as an attachment to submit@spamarchive.org

Illegal sales of drugs, medical devices, biological products, foods, dietary supplements, or cosmetics.
: Food and Drug Administration (FDA): webcomplaints@ora.fda.gov

Chain letters
: USPS, fraud@uspis.gov, and FTC. More info at: http://www.interleaves.org/~rteeter/pyramid.html.

Ads for Systemworks and other Symantec products
: Symantec spamwatch: spamwatch@symantec.com

Child pornography
: US department of Justice, National Center for Missing and Exploited Children, International reporting sites, and Cyberangels international reporting links.

General spam
: Federal Trade Commission (FTC): uce@ftc.gov

Also...
: Report persistent spams to the ISP hosting the spammer. "abuse@*the_isp_domain*" is a good address to try if you can't find a reporting address on their web site.

Existing mail in an IMAP folder

(Many thanks to Marion Bates for contributing this section.)

The above approach works fine for *new* mail, but what about mail that a user has already received?

If that mail is in an imap folder, you're in luck. Roger Binns has written IMAP Spam Begone to reprocess mail that is already in an IMAP folder.

There are a couple of assumptions:

- You already have spamassassin installed and configured on whatever machine this script is running on.
- Your main imap inbox is /var/spool/mail/username
- You store custom imap mail folders in ~/mail/ and your mail client is configured thusly. For Apple's Mail.app program, go to Preferences−>Account Information and select the Advanced tab, and enter "~/mail/" under "IMAP Path Prefix". For Netscape, Check under Edit, Preferences, Mail &Newsgroups, Mail servers, select the appropriate Incoming Mail Sever, Click Edit..., Select Advanced, and enter "~/mail" in the "IMAP server directory" box. You'll also want to uncheck "Show only subscribed folders". Restart Netscape for this to take effect.

1. Get isbg from http://www.rogerbinns.com/isbg/isbg.py .
2. chmod 755 isbg.py
3. Make a mail folder to hold all your suspected spam (in this example, ~/mail/ilovespam)

4. Run isbg.py like so:

```
./isbg.py --imaphost your-mailserver.com --imapinbox /path/to/your/inbox --spaminbox /path
```

Example:

```
./isbg.py --imaphost mail.goober.com --imapinbox /var/spool/mail/joeblow --spaminbox /home/joeblo
```

It will prompt you for your imap password, then it will go through the inbox you specified (which may take awhile), and then report what it found -- something like:

```
4 spams found in 6 messages
```

The --delete command marks the messages for deletion from your inbox. The --expunge option used in conjunction with --delete will cause the marked messages to be actually removed from inbox (they will still be in ilovespam though).

isbg is smart, and will only look through messages it hasn't seen before (i.e., it won't go through your ENTIRE inbox each time -- only new messages will be scanned.) It does this via imap's use of unique message ids.

If you run isbg with the --savepw option the first time, it will remember your imap password (saved on disk in an obfuscated way) such that you can then make a cron job to run the script automatically.

Because the isbg script and spamassassin can run on a machine other than the mailserver, this approach can be used to filter mail on a remote IMAP mailserver that may not be able to run Spamassassin directly.

Additional resources

- The aracnet procmail howto
- The Procmail home site, with lots of links to howtos
- Spamassassin home site.

Credits

The Spamassassin team - and that includes anyone that has contributed to it - gets 2 thumbs up from me for an *excellent* tool! This is the first spam filtering tool I've used that I feel confident is doing an accurate job of identifying and filtering the spam onslaught.

Bill Stearns wrote the main text of the article. Marion Bates contributed the section on ISBG. Both Marion and Drew Como were kind enough to review an early draft of this article. Dave Goldsmith, Brian Corcoran, Erik Wheeler, Johannes Ullrich, Marion Bates, and Alex Bates helped with the "How to redirect mail" section. Matt, sargon, Steve, Alan, Lee, and Ross submitted email domains for the manual blacklist project.

The spammers were kind enough to contribute the spam.  .

William is an Open-Source developer, enthusiast, and advocate from New Hampshire, USA. His day job at SANS pays him to work on network security and Linux projects.