ProcmailToForwardMail

How do I use SpamAssassin with procmail to forward mail and to do mistake-based Bayes training?

This procmail script is designed for people who have their mail forwarded through a server (e.g, example.com) but then read their mail on a non-publicized account on a different server (e.g., privateaddress@example.net). This is quite common for folks who have a vanity domain name but then read their mail through an office Exchange server, home DSL email account, etc. The idea is for procmail on the first server to run each message through SpamAssassin, and then forward the message on to the private address.

The trick for Bayes training is to add some extra procmail rules to specify special processing for false negatives (i.e., spam that SpamAssassin didn't catch). This script uses mistake-based training for false negatives. That is, it assumes that SpamAssassin can correctly autolearn on enough ham and spam to seed the Bayes database. Then, when SpamAssassin incorrectly marks a spam message as not spam, the user can train the database by redirecting the message to be learned as spam. Although a similar redirection scheme could be used to train on false positives (i.e., legimate mail incorrectly seen as spam), it's likely more effective to just ManualWhitelist mail from that legitimate sender.

The following is based on having at least two addresses (publicaddress@example.com and spam@example.com) trigger the same procmail script. In most vanity domain setups, all addresses are processed by the same procmail script. The script needs to be edited to include your real addresses and domain.

```
#Uncomment the following lines and use tail -f procmail.log to debug
#LOGFILE=$HOME/procmail.log
#VERBOSE=yes
#LOGABSTRACT=all

# Feed redirected spam to sa-learn, and also store a copy in a folder called spam.
# This folder of false negatives could be useful if we needed to rebuild our Bayes
# database in the future.

:0
* ^To:.*spam@example.com

   {
   * < 256000
   :0c: spamassassin.spamlock
   | sa-learn --spam

   :0: spamassassin.filelock
   spam
   }

# Send all other mail through SpamAssassin

:0fw: spamassassin.lock
* < 256000
| spamassassin


# Mail that is very likely spam (>15) can be saved on the server
# (not forwarded), or by moving the # down one line, even dropped
# on the floor.  Note that dropping mail on the floor is a *bad*
# idea unless you really, really believe no false positives will
# have a score greater than 15.  If you want all mail forwarded,
# just add #'s in front of each of these lines:

:0: spamassassin.filelock2
* ^X-Spam-Level: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
#/dev/null
almost-certainly-spam


# Forward all mail with a score less than 15 to my non-publicized address
:0
! privateaddress@example.net
```

This file is available procmailrc.forward.txt. If you don't currently have a procmail file, you can import this one by entering:

```
wget http://wiki.apache.org/spamassassin-data/attachments/ProcmailToForwardMail/attachments/procmailrc.forward.
txt
mv procmailrc.forward.txt .procmailrc
```

On your mail client, you'd then likely want to filter mail with a score of 5 or higher (i.e., where "X-Spam-Level: *****") into a Likely Spam folder. False Positives rarely score higher than 15. The advantage of leaving mail with a score of 15 or higher on the server is that it makes it easier to find false positives in the Likely Spam folder without being overwhelmed by hundreds of obvious spam. You can then ManualWhitelist those false positives.

For the mistake-based training, it's critical to redirect (or bounce) the message, rather than forwarding. Forwarding loses all of the critical header information, which is much of what Bayes trains from. See ResendingMailWithHeaders for details of how to do this.

Step-by-step instructions

Way more detail on how to do this is at SingleUserUnixInstall.

Other training options

An even easier form of mistake-based training is to use IMAP and create a Learn{{`As}}`Spam folder, as described in the IMAP section of SingleUserUnixInstall.

Contributors

- DanKohn