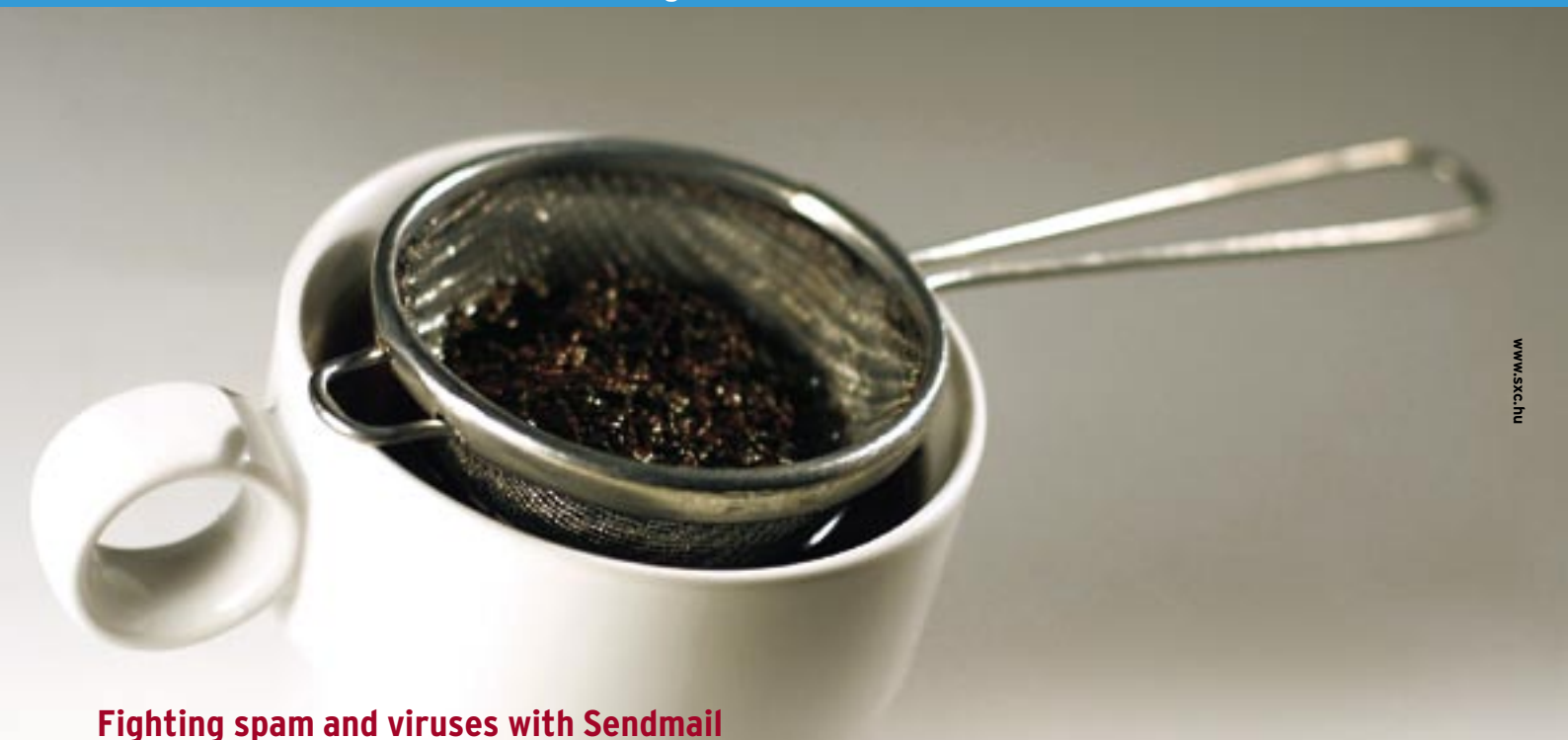

www.sxc.hu

Fighting spam and viruses with Sendmail

FILTERING STRATEGIES

A structured approach to Sendmail helps to maximize your spam and virus protection. **BY HANNES KASPARICK**

The versatile Sendmail Mail Transport Agent (MTA) offers several approaches to managing the virus and spam epidemic. This article introduces three workable anti-spam and anti-virus scenarios before finishing with a discussion of a radical approach that more or less entirely eliminates spam. The discussion assumes you have some basic familiarity with how to configure Sendmail. I will concentrate on practical add-on tools. Certain internal Sendmail configuration settings [1], as well as prevention strategies such as blacklists, also help to fight malware, though these techniques rarely provide a complete solution. For more information on configuring Sendmail, see the summary of sources at the Sendmail website [2].

Connecting Interface

Amavisd-new [3] is an interface between an MTA and a content filter such as a virus scanner and/or Spamassassin. See Linux Magazine, January 2006, for a discussion of Amavisd-new [4]. Various Amavis configurations are available for many MTAs. Amavis is pretty safe from a security point of view, as the Perl program is not susceptible to buffer overflows. Amavis does not normally need root privileges; it is happy to run in a chroot jail, and it gives administrators configuration options to prevent Denial of Service attacks based on mail bombs.

The interface is easily installed. For Debian-based systems, run *apt-get install amavisd-new*. As Amavis typically also checks for viruses, you will want to install a virus scanner. The configuration is located in */etc/amavis/amavisd.conf*.

There are several approaches to integrating Amavis with Sendmail. The recommended approach is the Sendmail Dual variant. In this scenario, Amavis accepts email on port 10024 of the local interface of the first Sendmail instance and forwards messages to the second Sendmail instance after verification.

Sendmail Dual

Running Amavis means setting up two Sendmail processes that manage separate mail queues. One process manages the receiving queue (MTA-RX), and the second the transmission side (MTA-TX): Amavisd-new plays piggy in the middle, and acts as a spam and virus filter. The MTA-RX process listens on TCP port 25, and reads its configuration from */etc/mail/sendmail-rx.cf* and */etc/mail/submit.cf*, along with the source file */etc/mail/hostname-rx.mc*.

Type *mkdir /var/spool/mqueue-rx* to create the mail queue directory. The following line gives you the required permissions:

```
chown root:amavis ➥
/var/spool/mqueue-rx && ➥
chmod 700 /var/spool/mqueue-rx
```

Then go on to define your own control socket to stop the two Sendmail processes from arguing:

```
define➥
(`confCONTROL_SOCKET_NAME',➥
`/var/run/sendmail/mta/➥
smcontrol-rx')dnl
```

The MTA-TX process binds port 10025 on the local interface and uses the normal mail queue and configuration file. To keep things readable, we will be using a shared source file */etc/mail/Hosname-tx.mc*. Define settings for receiving mail, including resource limits, in *hostname-rx.mc*. The sending side is handled by

hostname-tx.mc. One advantage of the Sendmail Dual installation is that the MTA-RX, which is accessible from the Internet, does not need to run with root privileges as it does not access the user home directories.

Solution 1: Amavis Home Alone

The simplest, and least flexible, approach to fighting spam and viruses is to use Amavis (Figure 1) on its own. The interface automatically integrates Spamassassin, its plugins, and an anti-virus program. If you fail to set the *$mydomain* variable, more or less nothing will work, so start by assigning a value. Define suitable values for both *$LOGFILE* and *$log_level* to enable troubleshooting and log analysis.

Then you have to consider what will happen to spam and virus-infected messages, and define *$final_virus_destiny* and *$final_spam_destiny* to match. Your options are *D_PASS*, *D_DISCARD*, *D_BOUNCE* >, and *D_REJECT*; responsible admins will opt for *D_BOUNCE*, rather than *D_REJECT* to allow for address spoofing.

This method uses the *$sa_** parameters to control Spamassassin. To tag every mail with an X-Spam header, set *$sa_tag_level_deflt = -999*. To allow users to identify spam by the email subject line, a clear *$sa_spam_subject_tag* is useful. To improve the detection rate, you may want Amavis to reference external sources (blacklists, DCC, Razor, Pyzor). To do so, set the *$sa_local_tests_only = 0* option.

Values between 2 and 2.5 have proved useful for *$sa_tag2_level_deflt*, which tags messages as spam or legitimate. If you have too many false positives, you will need to raise this value carefully. *$sa_mail_body_size_limit* lets you control the mail size above which Amavis does not perform spam checking on mail – large mails are very rarely spam. And *$sa_timeout* sets the number of seconds that Amavis waits for Spamassassin before forwarding messages. The whitelists and blacklists, as well as recipients that prefer to disable spam checking (*$spam_lovers*) can be configured flexibly.

Cascading Scanners

Solution 1 gives you virus checking, and it can even use multiple, parallel virus scanners. Once one virus scanner has detected a virus, it would be a waste of resources to feed the message to the other. The parameter *$first_infected_stops_scan = 1* prevents this.

You could also warn the sender of the infected mail by specifying the parameter *$warnvirussender = 1*; however, this service does not make much sense, as happy-go-lucky admins will probably treat your warnings as spam, or spoofed addresses will prevent you from ever being able to reach the perpetrator. If you prefer not to delete virus-infected messages automatically, you can set up a *$QUARANTINEDIR*.

Virus scanning can quickly consume resources on a mail server, thus leading to a Denial of Service condition. Mail bombs in particular expand to such an enormous size that when you unpack a mail bomb for scanning, you might easily run out of memory and swapspace. The *$MAXLEVELS* parameter mitigates this by restricting the nesting levels of files that have been compressed multiple times. After reaching the predefined level, Amavis stops trying to unpack the file. *$MAXFILES* restricts the number of files per mail.

The *$MAX_EXPANSION** parameter allows you to set a memory limit in bytes that can be used by the decompression process. Whenever an email exceeds one of these three limits, Amavisd-new version 20030616-p8 or later adds an ***UNCHECKED*** tag to the subject line. You should routinely filter

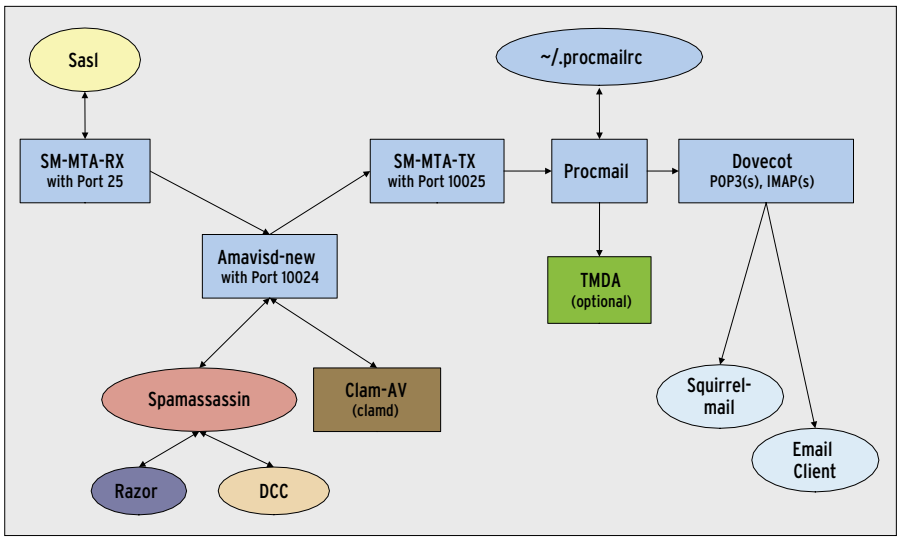

Figure 1: Simple, but fairly inflexible: Sendmail and Amavis battling spam and viruses on their own.

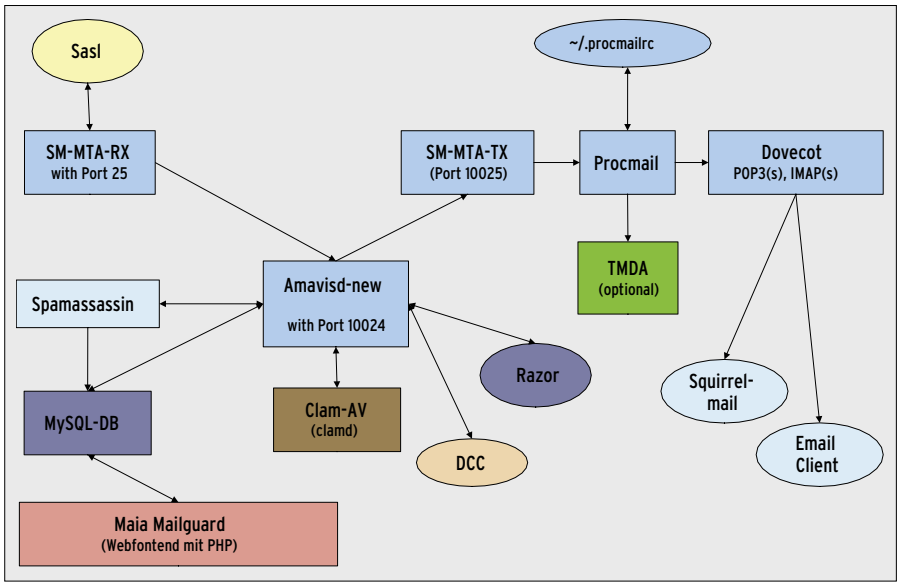

Figure 2: A more flexible, database-based scenario with Sendmail, Amavis, and Maia.

mails with this tag as they are likely to contain viruses.

Solution 2: Amavis with Maia

Solution 1 is easy to configure, although it does not leave you with much scope for user-specific settings. One way of extending Amavis is to add Maia Mailguard [7], as shown in the schematic in Figure 2. Maia is an easy-to-use, multiple-language Web front-end based on PHP and Perl. It uses a patched version of Amavisd-new, which stores user-specific settings in a MySQL or PostgreSQL database.

Users access their individual settings via the Web front-end. Users are required to log in with their complete (!) email address. After doing so, users can manage mailboxes, define individual blacklists (Figure 3), or change the threshold value for spam.

The price of more user convenience is more risk for the admin: more components always mean more risk of failure. If the database hangs, the server simply stops delivering mail. (It does store message temporarily, and delivers later if necessary.) To improve availability, admins can define multiple SQL servers.

The second critical point is that the Maia patch prevents you installing your distribution's binary security updates for Amavisd-new – this would simply remove the path. For mailservers with a large volume of mail, the performance hits due to repeated database access can also be significant. Again, administrators can work around this issue by assigning multiple servers.

Solution 3: Milter-based Spamd

For maximum individual configurability, without graphical overhead, a model like the one shown in Figure 4 is probably your best approach. It uses Sendmail's

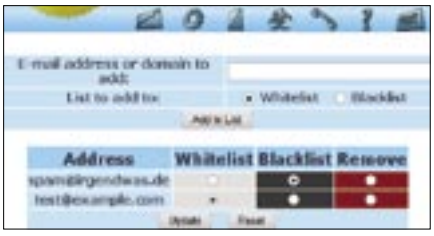

Figure 3: The Maia Web front-end gives mail users the ability to set up individual configurations – blacklists in this example.

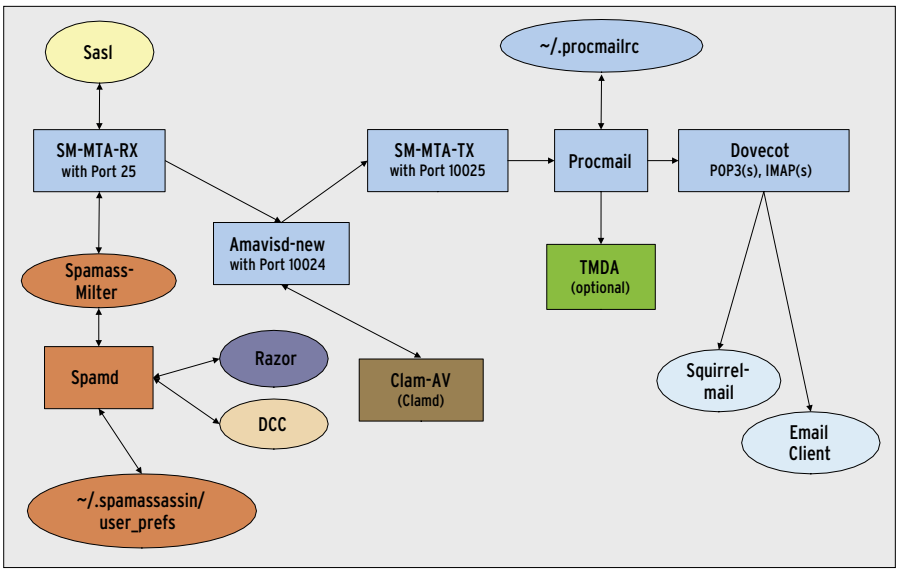

Figure 4: An even more flexible solution makes use of Sendmail's Milter interface.

Milter interface to access a separate instance of Spamd (the Spamassassin daemon). Each user can define their own settings in *~/.spamassassin/user_prefs*. Spamd accesses the *user_prefs* file and detects spam based on these values. If the file is missing, Spamassassin will automatically apply the administrator's preferences, which are located in */etc/spamassassin/local.cf*.

Sinvr Spamd needs access to the home directories, it runs under the root account. You can set up regular user-level access to the home directories for the Spamd process to avoid running the process as root, and therefore increase security. Solution 3 is easy to install: *apt-get install spamass-milter spamassassin*. Then add the following lines:

```
INPUT_MAIL_FILTER(➥
`spamassassin',`
S=local:/var/run/sendmail/➥
spamass.sock,
F=,T=S:4m;R:4m;E:10m')dnl
```

to the *hostname-rx.mc* configuration file for the MTA-RX, compile the configuration files by running *m4*, and relaunch Sendmail. Define global preferences in */etc/spamassassin/local.cf*. Check out [8] for a full list of options. The most important options are as follows: to enable individual user configurations, set *allow_user_rules 1*. To tag spam, use *required_score 2.5*, *rewrite_subject 1*, and *rewrite_header Subject ***Spam***.

The default detection criteria are fairly good; more granular configuration options are detailed at [9]. Spamassassin scores email content, and if the score exceeds a threshold, or *required_score*, it tags the message as spam. To boost the spam detection rate, it is advisable to assign a higher score to *RAZOR2_CHECK*, *RAZOR2_CF_RANGE_51_100*, and *DCC_CHECK*. The *PYZOR_CHECK* value is high enough.

Three Hunters: DCC, Razor, and Pyzor

The parameters just referred to specify the evaluation criteria for data from the three global mail networks, DCC, Razor, and Pyzor, which collect and evaluate mail hashes. The hash functions used for this purpose work in a different way than the better known MD5 and SHA1 hashes. They use what are called soft hashes, which allow some items in the mail to change, such as the target, without changing the hash.

Each mail server that uses these external networks sends the hashes of any mails it receives. If 100,000 hashes reach the networks – and this value is configurable – you can assume that the mail is spam. After all, it is unlikely that anyone would send 100,000 identical emails with legitimate content.

Run *apt-get* to install DCC, Razor, and Pyzor. Spamassassin automatically detects their existence. DCC requires a separate DCC server for volumes above 250,000 emails and will block any access above. To check whether these plugins are working, you can run Tcpdump to check the email headers of spam mails,

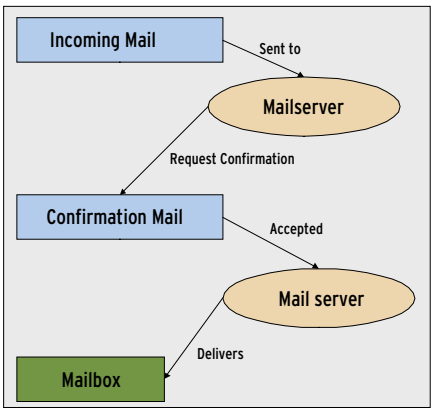

Figure 5: A challenge response approach to preventing spam by means of a Tagged Message Delivery Agent.

or the ports 6277/UDP (DCC), 2703/TCP, 7/TCP (Razor), and 24441/UDP (Pyzor).

You need to configure your firewall to allow outward bound traffic on these ports to allow connections to be opened to the respective networks. It may seem slightly paranoid to use all three programs in parallel, but doing so genuinely improves your spam detection rate.

Solution 4: Zero Percent Spam

A less-known method of fighting spam, which will only work under specific circumstances, is the Tagged Message Delivery Agent (TMDA) [10]. The TMDA adopts a challenge-response approach in which each message sent to the server for delivery has to prove its legitimacy to the server.

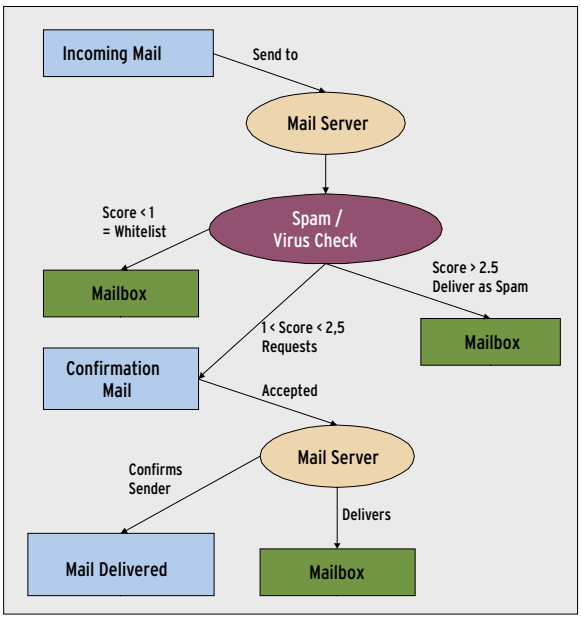

Figure 6: In a more sophisticated TMDA scenario, Spamassassin preselects messages.

Figure 5 shows you how this works in principle: an external mail server is attempting to send a message to a mailbox on your mail server. Your own mail server asks the sender to confirm and does not deliver the original mail until it has received the confirmation. Spammers will not send a confirmation message, as they automatically send mass mail, and will not receive the confirmation request due to address spoofing. This reduces undesirable messages to zero percent but at the cost of considerable overhead. Figure 6 shows a more sophisticated approach to TMDA.

If you opt for this zero tolerance approach, remember the following:

- As spammers typically use other people's addresses, your mail server should not bother these innocent bystanders by asking them for confirmation. In other words you should not give the TMDA messages that the filter has already identified as spam.
- Emails with a very low spam level can be delivered directly, without passing through the TMDA.
- If two TMDA systems happen to meet, infinite loops are possible. The TMDA homepage [10] states that this should not happen, but there are no guarantees. The FAQ on the homepage looks at more issues and suggests solutions.
- You are advised to set up a personal whitelist, independently of Spamassassin; your best approach is to add your addressbook. Whitelists are particularly important for registering with and using mailing lists.

One positive side effect is that anyone who sends you mail, and doesn't happen to be on your whitelist, receives confirmation that the message has arrived. At the same time, it should be noted that the TMDA is not the perfect solution for scenarios due to its complexity.

Performance

A mail server that attempts to provide a perfect solution in any scenario is always in danger of affecting performance whenever mail volume starts to increase. This prompted us to put the solutions discussed in this article through a performance test.

We used Postal [11] to stress test a Pentium 4 with a 2.8 GHz CPU and 512MB RAM. The benchmark program send emails with a maximum size of 15KB to the server over a period of five minutes. Solution 1 (Amavis Home Alone) proved capable of handling 600 emails per minute. It took another five minutes to deliver all the messages. Enabling ClamAV did not slow the process noticeably. If you use Spamassassin with external tests as described in Solution 3, the performance depends to a great extent on the available Internet bandwidth and latency. A TMDA (Solution 4) has a significant effect on performance.

All's Well that Ends Well

The first of the three designs we looked into should work fine at first attempt in most environments. Before enabling Solution 4, weigh the pros and cons carefully. What all four have in common is that they scale well thanks to their extremely modular design. ■

INFO

[1] Sendmail anti-spam features: *http://www.sendmail.org/m4/anti_spam.html*

[2] Sendmail: *http://www.sendmail.org*

[3] Amavis-new: *http://www.ijs.si/software/amavisd/*

[4] "Scan Manager: Filtering spam and viruses with Amavisd-new," by Larkin Cunningham; Linux Magazine, January 2006, p. 35.

[5] Securityfocus and Frsirt: *http://www.securityfocus.com* *http://www.frsirt.com*

[6] "Virus Checkers: Virus protection tools for the Linux environment," by James Mohr; Linux Magazine, January, 2006, p. 25.

[7] Maia Mailguard: *http://www.renaissoft.com/maia/*

[8] Spamassassin setup for incoming mail: *http://spamassassin.apache.org/full/3.0.x/dist/doc/Mail_SpamAssassin_Conf.html*

[9] Default score values for Spamassassin: *http://spamassassin.apache.org/tests_3_0_x.html*

[10] TMDA: *http://www.tmda.net*

[11] Postal: *http://www.cocker.com/postal/*