Filtering malware and spam with Postfix

http://advosys.ca/papers/postfix−filtering.html
Jul 08 2002

Introduction

Postfix (http://www.postfix.org/) is the popular Sendmail replacement that is fast, secure, and easy to administer.

Besides being a drop−in replacement for Sendmail, Postfix is widely used for secure mail gateways that act as "e−mail firewalls", relays protecting fragile Intranet mailers such as MS Exchange and Lotus Notes.

The built−in capabilities of Postfix can be enhanced by passing mail to external programs, similar to what Sendmail can now do with its experimental "milter" extension.

This paper demonstrates how to extend Postfix to perform high−level content filtering including:

- Dropping hostile attachments
- Preventing MS Outlook from auto−executing attachments
- Preventing MIME and HTML exploits
- Removing "web bugs"
- Identifying spam using a content−based filter

The method shown here can be used for locally delivered mail, and also be used when Postfix is configured as a "mail firewall," relaying mail for other systems.

The software

In addition to Postfix itself, we extend it's capabilities using two best−of−breed open source programs that do content filtering and spam identification:

Anomy Sanitizer

Anomy Sanitizer (http://mailtools.anomy.net/) is not a virus scanner (though a virus scanner can be used with it). Instead, it removes mail security problems that e−mail virus scanners do not, such as MIME boundary overflows and HTML exploits.

Anomy Sanitizer is written in Perl and filters SMTP messages checking for common exploits and hostile file attachments. It can remove attachments, rename unknown file types, "defang" HTML messages, fix MIME headers and other essential mail security tasks.

SpamAssassin

SpamAssassin (http://spamassassin.org/) is a mail filter that attempts to identify junk e−mail ("spam"). It examines the content of mail messages looking for key phrases and other identifiers common to most spam.

The approach is more sophisticated than the simple keyword matching used by most SMTP anti−virus software. SpamAssassin uses a scoring system: messages are tagged only when they have enough spam characteristics in total. This in combination with other features results in very few false positives. In our experience, SpamAssassin correctly identifies 90% to 95% of spam with less than 1% false positives.

SpamAssassin doesn't block spam; it only changes the Subject line and message headers to identify the message. Users then can create rules in their e−mail software to delete them or move an identified message to a special folder. This ensures incorrectly labeled mail is not lost.

Tagging messages also gets around certain legal issues organizations like ISPs and governments face when filtering mail. Letting each user decide what to do with spam, instead of deleting it at the gateway as corporate policy, is less open to challenge in some jurisdictions.

Preparing the server

Creating filter account

To use external filtering with Postfix, create a Unix group on the server named "filter".

Next, create a user account named "filter" on the server and make it a member of group "filter". This will be a least−privileged account used by the scripts.

No other user should belong to group "filter". Logins for the "filter" account should be locked (eg. 'passwd −l filter' on Linux and Solaris) and the shell in /etc/passwd should be set to an invalid shell such as /bin/false.

The account must have a valid home directory and it must be writable by user "filter" for SpamAssassin to function. The home directory can be located in the system's normal user directory (eg. /home on Linux, /export/home on Solaris) or you can use /var/spool/filter as the home directory.

Installing required Perl modules

Perl 5.005 or higher must be available on the server. Also, download and install the following Perl modules from CPAN (http://search.cpan.org/):

- MIME::Base64
- MIME::QuotedPrint
- Mail::Audit (and all prerequisite modules needed by Mail::Audit)

If you've set up the CPAN module in your Perl installation, the easiest way to add these modules is to use CPAN to download, build and install automatically:

```
perl −MCPAN −e shell
o conf prerequisites_policy ask
install MIME::Base64
install MIME::QuotedPrint
install Mail::Audit
quit
```

Installing Anomy Sanitizer

Anomy Sanitizer is installed simply by unpacking the tar file into a suitable directory on your mail server. We use /usr/local/anomy but any directory will work. Read the instructions in file sanitizer.html included with Anomy.

In particular, make sure the prerequisite Perl modules listed above are installed.

Change ownership of the entire anomy directory tree to owner "root", group "filter". For example:

```
        chown −R root:filter /usr/local/anomy
```

Change permissions on the anomy directory tree so it is not world readable. For example:

```
        chmod 0750 /usr/local/anomy
```

Anomy configuration

In directory /usr/local/anomy, create a file named anomy.conf and put in your anomy configuration rules. What settings you choose depends on your particular e−mail policies and the type of e−mail software you are protecting. The "real world configuration" shown in the Anomy documentation is a good starting point.

Below is a configuration we've used on mail gateways that protect MS Exchange and MS Outlook users. It defangs HTML and MIME exploits, plus drops all executable attachments (we strongly recommend dropping executable attachments as a general policy: see our paper "E−mail policies that prevent viruses" at http://advosys.ca/papers/mail−policies.html).

```
   # Example configuration file for Anomy Sanitizer
   #
   # From http://advosys.ca/papers/postfix−filtering.html
   #
   # Advosys Consulting Inc., Ottawa
   #

   # Do not log to STDERR:
   feat_log_stderr = 0

   # Don't insert log in the message itself:
   feat_log_inline = 0

   # Advertisement to insert in each mail header:
   header_info = X−Sanitizer: Advosys mail filter
   header_url = 0
   header_rev = 0

   # Enable filename based policy decisions:
   feat_files = 1

   # Protect against buffer overflows and null values:
   feat_lengths = 1

   # Replace MIME boundaries with our own:
   feat_boundaries = 1

   # Fix invalid and ambiguous MIME boundaries, if possible:
   feat_fixmime = 1

   # Trust signed and/or encrypted messages:
   feat_trust_pgp = 1
   msg_pgp_warning = WARNING: Unsanitized content follows.\n

   # Defang shell scripts:
   feat_scripts = 0

   # Defang active HTML:
   feat_html = 1
```

```
# Defang UUEncoded files:
feat_uuencoded = 0

# Sanitize forwarded content too:
feat_forwards = 1

# Testing? Set to 1 for testing, 0 for production:
feat_testing = 0

# # Warn user about unscanned parts, etc.
feat_verbose = 1

# Force all parts (except text/html parts) to
# have file names.
feat_force_name = 1

# Disable web bugs:
feat_webbugs = 1

# Disable "score" based mail discarding:
score_panic = 0
score_bad = 0

msg_file_drop  = \n*****\n
msg_file_drop += NOTE: An attachment named %FILENAME was deleted from
msg_file_drop += this message because was a windows executable.
msg_file_drop += Contact the system administrator for more information.

##
## File attachment name mangling rules:
##

file_name_tpl       = /var/quarantine/att-$F-$T.$$

# Number of rulesets we are defining:
file_list_rules = 2
file_default_policy = defang

# Delete probably nasty attachments:
file_list_1 = (?i)(winmail.dat)|
file_list_1 += (\.(vb[se]|exe|com|cab|dll|ocx|msi|cmd|bat|pif|lnk|hlp|ms[ip]|reg|asd))$
file_list_1_policy = drop
file_list_1_scanner = 0

# Allow known "safe" file types and those that can be
# scanned by the downstream virus scanner:
file_list_2 = (?i)\.(doc|dot|rtf|xls|ppt|xlw|jpg|gif|png|tiff?|txt|zip|tgz|gz)
file_list_2_policy = accept
file_list_2_scanner = 0

# Any attachment not listed above gets renamed.
```

Installing SpamAssassin

The SpamAssassin documentation describes how to install and configure that software. As described above, if you have the CPAN module set up on your server, it can download and install SpamAssassin automatically. For example:

```
perl -MCPAN -e shell
o conf prerequisites_policy ask
install Mail::SpamAssassin
```

```
        quit
```

SpamAssassin configuration

The default rules installed with SpamAssassin accurately identify most spam so changes shouldn't be needed at first. Once things are working you can fine tune SpamAssassin as required.

However, one change you may want is to add "whitelist well−known senders so their mail will never be identified as spam. It's a good idea to whitelist important clients and well known spam−free sources. Edit the file /etc/mail/spamassassin/local.cf and add whitelist_from statements similar to the following:

```
        whitelist_from   director_8345@hotmail.com   # whitelist a specific sender
        whitelist_from   @advosys.ca                 # whitelist an entire domain
        whitelist_from   @securityfocus.com
```

Configuring filtering in Postfix

We assume you have a basic working installation of Postfix. The Postfix web site and documentation included with the distribution file describe how to compile and install the software (Hint: If you're doing a first time installation of Postfix, save yourself hours of headache: get Postfix working first, add filtering later).

In the Postfix distribution tar file, the file `/README_FILES/FILTER_README` in the Postfix sourcecode directory describes ways to add external content filtering to Postfix.

Here we use the "Simple content filtering" method: plug a shell script into Postfix by modifying the `master.cf` file.

Creating the filter script

The following is a variation of the filter script provided with Postfix. It simply pipes each message sent to it through Anomy Sanitizer, then through SpamAssassin. Postfix then picks up the result and re−injects it for final delivery.

```
/var/spool/filter
r/lib/sendmail
ocal/anomy
usr/local/anomy/anomy.conf
=/usr/bin/spamassassin



 from <sysexits.h>
75
LE=69

DIR || { echo $INSPECT_DIR does not exist; exit $EX_TEMPFAIL; }

hen done or when aborting.
in.$$; rm −f out.$$" 0 1 2 3 15

/bin/sanitizer.pl $ANOMY_CONF 2>>/tmp/anomy.log | $SPAMASSASSIN −P > out.$$ || { echo Message content rejected; ex

@" < out.$$
```

Place the filter.sh script into the directory you installed Anomy Sanitizer (eg. /usr/local/anomy). The script should have the following permissions:

```
        -rwxr-x---    owner=root  group=filter
```

Create a temporary directory

The script needs a directory to store temporary files. Create a directory that is writable by group "filter". For example:

```
        mkdir /var/spool/filter
        chown filter:filter /var/spool/filter
        chmod 0750 /var/spool/filter
```

The directory should be located on a partition large enough to store a few incoming mail messages. The /var paritition on most systems is the best choice. Whatever directory you choose, change the INSPECT_DIR setting in filter.sh to match.

Changing master.cf

Finally, we need to tell Postfix to send all mail arriving via SMTP through the filter.sh script.

Add the following line to the *bottom* of /etc/postfix/master.cf:

```
filter    unix  -   n   n   -   -   pipe
    user=filter argv=/usr/local/anomy/filter.sh -f ${sender} -- ${recipient}
```

(The above must all be on a single line.)

Next, change the line in master.cf that controls the Postfix smtp daemon as follows:

```
        smtp   inet  n   -   n   -   -   smtpd -o content_filter=filter:
```

(Again, the above must all be on a single line.)

Save the changes to master.cf and use 'postfix reload' to force it to re-read it's configuration files.

Testing it out

Postfix should now be filtering every message it receives through both Anomy Santizer and Spamassassin.

Test it by sending a few messages with MIME attachments through Postfix. The Postfix log file should now show each the message being received *twice*: once by the SMTP daemon with "relay=filter", then again by the command-line "sendmail" compatibility program.

Separating inbound mail from outbound

The configuration described here sends both inbound and outbound mail through the filters. This is usually not desirable: most organizations only want to filter spam and malware from inbound mail and leave outbound mail untouched.

Unfortunately, Postfix has no inherent ability to distinguish inbound and outbound mail. There a few convoluted ways to trick it into handling inbound mail separately from outbound, but they are complex and may not work in newer versions of the software.

The simplest and most reliable way we've found to separate traffic is simple run two instances of Postfix: one for inbound mail, another for outbound. Each instance runs on the same server, but uses the Postfix `inet_interfaces` setting to receive mail on separate IP addresses: one IP for inbound mail, another for outbound.

Using two instances is simple and reliable: each instance have independent configuration files and spool directories. The inbound mail instance is configured as described above to filter via Anomy Sanitizer and SpamAssassin. The outbound instance is configured as a more generic Postfix installation, with no external filtering. We've used the two−instance method in large installations with great success. Fortunately, Postfix is reasonably light weight: running multiple instances on one server doesn't consume huge amounts of memory or disk resources.

Creating two instances of Postfix is straightforward: copy the `/etc/postfix` directory to some other directory (for example, copy `/etc/postfix` to `/etc/postfix−out`), create a separate spool directory for the new instance (for example, create `/var/spool/postfix−out` Each Postfix instance *must* have a separate queue directory). Edit the appropriate lines in `/etc/postfix−out/main.cf` to turn off filtering for outbound mail, bind to a different IP address and so on. Be sure to change the `queue_directory` setting to the location of the new queue directory.

The second Postfix instance shares the same binaries as the first instance so you don't have to recompile and install Postfix binaries a second time. To start the second instance, use the usual `postfix start` command except point to the directory containing the second instance configuration files. For example: `postfix −c /etc/postfix−in start`

A word about performance

The "simple content filtering" method is the easiest and most reliable way to filter messages with Postfix. However, performance suffers because each e−mail message has the overhead of the Perl interpreter startup, plus overhead of temp file creation.

Even so, the method described here has been able to filter upwards of 50,000 messages per day on a single Sun e250 server with 512MB memory, using two Postfix instances to separate inbound and outbound mail as described above. Linux servers on Intel 750Mhz platforms with 512MB demonstrated similar performance levels.

Ideally the Anomy program should be able to exist as a daemon, allowing use of the Postfix "advanced" filtering example. SpamAssassin already provides a daemon option. Unfortunately, Anomy Sanitizer is not written that way. We're currently evaluating various modifications to Anomy Sanitizer and will update this document if successful.

Comments, suggestions, criticisms, additions to this document?
Please e−mail **papers (at) advosys.ca**

Latest version of this document available at http://advosys.ca/papers/postfix−filtering.html
Copyright © Advosys Consulting Inc. Ottawa Canada. All Rights Reserved.

Last modified Jul 08 2002

Copyright and terms of use

Use of this document

Permission to use this document from the Advosys Consulting web site is granted, provided that (1) This notice notice appears in all copies, (2) use is for informational and non−commercial or personal use only and will not be copied, reprinted, or posted on any network, computer or broadcast in any media, and (3) no modifications of the document are made.

Educational institutions (specifically K−12, universities and community colleges) may reproduce the Documents for distribution in the classroom, provided that (1) the below copyright notice appears on all copies, and (2) the original Uniform Resource Locator ("URL") of the document on the Advosys Consulting web site appears on all copies.

Use of this document for any other purpose requires written permission of Advosys Consulting Inc.

Copyright Notice:
Copyright © Advosys Consulting Inc., Ottawa Ontario Canada.
All rights reserved.

Limitation of liability

Advosys Consulting take no responsibility for the accuracy or validity of any claims or statements contained in the Documents and related graphics ("the content") on the Advosys web site. Further, Advosys Consulting Inc. makes no representations about the suitability of any of the information contained in the content for any purpose. All such documents, related graphics, products and services are provided "as is" and without warranties or conditions of any kind. In no event shall Advosys Consulting Inc. be liable for any damages whatsoever, including special, indirect or consequential damages, arising out of or in connection with the use or performance of information, products or services available on or through the Advosys Site.

Trademarks

Product, brand and company names and logos used on the Advosys web site site are the property of their respective owners.