

# Anti-Spam configuration with Amavis, SpamAssassin and Postfix

Linuxdays.lu 2006 Server tutorial

Christian Mock, linuxwochen.at/quintessenz.org

<cm@quintessenz.org>

## Contents

- Short intro and background on spam
- DNSBLs: background, choosing, usage in Postfix
- SpamAssassin: background, configuration
- Amavis: spam-related configuration

## Short intro on spam

### *Risk assessment*

Automatically classifying messages as either spam or ham (non-spam) has the risk of mis-classification. A piece of spam classified as ham is a False Negative, a piece of ham classified as spam is a False Positive. In an ideal world, you would be able to reduce both counts to zero<sup>1</sup>.

Since we don't live in an ideal world, you need to think about the issue: what level of FP and FN can you and your organization accept? Would you rather have your users see no spam at all and risk losing legitimate mail, or can they live with a few spams per day and person and virtually never lose legitimate mail?

The next question, then, is what to do with the spam. The choices are

- Rejecting during the SMTP session
  - This means the sender will get a bounce message in the case of an FP, but the faked sender of a spam will not get one (because spam-sending SMTP engines don't generate bounces).
- Bouncing
  - This means the sender of an FP will get a bounce, but also that the faked sender of a spam will be spammed by you with a bounce for a message he didn't send.
- Dropping
  - This means you leave the faked spam sender alone, but the sender of an FP gets no notice of his mail disappearing, either.
- Tagging
  - You pass along the message to the recipient, but tag it in the subject and/or header. The recipient can automatically filter it to his “junk” folder, which also means that it most probably has the same effect as dropping, except you (as the mail admin) aren't the one dropping the important message from the important customer (which will inevitably come up in the discussion), but the end-user will.

### *Where does spam come from?*

Spam is either sent from the spammer's machine (hosted in a datacenter of an anti-social ISP somewhere), or from a machine the spammer abuses for his purposes. This can be an open SMTP relay, an open proxy, or a “zombie” machine infected with

---

<sup>1</sup> Actually, in an ideal world you wouldn't need spam filters at all.

malware that is used to spam and DDoS.

### ***What characteristics can we use to detect spam?***

- Where does the message come from? (IP address, hostname, ISP, country)
- How is it sent? (Characteristics of the SMTP session)
- Technical features (Headers, MIME structure, Text and HTML parts)
- Content (Easy for humans, hard for computers)

More material is in my talk held at the last linuxdays.lu:  
<http://www.tahina.priv.at/~cm/talks/spamblocking.pdf>

### **Blacklists/RBLs/DNSBLs**

"Realtime Blackhole Lists"/"DNS based Blacklists"

DNSBLs are lists of IP addresses or host/domain names that match certain criteria, like "open relay", "open proxy", "has sent mail to a spamtrap address recently", "assigned to a well known spammer", "located in *country*".

DNSBLs are queried via DNS; for IP address lists, the address is reversed like for a TR query; if we were to look for 209.88.103.4:

```
$ host -a 4.103.88.209.proxies.relays.monkeys.com
4.103.88.209.proxies.relays.monkeys.com IN A 127.0.0.2
4.103.88.209.proxies.relays.monkeys.com IN TXT "BLOCKED: See
http://www.monkeys.com/upl/listed-ip-0.cgi?ip=209.88.103.4"
```

For lists of host/domain names ("RHSBLs" for "Right Hand Side"), the hostname is prepended to the list's domain:

```
# host -t txt dvd-porno-shop.com.multi.surbl.org
dvd-porno-shop.com.multi.surbl.org text "Blocked, dvd-porno-shop.com
on lists [ws][ob][ab][jp], See: http://www.surbl.org/lists.html"
```

IP DNSBLs can be used to refuse mail from the listed machines; RHSBLs can be used to check URLs in the messages against.

You rely on an external entity, so choose a trustworthy source; DNSBL operators have listed ISPs they didn't like in the past, and one has closed down his list by listing ALL IP addresses.

My recommendations:

- [cbl.abuseat.org](http://cbl.abuseat.org/) - based on spamtraps, lists proxies/zombies, no FP seen;
- [list.dsbl.org](http://dsbl.org/) - open relays/proxies/etc; no FP seen;
- [sbl.spamhaus.org](http://www.spamhaus.org/) - lists well-known spammer's hosting IP addresses; virtual no chance of FP, operators highly regarded in the community;

More:

- <http://www.decluce.com/Articles.asp?ID=97>
- <http://www.moensted.dk/spam/>
- <http://openrbl.org/>

- <http://www.sconconsult.com/bill/dnsblhelp.html>

## Using DNSBLs in Postfix

Using a DNSBL in Postfix means mail from IP addresses on that DNSBL will be REJECTED! Does your policy allow that?

Using IP-based DNSBLs in Postfix is very simple, you just add the list to `smtpd_recipient_restrictions` in `/etc/postfix/main.cf`.

Postfix also has “`smtpd_client_restrictions`” (for the connecting host), “`smtpd_helo_restrictions`” (to check the SMTP HELO/ESMTP EHLO parameter given by the client), “`smtpd_sender_restrictions`” (for the envelope sender address); these react directly to the respective SMTP command by giving an error code if the result is “REJECT”, but some SMTP implementations ignore that, compatibility is biggest if the error is returned only when the client issues the RCPT TO command, even if it is the result of an earlier command. Plus we get a single configuration value to implement our SMTP-level policy...

“`smtpd_sender_restrictions`” contains a list of tests which can return "PERMIT", "REJECT" or nothing; the first match is used, so the general layout should be:

```
smtpd_recipient_restrictions =  
    <permit LAN users to send mail>,  
    <deny mail to non-local destinations>,  
    <whitelist for "spam lovers">,  
    <whitelist for spam FP>,  
    <block spam>,  
    permit
```

DNSBLs can be queried via "`reject_rbl_client dnsbl.domain.name`".

Example:

```
smtpd_recipient_restrictions =  
    permit_mynetworks,  
    reject_unauth_destination,  
    check_recipient_access hash:/etc/postfix/spamlovers,  
    check_sender_access hash:/etc/postfix/sender_whitelist,  
    check_client_access hash:/etc/postfix/client_whitelist,  
    check_helo_access hash:/etc/postfix/helo_restrictions,  
    reject_non_fqdn_sender,  
    reject_unknown_sender_domain,  
    reject_rbl_client sbl.spamhaus.org,  
    reject_rbl_client cbl.abuseat.org,  
    permit
```

In Detail:

```
permit_mynetworks,
```

Permit LAN users to send mail (LAN addresses defined in `mynetworks` in `main.cf`)

```
reject_unauth_destination,
```

Deny mail for non-local destinations (i.e. not `mydestination`, `virtual_*_domains`, `relay_domains`, ...); prevent unauthorized relaying. Everything which passes this line is therefore for a local recipient.

```
check_recipient_access hash:/etc/postfix/spamlovers,  
Allow certain addresses to receive unfiltered mail, e.g. <postmaster@luxembourg.tux-  
industries.com>
```

```
check_sender_access hash:/etc/postfix/sender_whitelist,  
check_client_access hash:/etc/postfix/client_whitelist,  
Whitelists for sender addresses (SMTP envelope!) and client IP addresses/hostnames,  
so you can whitelist people whose servers/ISPs happen to be on DNSBLs.
```

```
check_helo_access hash:/etc/postfix/helo_restrictions,  
Reject HELO/EHLO with my hostname, domain name or IP address
```

```
reject_non_fqdn_sender,  
Reject sender addresses without fully qualified domain part
```

```
reject_unknown_sender_domain,  
Reject sender addresses with non-existing domain (returns temporary 4xx error code,  
can take 5 days for mail to bounce if sender has DNS problems)
```

```
reject_rbl_client sbl.spamhaus.org,  
reject_rbl_client cbl.abuseat.org,  
Reject IP addresses on one of these lists
```

```
permit  
Anything else will be accepted
```

Example map files:

```
spamlovers:  
postmaster@luxembourg.tux-industries.com PERMIT  
abuse@luxembourg.tux-industries.com PERMIT
```

```
sender_whitelist:  
user@example.com PERMIT  
some-domain.com PERMIT
```

```
client_whitelist:  
10.2.3.4 PERMIT  
mail.example.com PERMIT
```

```
helo_restrictions:  
mailserver.luxembourg.tux-industries.com 554 That's me, liar!  
tux-industries.com 554 That's me, liar!  
10.4.5.6 554 That's my address, spammer!  
my-other-domain.lu 554 That's also me, liar!
```

Don't forget “postmap filename” after changing a map file.

## Testing the setup

If you know what you want, to avoid configuration errors (typos, ...):

```
soft_bounce = yes  
Turns all 5xx SMTP errors (hard errors) into 4xx (soft errors), which causes the  
sender to retry, so mail won't get lost.
```

If you want to know what a certain configuration would achieve without interfering with mail delivery, use `warn_if_reject`, e.g.

```
smtpd_recipient_restrictions =  
    [...]
    warn_if_reject reject_rbl_client new-dnsbl.org,  
    [...]
```

“`reject_rbl_client new-dnsbl.org`” is evaluated, but the results are only logged (“`reject_warning`”), not used to block. Analyze the log, and if you're happy, remove the “`warn_if_reject`” from the line.

After a while, the log file `/var/log/mail.log` should contain lines like these:

```
Jan 23 06:21:21 trumm postfix/smtpd[25929]: NOQUEUE: reject: RCPT  
from  
    DSL01.212.114.234.165.NEFkom.net[212.114.234.165]: 571 Service  
    unavailable; Client host [212.114.234.165] blocked using  
    list.dsbl.org; http://dsbl.org/listing?212.114.234.165;  
    from=<savspkseakkoqf@dcemail.com> to=<user@tahina.priv.at>  
    proto=SMTP helo=<DSL01.212.114.234.165.NEFkom.net>  
Jan 23 06:22:09 trumm postfix/smtpd[25929]: NOQUEUE: reject: RCPT  
from  
    unknown[211.187.64.233]: 554 <194.152.163.253>: Helo command  
    rejected: Don't lie to me.; from=<savspkseakkoqf@dcemail.com>  
    to=<user@tahina.priv.at> proto=SMTP helo=<194.152.163.253>
```

The first was rejected because the sending machine's IP address is listed in the dsbl.org DNSBL, the second because the HELO command contained my IP address.

## SpamAssassin

SpamAssassin (<http://www.spamassassin.org/>) is *the* Open Source anti-spam filter, continually improved and adapted to spammer's new techniques.

### *How does it work?*

SpamAssassin analyzes structure and content of a message with about 670 tests. Each test has a score associated with it; SA sums up the scores of the tests that matched, and if the sum is bigger than the (configurable) limit then the message is considered spam and can have “\*\*\*\*\*SPAM\*\*\*\*\*” added to the subject, X-Spam-Status headers, etc. The default limit is 5.0 points.

Tests include: MIME structure, keywords and key-phrases in headers and body, header analysis, DNSBL and RHSBL lookups, and Bayesian filtering.

Bayesian filtering is a technique where a database is trained on existing data sets; it works on single words (or “tokens”), where the training phase supplies the probability that a token occurs in spam and ham; then, you can look up the tokens from a new message in the DB and calculate the probability that it is spam or ham.

There's an “auto-whitelisting” feature which adapts the score of the current message based on the past scores messages from that sender have got; so if somebody who sent you low-scoring mail in the past and sends a message which triggers a few tests in SA now, the score will be adjusted downwards. The AWL database works on a combination of sender address and sending IP address, so a spam faking your friend's email address will not have its score adjusted since it will not come from your friend's

mail server.

## **Installation**

*Use a current version of SpamAssassin!* If you're using Debian Woody (3.0), put

```
deb http://www.backports.org/debian woody spamassassin
```

in your `/etc/apt/sources.list`, or get the source from <http://www.spamassassin.org/> and install it yourself. Sarge, the Debian release we're using, has an up-to-date version, but this will be frozen once Sarge becomes stable...

Take care to install all the recommended modules, too – especially `libnet-dns-perl` and `libmail-spf-query-perl` on Debian, or you'll lose all DNS and SPF-related functionality. Do not start `spamd` – we'll integrate SA into Amavis in a few moments.

Debian's SpamAssassin installs into  
`/etc/spamassassin` – Site-Wide configuration  
`/usr/share/spamassassin` – Rules

## **Configuration**

The system-wide SpamAssassin config file is in `/etc/spamassassin/local.cf`, per-user config is in `~/.spamassassin/user_prefs`. For reasons of simplicity, we'll only use site-wide config here.

We're using SA 3.0.2 in the examples; make sure to read the `Mail::SpamAssassin::Conf` documentation either via “man” or “perldoc” to see which options other versions support.

What do we want to achieve?

- Messages classified as spam are marked with “\*\*\*\*\*SPAM\*\*\*\*\*” in the subject
- A system-wide bayesian database with a reasonable size
- Use network tests (DNSBLs etc)
- Use the auto-whitelisting feature

Debian's `local.cf` is empty. We add the following statements:

```
auto_whitelist_path    /var/lib/amavis/.spamassassin/whitelist
bayes_path             /var/lib/amavis/.spamassassin/bayes
bayes_expiry_max_db_size 1500000
trusted_networks      127/8 192.168.1/24
skip_rbl_checks       0

ok_languages          de en it
ok_locales            en

rewrite_header subject *****SPAM*****

report_safe           0
report_contact        postmaster@tux-industries.com

lock_method flock
```

Explanation:

```
auto_whitelist_path    /var/lib/amavis/.spamassassin/whitelist
bayes_path             /var/lib/amavis/.spamassassin/bayes
```

These are in `~/spamassassin/` (per-user) normally, we put them in a central location to make them system-wide.

```
bayes_expiry_max_db_size 1500000
```

The default is much too low; 1 Mio uses about 32MB of disk space, so see how much “cached” RAM there is in the “free” output and size accordingly if you run a busy installation – you want the bayes DB to be cached all the time, and you want a large DB so the spams with lists of random words in them don't force the interesting spam keywords out of the DB.

```
trusted_networks 127/8 192.168.1/24
```

This should be automatically detected, but the algorithm isn't very reliable; give all IP addresses/networks of machines you trust, i.e. your LAN, your secondary MX etc.

```
skip_rbl_checks 0
```

Use the DNSBLs.

```
ok_languages de en fr
```

SA can guess the language used in a message and score messages in unwanted languages accordingly. The docs have a full list of the known languages.

```
ok_locales en
```

Which character sets do you expect mail to be in? “en” includes all western character sets, so e.g. spam from Korea is scored.

```
rewrite_header subject *****SPAM*****
```

How to mark the messages classified as spam (this setting has no effect in conjunction with Amavis).

```
report_safe 0
```

If set to “1”, spam is attached to a new message, or “wrapped”, so MUAs don't automatically display the HTML (which could verify the email address via a web bug or use an IE exploit to install malicious code); set this if you use Outlook. Has no effect when used with Amavis.

```
report_contact helpdesk@tux-industries.com
```

What address to use in the various text templates. Has no effect when used with Amavis.

```
lock_method flock
```

Set to “flock” for performance if you don't use NFS, RTFM otherwise.

## ***Feeding the Bayes DB***

Copy a few hundred pieces of ham and spam to two mbox files for training the bayes DB; it will train itself during operation, but that can take a few days, we'll give it better training (since these are human-selected messages) and get better results.

SpamAssassin needs at least 200 pieces of spam and ham before it starts using the Bayes DB.

There's a few such files on <http://cms-machine/>, use ham-small.mbox and spam-small.mbox for reasons of speed. When you set up a real server, try to get a good sample of *real* mail traffic for training, since this will heavily influence the quality of the result (and therefore, the acceptance from the users).

## Run

```
su amavis -c 'sa-learn --mbox --spam --showdots' < spam.mbox
su amavis -c 'sa-learn --mbox --ham --showdots' < ham.mbox
```

The “su” command is so the files get created with the right owner.

```
# su amavis -c 'sa-learn --ham --mbox --showdots' < ham-small.mbox
.....
.....
.....
.....
.....
```

Learned from 297 message(s) (302 message(s) examined).

The difference (302 vs 297) is because SpamAssassin doesn't learn from all messages; there may be duplicates (by Message-ID), or messages may be too old or too small.

In the end, it should look like this:

```
# ls -l /var/lib/amavis/.spamassassin/
total 1116
-rw----- 1 amavis amavis 1520 Jan 23 03:15 bayes.mutex
-rw----- 1 amavis amavis 86016 Jan 23 03:15 bayes_seen
-rw----- 1 amavis amavis 1314816 Jan 23 03:15 bayes_toks
# sa-learn --dump magic
0.000 0 3 0 non-token data: bayes db version
0.000 0 299 0 non-token data: nspam
0.000 0 297 0 non-token data: nham
0.000 0 47549 0 non-token data: ntokens
0.000 0 1094658931 0 non-token data: oldest atime
0.000 0 1106314819 0 non-token data: newest atime
0.000 0 0 0 non-token data: last journal sync atime
0.000 0 0 0 non-token data: last expiry atime
0.000 0 0 0 non-token data: last expire atime delta
0.000 0 0 0 non-token data: last expire reduction
count
```

“bayes.mutex” is a lock file; “bayes\_seen” contains the Message-IDs of the messages trained (and auto-learned), and bayes\_toks has all the token data.

“nspam” and “nham” are the number of learned spams and hams, respectively; “ntokens” is the number of tokens (words) in the DB, and the “atime” values are related to the expiry of the DB, which is the process to keep the size below the limit (bayes\_expiry\_max\_db\_size).

## Testing SpamAssassin

Save a single piece of spam to a file, in mbox format (e.g. “save” from mutt); run “spamassassin -t < file | less” and look at the output. Try this with a few messages and make sure there's RCVD\_IN\_DNSBL and BAYES\_nn in at least some of the X-Spam-Status headers, i.e. that DNS lookups and the bayes DB work.

At the end of the output, it shows a summary:

```
Content analysis details: (3.4 points, 5.0 required)

pts rule name description
-----
0.7 DATE_IN_PAST_12_24 Date: is 12 to 24 hours before Received: date
0.7 SUBJ_ALL_CAPS Subject is all capitals
1.6 PORN_URL_MISC URI: URL uses words/phrases which indicate porn (misc)
0.4 BAYES_60 BODY: Bayesian spam probability is 60 to 80%
[score: 0.7436]
```

There's something wrong here – no DNS tests seem to have happened. Let's look at SA's debug output (the “-D” flag):

```
# spamassassin -D -t < file >/dev/null
[...]
debug: failed to load Net::DNS::Resolver: Can't locate Net/DNS.pm in
@INC (@INC contains: /usr/share/perl5 /etc/perl
/usr/local/lib/perl/5.8.4 /usr/local/share/perl/5.8.4 /usr/lib/perl5
/usr/lib/perl/5.8 /usr/share/perl/5.8 /usr/local/lib/site_perl) at
/usr/share/perl5/Mail/SpamAssassin/Plugin/URIDNSBL.pm line 113.
[...]
```

Seems I didn't follow my own instructions and forgot the Net::DNS perl module; so “apt-get install libnet-dns-perl libmail-spf-query-perl” and the result looks much better:

```
Content analysis details: (18.0 points, 5.0 required)

pts rule name          description
-----
0.7 DATE_IN_PAST_12_24 Date: is 12 to 24 hours before Received: date
0.7 SUBJ_ALL_CAPS      Subject is all capitals
1.6 PORN_URL_MISC      URI: URL uses words/phrases which indicate porn (misc)
0.4 BAYES_60           BODY: Bayesian spam probability is 60 to 80%
                        [score: 0.7436]
3.8 RCVD_IN_DSBL       RBL: Received via a relay in list.dsbl.org
                        [<http://dsbl.org/listing?61.143.199.38>]
1.2 RCVD_IN_BL_SPAMCOP_NET RBL: Received via a relay in bl.spamcop.net
                        [Blocked - see <http://www.spamcop.net/bl.shtml?61.143.199.38>]
0.4 RCVD_IN_NJABL_PROXY RBL: NJABL: sender is an open proxy
                        [61.143.199.38 listed in combined.njabl.org]
3.1 RCVD_IN_XBL        RBL: Received via a relay in Spamhaus XBL
                        [61.143.199.38 listed in sbl-xbl.spamhaus.org]
1.0 URIBL_SBL          Contains a URL listed in the SBL blocklist
                        [URIs: dvd-porno-shop.com]
0.4 URIBL_AB_SURBL     Contains a URL listed in the AB SURBL blocklist
                        [URIs: dvd-porno-shop.com]
1.5 URIBL_WS_SURBL     Contains a URL listed in the WS SURBL blocklist
                        [URIs: dvd-porno-shop.com]
3.2 URIBL_OB_SURBL     Contains a URL listed in the OB SURBL blocklist
                        [URIs: dvd-porno-shop.com]
```

We went from 3.4 points to 18.0, i.e. from “not detected” to “very high probability of spam”.

## Amavis

Config file is /etc/amavis/amavisd.conf; the following variables are of interest:

```
##@bypass_spam_checks_acl = qw( . );
$sa_local_tests_only = 0;
$sa_auto_whitelist = 1;
$sa_mail_body_size_limit = 200*1024;
$sa_tag_level_deflt = -1000;
$sa_tag2_level_deflt = 5.0;

$final_spam_destiny = D_PASS;
$warnspamsender = undef;
```

Explanation:

```
#@bypass_spam_checks_acl = qw( . );
```

By default, no spam checks are run so Amavis needn't depend on SpamAssassin; we have to comment out this line to enable SA.

```
$sa_local_tests_only = 0;
```

Enable the DNS-based tests.

```
$sa_auto_whitelist = 1;
```

Use the auto-whitelist feature.

```
$sa_mail_body_size_limit = 200*1024;
```

Spam is usually below 200kB, and SA can take up a lot of RAM for big mails.

```
$sa_tag_level_deflt = -1000;
```

```
$sa_tag2_level_deflt = 5.0;
```

We want X-Spam-headers in all messages (first line), “\*\*\*SPAM\*\*\*” in the subject from 5 points on.

```
$final_spam_destiny = D_PASS;
```

D\_PASS means tagged spam goes to the recipient, who can filter it into a folder. D\_DISCARD drops the message (meaning nobody will detect false positives!), and D\_BOUNCE would harass the innocent victims whose mail addresses were stolen for this spam run.

```
$warnspamsender = undef;
```

DO NOT CHANGE THIS! Spam is sent with faked sender addresses. The only situation where setting this variable could make sense is when your mailserver is outgoing only and you want to alert your users that their mail is detected as spam...

There's many more Amavis features, but these are the basics to get going, and the Amavis config file is 90% comments explaining what all the options mean.

Run `/etc/init.d/amavis reload` to apply the new settings.

## Testing

Since all mail through our server will be scanned, we just need to send ourselves a piece of mail; start with a simple test to see if there's any signs of SpamAssassin in the headers:

```
date | mail -s test user
```

and look at it with “mutt” - the “h” keys shows all headers and should reveal something like this:

```
X-Virus-Scanned: by amavisd-new-20030616-p10 (Debian) at tux-
industries.com
X-Spam-Status: No, hits=-4.3 tagged_above=-1000.0 required=5.0
tests=ALL_TRUSTED, BAYES_00, NO_DNS_FOR_FROM
X-Spam-Level:
```

This means SpamAssassin is being used by Amavis. So let's try with our previously used sample:

```
/usr/sbin/sendmail user < file
```

and look at the result in mutt, again:

```
X-Virus-Scanned: by amavisd-new-20030616-p10 (Debian) at tux-
industries.com
X-Spam-Status: Yes, hits=21.1 tagged_above=-1000.0 required=5.0
      tests=BAYES_99, DATE_IN_PAST_12_24, NO_DNS_FOR_FROM,
PORN_URL_MISC,
      RCVD_IN_BL_SPAMCOP_NET, RCVD_IN_DSBL, RCVD_IN_NJABL_PROXY,
      RCVD_IN_XBL, SUBJ_ALL_CAPS, URIBL_AB_SURBL, URIBL_OB_SURBL,
      URIBL_SBL, URIBL_WS_SURBL
X-Spam-Level: *****
X-Spam-Flag: YES
```

## Advanced issues

Or: “Issues I will only mention in passing because time will run out well before the basics are done”

### ***Whitelisting/Blacklisting in SpamAssassin***

You may be getting newsletters which you actually subscribed to tagged as spam; or you may get spammed from a constant From: address and SA doesn't detect it. The solution is “whitelist\_from add@ress” or “blacklist\_from add@ress” in /etc/spamassassin/local.cf; don't forget “/etc/init.d/amavis reload” afterwards.

If you are subscribed to a mailinglist which gets messages tagged as spam (e.g. a ML for discussing spam issues), use Amavis' “whitelist\_sender”; there's already a long list of addresses in the default amavisd.conf (search for “nobody@cert.org”), make sure you use the envelope sender address.

### ***Performance Issues***

The maximum amount of messages/time SpamAssassin can handle is limited by the scan time per message; this is usually around 5-10 seconds for SpamAssassin, plus the virus scanning overhead. Our settings limit amavis to 2 concurrent connections, both in /etc/postfix/master.cf

```
smtp-amavis unix - - - - 2 smtp
and in /etc/amavis/amavisd.conf
```

```
$max_servers = 2;
```

If that is too low (i.e. your mail queue gets long during high activity periods), you need more concurrent connections. For SA processing the limiting factor is RAM (assuming you're running on a CPU dating from this millenium), roughly 20-30MB per concurrent process, and the on-disk size of the databases, so those can stay cached in RAM (“du -sh /var/lib/amavis/.spamassassin” gives an estimate for this value).

Increase both of these values depending on the amount of RAM in your machine; I run 5 concurrent processes on 256 MB mail gateways for a customer which do nothing but spam checking.

### ***Getting Feedback to SpamAssassin***

The efficiency of the Bayesian Filter depends heavily on the feedback your give – the auto-learning feature is nice, but can only detect extreme cases, so the fine-tuning is left to the user.

Basically, what you need to do is to feed back at false positives and false negatives using “sa-learn --ham” or “sa-learn --spam”. Doing this the manual way can get

boring quite fast, so set up a pair of aliases in /etc/aliases:

```
secret+spam: amavis+spam
secret+ham: amavis+ham
```

(The “secret” part is so that spammers can't easily guess the address and send their spam to the ham training address).

Create .forward files in /var/lib/spamassassin

```
# ls -la
[...]
-rw-rw-r-- 1 amavis amavis 40 Mar 2 2004 .forward+ham
-rw-rw-r-- 1 amavis amavis 41 Mar 2 2004 .forward+spam
[...]
# cat .forward+ham
"|/usr/bin/sa-learn --ham -p /dev/null"
# cat .forward+spam
"|/usr/bin/sa-learn --spam -p /dev/null"
```

What this does is to use Postfix' “recipient\_delimiter” feature to create two “sub-aliases” for the amavis account, which then get handled by Postfix according to the “.forward+extension” files; those feed the messages to sa-learn with the correct parameters.

You then can use the “bounce” functionality (and only that!) of your MUA to send a 1:1 copy of the interesting messages to these accounts; a pair of keybindings (I use “H” for “learn as ham” and “S” for “learn as spam”) makes this very easy to use.

If you happen to use a MUA which doesn't allow bouncing, you can forward the messages as MIME attachments and need to write a little script which extracts the attachment and feeds that to sa-learn... Or a script which extracts multiple of those attachments, sanitizes them, and feeds them to sa-learn, so Outlook users can drag-and-drop their false positives/false negatives into a new message (as attachments) and send this to the learning aliases; you get the idea.