Spam Server Tuning Documentation

Cooper Stevenson
"Super Genius"*

September 29, 2010

1 GPL Notice

2 Real-Time Blacklist Configuration

I added the following entries to the SMTP deamon Postfix. These entries are made in `/etc/postfix/main.cf`:

smtpd_client_restrictions = reject_maps_rbl, reject_non_fqdn_sender

reject_rbl_client = rbl-plus.mail-abuse.org,
opm.blitzed.org,
dnsbl.njabl.org,
dynablock.njabl.org,
dul.dnsbl.sorbs.net

The first `smtpd_client_restrictions` line tells the server to reject emails delivered from domains on one of several real-time black lists listed under the `reject_rbl_client` variable.

*(541) 971–0366/cooper@cooper.stevenson.name

The `reject_non_fqdn_sender` variable instructs the server to ignore emails sent from "fake" domains that do not have a fully qualified domain name.

I further added the following recipient restrictions to `/etc/postfix/main.cf`. These are designed to help prevent message relaying from unauthorized email servers:

> smtpd_recipient_restrictions = permit_mynetworks, reject_unknown_sender_domain, reject_unauth_destination

I note the following in the log files indicates that these changes seem to be working and that the mail is getting delivered correctly.

3 Bayesian Filter Preparation

The Bayesian filter requires at least 200 messages to become trained properly. To do this, I modified the `/etc/promailrc` file to create a copy of all **almost-certainly-spam** to a mail spool for later learning by the Bayesian filter.

I modified the `/etc/procmailrc` file to send a copy of the spam message to `/var/spool/mail/filtered_spam` and to forward it on to **filteredspam@ci.[client].or.us**:

```
:0fw
* < 256000
| spamc

# Mails with a score of 9 or higher are almost certainly spam (with 0.05%
# false positives according to rules/STATISTICS.txt). Let's put them in a
# different mbox. (This one is optional.)
:0
* ^X-Spam-Level: \*\*\*\*\*\*\*\*\*
{
  :0c:    #Preserve a copy of the email
    /var/spool/mail/almost-certainly-spam
  # Forward the email
  :0
     !filteredspam@cvomailin.ci.[client].or.us
}
```

I further modified `/etc/procmailrc` to preserve a copy of all unflagged emails to `/var/spool/mail/sent_good_email`

```
# Forward the rest to "ci.[client].or.us" (assumes mx lookup to internal network wo
     :0
{
  :0c: #Preserve a copy of the mail
    /var/spool/mail/sent_good_email
  # Forward the email
  :0
```

```
    ! "$ENV_TO"@cvomailin.ci.[client].or.us
}
```

I emerged the mail client `mutt` to give me an easy way to read file-based mail stores such as /var/spool/mail/almost-certainly-spam:

emerge mutt

There was a period of time when I was only putting spam in a file on the Linux host and not forwarding it on to `filteredspam@ci.[client].or.us`. I used teh following command to later get those emails forwarded:

```
formail -ds < almost-certainly-spam /usr/lib/sendmail -v \
filteredspam@ci.[client].or.us
```

3.1 Teaching The Filter

The documentation for the Bayesian filter entitled, "Effective Training" (`http://www.spamassassin.org/d`
`training`) notes the following:

> Learning filters require training to be effective. If you don't train them, they won't work. In addition, you need to train them with new messages regularly to keep them up-to-date, or their data will become stale and impact accuracy.
>
> You need to train with both spam and ham mails. One type of mail alone will not have any effect.
>
> If the messages you are learning from have already been filtered through SpamAssassin, the learner will compensate for this. In effect, it learns what each message would look like if you had run spamassassin -d over it in advance.
>
> Another thing to be aware of, is that typically you should aim to train with at least 1000 messages of spam, and 1000 ham messages, if possible. More is better, but anything over about 5000 messages does not improve accuracy significantly in our tests.
>
> **Unsupervised learning from SpamAssassin rules**
>
> Also called 'auto-learning' in SpamAssassin. Based on statistical analysis of the SpamAssassin success rates, we can automatically train the Bayesian database with a certain degree of confidence that our training data is accurate.
>
> It should be supplemented with some supervised training in addition, if possible.
>
> This is the default, but can be turned off by setting the SpamAssassin configuration parameter bayes_auto_learn to 0.

Late in the afternoon I browsed through the good email folder to check for messages that were actually spam. There is more work to be done in this area but I will be finished well before Tuesday evening.

Since I had enough emails for the "good" email store, I disabled the copying feature described above in `/etc/procmailrc`. Total number of good emails: 1023.

4 Spam Extraction Script

When I began this project, I thought that I would remove the spam from the reported email messages via Procmail scripts. As I looked at the reported mail store file, however, it started to become apparent that a script would be more appropriate. The mbox format makes it possible to parse a large file of mail into just the original spam.

I first made a backup of the "reported good" and "reported-spam" message folders on the servers.

```
cp -P ./reported_spam ./reported_spam.0
cp -P ./reported_good ./reported_good.0
```

I then wrote the following script to extract the originally attached spam messages from the email. It doesn't matter if the user attaches one or a thousand spam messages per message, the script will extract them. The script is currently `/root/bin/extract.sh`

```
#!/bin/ksh
############################################################
#Cooper Stevenson       23-Feb-03       extract_spam.sh#
#                                                         #
#This script processes an mbox file and extracts the      #
#actual spam reported by the users, leaving the "good" #
#header information behind.                               #
############################################################

process_file=/var/spool/mail/reported_spam
border_string='Content-Type: message/rfc822'
tmp_output_file=/var/spool/mail/processed_reported_spam.tmp
output_file=/var/spool/mail/reported_spam
sed_cleanup_script=/root/bin/email_strip.sed
print_line=0
border_check=0

touch $tmp_output_file #create the file for later output

cat $process_file | while read line
do
```

```
  if [[ $line = $border_string ]] # Are we at the start of the spam?
  then
 printline=1 # Turn the "print line" switch on
  fi
  if [[ $printline = 1 ]] # If we're supposed to print the line...
  then
   {
     # Known Good line!
     # print -- $line | grep -e ------_=_NextPart -
     if [[  $(print -- $line | grep -e ------_=_NextPart -) ]] #End check
     then
         printline=0
     fi
   }
   if
     [[ $line != "" ]] #take the spaces out for better mbox format
   then
     print -- $line >> $tmp_output_file #if all is well, print it!
   fi
   if [[ $line == "Content-Transfer-Encoding: 7bit" ]]
   then
     print "\n" >> $tmp_output_file
   fi
  fi
done
#After we're finished processing the file, we have to "tidy up" so that the resultar

sed -f $sed_cleanup_script $tmp_output_file > $output_file
```

Note that I call a sed script called, "email_strip.sed" The following is the short sed script that strips out the unecessary header lines after processing:

```
/^Content-Type: message\/rfc822/d
/^Content-Transfer-Encoding: 7bit/d
/^X-MimeOLE: Produced By Microsoft Exchange/d
/^------_=_NextPart/d
/^Return-Path:/d
/^MIME-Version: 1.0/d
/^Content-Type:/d
/^charset="iso-8859-1"/d
/^Content-Transfer-Encoding:/d
/^X-Virus-Scanned:/d
/^Old-Return-Path:/d
/^Content-Class:/d
/^X-MS-Has-Attach:/d
/^X-MS-TNEF-Correlator:/d
```

After the reported spam is extracted from the original message, it's time to teach the Bayesian filter both the good and spam email messages. The documentation calls for doing this at a minimum of 1,000 messages each. My sense is that one should have a balance of good and spam emails for the Bayesian training to be most effective.

What follows is the `/root/bin/process_spam_reports.sh` script:

```
reported_good_file=/var/spool/mail/reported_good
reported_spam_file=/var/spool/mail/reported_spam
tmp_reported_good_file=/var/spool/mail/reported_good.tmp
tmp_reported_spam_file=/var/spool/mail/reported_spam.tmp
strip_script=/root/bin/email_strip.sed
formail=/usr/bin/formail

if [[ -f $reported_good_file ]]
then
    $formail -ds < $reported_good_file >> $tmp_reported_good_file
    sa-learn --ham --mbox $tmp_reported_good_file
    rm $reported_good_file
    rm $tmp_reported_good_file
fi

if [[ -f $reported_spam_file ]]
then
    $formail -ds < $reported_spam_file >> $tmp_reported_spam_file
    sa-learn --spam --mbox $tmp_reported_spam_file
    rm $reported_spam_file
    rm $tmp_reported_spam_file
fi
```