

9

Bayesian Filtering

The Bayesian filter in SpamAssassin is one of the most effective techniques for filtering spam. Although Bayesian statistical analysis is a branch of mathematics, one doesn't necessarily need to understand the mathematics to use SpamAssassin's Bayesian filter.

Bayesian analysis involves *teaching* a system that a particular input gives a particular result. For Spam filtering, this teaching is repeated, many times over, with many spam and ham emails. Once this is finished, a Bayesian system can be presented with a new email and will give a probability of the result being spam. For best results, teaching should be a constant process.

To filter spam emails, the system is taught both ham and spam emails, until the filter has learned to differentiate between the two. Then, emails passed through the filter will be assigned a probability of being spam. When Bayesian filtering is used in conjunction with SpamAssassin's other spam detection rules, SpamAssassin approaches 100% detection of spam, with false positives (legitimate emails misclassified as spam) close to 0%.

Internally, the Bayesian engine provides a single probability figure for each email processed. This probability ranges from 0 (0% likelihood that an email is spam) up to 99 (99% likelihood).

In this chapter, the focus is on users who have an account on the local machine. A Bayesian database can be implemented using an SQL database. The principles of the Bayesian database are also valid for an SQL Bayesian database. Creating an SQL Bayesian database is covered in Chapter 14.

Scoring

SpamAssassin uses the same scoring system for Bayesian filtering as for other rules. Rather than using a multiplier, where each percentage point adds a fixed amount to the score, the probabilities are banded, and the bands assigned individual scores. This allows for greater flexibility in scoring.

In the default SpamAssassin configuration, a number of levels or bands are identified, and each is assigned a different score. The default score values are relatively conservative—from 0% to 60% the scores are negligible and beyond that the bands are assigned an increasingly high score. This approach helps to avoid false-positives. There is no direct relationship between the Bayesian probability of an email being spam and the value that is added to SpamAssassin's score for the email as a result of that probability.

The scores for Bayesian rules can be edited as for other rules. Bayesian rules have names such as BAYES_00 and BAYES_10. The exact rules for a particular version can be retrieved with the following commands:

```
$ cd /usr/share/spamassassin/
$ grep "^score\WBAYES_" *.cf
score BAYES_00 0 0 -1.665 -2.599
score BAYES_05 0 0 -0.925 -0.413
score BAYES_20 0 0 -0.730 -1.951
score BAYES_40 0 0 -0.276 -1.096
score BAYES_50 0 0 1.567 0.001
score BAYES_60 0 0 3.515 0.372
score BAYES_80 0 0 3.608 2.087
score BAYES_95 0 0 3.514 2.063
```

score BAYES_99 0 0 4.070 1.886The default scores can then be overridden as required, by adding alternative scores to `/etc/mail/spamassassin/local.cf` for a site-wide change, or to `~/ .spamassassin/user_prefs` for a change that only affects one user:

```
score BAYES_00      -1
score BAYES_80      3
score BAYES_95      4
score BAYES_99      4.5
```

Note the use of a negative score for emails that are unlikely to be spam.

Altering the scores for rules is discussed in greater detail in Chapter 12.

Training

When using the standard disk-based Bayesian database, SpamAssassin keeps a Bayesian database for each user account it is run under, not for each user it processes email for. The alternative is to use SQL; this is covered in Chapter 14.

Although Bayesian filtering is enabled by default, SpamAssassin will not use the filter until it has learned enough spam and ham emails to be able to make a decision when processing an email. SpamAssassin has an auto-learn facility, which allows emails to be used to train the Bayesian filter automatically. It also has command-line tools to allow a user or system administrator to train the Bayesian filter manually.

To use automatic learning, set the `bayes_auto_learn` flag to 1. This can be configured site-wide in the `/etc/mail/spamassassin/local.cf` file, and can be overridden in a user's `~/ .spamassassin/user_prefs` file. Two other configuration flags also affect auto-learning, and are the thresholds for learning ham and spam. These values are in the same units as SpamAssassin's score for each email.

```
bayes_auto_learn      1
bayes_auto_learn_threshold_nonspam  0.1
bayes_auto_learn_threshold_spam     12.0
```

When auto-learning is enabled, any email that is assigned a score of less than `bayes_auto_learn_threshold_nonspam` is learned as ham. Any email that is assigned a value of greater than `bayes_auto_learn_threshold_spam` is learned as spam. There are rules whose scores

are *not* taken into account when the decision to auto-train is made. This includes any auto-whitelist rules (the auto-whitelist is covered in Chapter 13), and certain other rules.

It is recommended that the `bayes_auto_learn_threshold_nonspam` threshold is kept low (close to or below zero). This will avoid the situation where a false-negative spam email is used as an example of ham to train the Bayesian filter. Keeping the `bayes_auto_learn_threshold_spam` threshold high is to some extent a matter of choice, although it should be above the scores of any false-positives that have been received in the past. For the default spam threshold of 5, false positives may occur, possibly up to a score of 10. Thus, using an auto-learn threshold of less than 10 for spam may cause ham to be accidentally learned as spam. As long as there are no false-positives there will be no issue. Once a false positive occurs the Bayesian database will begin to lose effectiveness, and future Bayesian results will be compromised.

Using auto-learning to train the Bayesian filter may take some time. It typically takes around 200 spam and 200 ham emails to train the Bayesian filter. Not all spam and ham emails have scores that enable them to be used for auto-learning. Never rely only on auto-learning, as SpamAssassin does not learn from all spam or ham emails. This makes SpamAssassin less effective.

The `sa-learn` command is used to train the Bayesian filter with email messages that are known ham or spam. The SpamAssassin installation routine will have placed `sa-learn` in the path, normally in `/usr/bin/sa-learn`.

It is used on the command-line, and is passed a directory, file, or series of files. If the maildir format is used, then a directory or series of directories can be passed in:

```
$ sa-learn --ham ~/.maildir/.Trash/cur/ ~/.maildir/cur
Learned from 75 message(s) (175 message(s) examined).
```

If the mbox format is used, then the `mbox` flag should be used, so that SpamAssassin searches the file for more than one email:

```
$ sa-learn -mbox --spam ~/mbox/spam ~/mbox/bad-spam
Learned from 75 message(s) (175 message(s) examined).
```

If SpamAssassin has already learned from an email, `sa-learn` detects this and will not process it twice. In the example above, 100 of the 175 emails had been processed already (either auto-learned or through a previous invocation of the `sa-learn` command) and were ignored on this run. The remaining 75 emails had not been processed before.

If `sa-learn` is passed a number of messages, then there may be no feedback for some time. The `--showdots` flag provides feedback in the form of dots '.' whenever an email is processed:

```
$ sa-learn --spam --showdots ~/.SPAM/cur ~/.SPAM/new
.....
Learned from 20 message(s) (25 message(s) examined).
```

Confirming Operation

Once SpamAssassin has been trained with enough spam and ham emails, it will begin to apply the Bayesian test automatically, if it is enabled. However, not every email will receive a Bayesian score. The Bayesian subsystem only gives a probability if it can make an authoritative estimate. To do this, it needs to recognize words (more correctly **tokens**) that it has seen in previous ham or spam emails. One way to confirm the operation of the Bayesian filter is to examine the headers of

emails that have been processed by SpamAssassin and search for the result of a `BAYES_` test. The `x-spam-status` header in an email may look like this:

```
X-Spam-Status: No, hits=3.5 required=4.0 tests=BAYES_50,DATE_IN_PAST_03_06,
RCVD_IN_RFCI,RCVD_IN_SBL autolearn=no version=3.00
```

The `x-spam-status:` header lists all the tests that the email triggered along with some other details. In this example, the `BAYES_50` test was fired, indicating around 50% probability that the email was spam. If any `BAYES_` tests are listed in an email header, then the Bayesian filter is in operation. Headers are covered in more detail in Chapter 10.

Filter Training

As spam is constantly changing, the Bayesian filter must be kept up to date. Keep the following in mind when using the auto-learn feature of SpamAssassin:

- The auto-learn feature should be configured to only learn from the emails that are definitely ham or definitely spam, but not the ones in between the two auto-learn thresholds. Emails with very low or very high scores are the ones that SpamAssassin can detect without Bayesian filtering. It is the middle ground that Bayesian filtering can assist with most. By omitting these emails from the Bayesian learning process, the effectiveness of the Bayesian filter is decreased.
- There is a chance that emails may be mistakenly tagged if auto-learn alone is relied upon. If a spam is taught to SpamAssassin as a ham, or vice-versa, then this will adversely affect all future Bayesian results.

For these reasons, it is recommended that a policy of regular manual training is used.

It is important to teach SpamAssassin a varied selection of both spam and ham email. If this is not done, the Bayesian filter will become ineffective. If a manual learning process is used, it should be used weekly or more frequently.

User Involvement

Only users can accurately decide if an email is spam or ham. Here is one approach to ensure that spam emails are not tagged as ham:

1. When users receive a spam email in their inbox, they should not delete it. Instead they should move the email to a separate folder for incorrectly tagged spam. This ensures that the inbox and trash folder only contain ham, and the spam folder only contains false-negative spam.
2. Periodically, run `sa-learn` for that user, using the spam in the folder for incorrectly tagged spam.
3. Since all received spam emails are in the spam folder, the contents of the inbox, trash, and any other folders can be used with `sa-learn` as ham.
4. If spam email is automatically filtered to a separate folder, then users should review the contents before running `sa-learn` on it. Any wrongly identified emails should be moved to a separate folder for false-positive ham email, and these should be processed with `sa-learn` separately.

5. The contents of the folders that SpamAssassin has learned from should be archived or deleted.

If the user can be relied on to perform Step 1 on a day-to-day basis, then Steps 2 and 3 can be automated on a daily or weekly basis, provided the user is checking email and not on vacation. Step 4 will have to be a manual process. However, it can be made much easier for the user by limiting the amount of spam that has to be checked. Chapter 15 describes how to separate spam into different folders depending on the score that SpamAssassin gives each email. This will limit the amount of email that a user has to check.

Local Users

Users who can `telnet` or `ssh` to the host machine can access the command line directly to tag spam and ham emails. If they use an email client on the host, and it can be configured to run a command, then it can invoke `sa-learn` directly.

Pine is a text-based email client, available from <http://www.washington.edu/pine/>. It can be configured to run a command using the '|' (vertical bar or pipe symbol) key when reading an email. Pine has to be explicitly configured to allow piped commands. This is done by setting `enable-unix-pipe-command` from the `config` section of the `settings` page. If this is done, Pine will prompt for the command:

```
Pipe message 1 to :
```

The user should type the following command:

```
Pipe message 1 to : sa-learn --ham --no-sync
```

Pine will report the result of the command:

```
Learned from 1 message(s) (1 message(s) examined).
```

This learns an email as ham. The `--no-sync` flag defers the rebuilding of the Bayesian database, which can be time-consuming, and allows control to be returned to the mail client more quickly. In SpamAssassin versions before 3.0, the `--norebuild` flag should be used. SpamAssassin adds the entry to the `bayes_journal` database, until the data can be incorporated into the main Bayesian database. Choosing not to rebuild the database is slightly inefficient for subsequent Bayesian queries, and at some later time, `sa-learn` should be run with the `--sync` flag to ensure the data is added to the main Bayesian database, and efficiency is improved.

In SpamAssassin versions before 3.0, the `--rebuild` flag should be used in place of the `--sync` flag.

```
$ sa-learn --sync
sync'd Bayes databases from journal in 0 seconds: 925 unique entries (1115
total entries)
```

This could be added to a daily or hourly cron task with the following entry, which runs `sa-learn --sync` at five minutes past every hour:

```
5 * * * * /usr/bin/sa-learn --sync
```

Unlearning

If an email has been incorrectly learned as either ham or spam, it can be unlearned by passing the `--forget` flag to `sa-learn`. This reverses the learning of the message, and it can then be relearned as the opposite type of message. To unlearn an email, it should be placed in a separate mailbox or maildir folder. The command for to process an mbox format folder where the email is in the `unlearn` folder would be:

```
$ sa-learn --forget --mbox /path/to/unlearn
Learned from 8 message(s) (9 message(s) examined).
```

For a maildir configuration, the whole folder can be unlearned using wildcards:

```
$ sa-learn --forget /path/to/.maildir/unlearn/cur/*
Learned from 8 message(s) (9 message(s) examined).
```

Auto-learn Thresholds

The auto-learn thresholds can be altered on a site-wide basis, or for an individual user. These can be configured site-wide in the `/etc/mail/spamassassin/local.cf` file, and overridden for each user in their `user_prefs` file:

```
bayes_auto_learn_threshold_nonspam    0.1
bayes_auto_learn_threshold_spam       12.0
```

Bayesian Database Files

If SpamAssassin is run for each user separately, each will have a separate Bayesian database, reflecting only the spam that they receive. This is stored in `~/.spamassassin/`.

If SpamAssassin is run under a single system account for all users, either via the MTA, global `procmailrc` or `spamd`, then there will be only one Bayesian database shared by all the users. This will be stored in `~/.spamassassin/` for the user account used.

The Bayes database that SpamAssassin uses does not grow in size unchecked. SpamAssassin keeps track of the various words it has learned, and also when they were learned, and automatically removes old entries to make way for new ones. This feature is called **auto-expiry**. Auto-expiry can be turned off by setting the `bayes_auto_expire` value from 1 to 0:

```
bayes_auto_expire 0
```

SpamAssassin keeps the Bayesian database in three files in the `.spamassassin` directory within a user's home directory. The format used is usually Berkeley DB format:

```
bayes_journal
bayes_seen
bayes_toks
```

The `bayes_journal` file is used as a temporary storage area. Sometimes the `bayes_journal` file is not present. This file is generally relatively small, with a size of around 10kb. The `bayes_seen` and `bayes_toks` files can each be several megabytes in size.

The configuration parameter `bayes_expiry_max_db_size` can be altered to specify the maximum number of tokens that should be stored in the database. This parameter does not specify an absolute file size. The default value of 150,000 normally results in a database file of around 6Mb.

Removing a Bayesian Database

A Bayesian database can become corrupt, or the data inside it unreliable. This may happen when many ham emails have been learned as spam, or vice versa, and the Bayesian subsystem of SpamAssassin subsequently produces incorrect results. If there is no audit trail of emails that have been processed incorrectly, and the Bayesian filter appears to be working incorrectly, then the recommended practice is to remove the database and start afresh.

To remove a Bayesian database, remove (or archive) all files matching `bayes_*` in the user's `~/ .spamassassin/` directory. Unless Bayesian auto-learning is disabled, SpamAssassin will create a new database when suitable emails are processed.

Once a Bayesian database has been removed, if auto-learning is configured or the `sa-learn` command is used, SpamAssassin will recreate the Bayesian database. It is advisable to use the `sa-learn` command to train the Bayesian filter as quickly as possible.

Sharing a Bayesian Database

Sharing a Bayesian database has pros and cons. The cons are:

- A single large database will take slightly longer to process, when learning and processing emails. This will slow SpamAssassin slightly in normal use, and may slow SpamAssassin considerably for tasks such as bulk learning using `sa-learn`.
- The spam and ham that a user receives alters the contents of the Bayesian database. There may be an overlap where some words appear in ham emails of one user, yet appear in spam emails of another user. In general, when users share a single Bayesian database, the effect is to dilute the effect of the Bayesian subsystem.

The pros of sharing a database are:

- Sharing a database gives some protection against new phases in spam. When spammers develop a new technique, the larger volume of emails passing through a shared database may result in quicker learning of the new technique.
- If the volume of email is low, then a shared database may be more effective as individual databases may not be updated frequently enough for the Bayesian subsystem to be worthwhile.

The best practice is to use a separate database for each user. If all users share the same SpamAssassin configuration, perhaps because SpamAssassin is integrated into the mailer and is always run under the same user account, then a shared Bayesian database is in use.

If SpamAssassin is being run for individual users and they are to share a single Bayesian database, then the database files should be located in a common location and be writeable by all users, for example, with the following commands:

```
# groupadd bayes_users
# mkdir /var/spool/bayes_db
# chown root: bayes_users /var/spool/bayes_db
# chmod 0770 /var/spool/bayes_db
```

Add the appropriate users to the `bayes_users` group.

Override the default location for the Bayesian database, either globally in `/etc/mail/spamassassin/local.cf` or for each user in `~/.spamassassin/user_prefs`.

```
bayes_path      /var/spool/bayes_db/bayes
bayes_file_mode 0770
```

The path should include an additional part of the filename. The actual files used are created by adding `_journal`, `_seen`, and `_toks` to this path.

To test this configuration, each user should run `sa-learn` with the `-D` flag. Any errors will be noted on the command line.

```
$ sa-learn -D --spam ~/.maildir/.SPAM/new/
bayes expire_old_tokens: lock: 29610 cannot create tmp lockfile
/var/spool/bays_db/bayes.lock.hostname.domain.com.29610 for
/var/spool/bays_db/bayes.lock: No such file or directory
```

In this example, the user has not edited the `user_prefs` file correctly, and the path to the Bayesian database is incorrect.

Once the Bayesian database is suitably trained by using `sa-learn`, confirm that emails are being tagged, as described above.

Disabling Bayesian Filtering

To disable Bayesian filtering, set `use_bayes` to 0 globally in `/etc/mail/spamassassin/local.cf` or for each user in `~/.spamassassin/user_prefs`:

```
use_bayes 0
```

In addition to turning off the `BAYES_*` rules, turn off SpamAssassin's auto-learn feature. This is located in the `/usr/share/spamassassin/10_misc.cf` file, and can be overridden at the user level in the `user_prefs` file too:

```
bayes_auto_learn 0
```

Summary

The Bayesian filter is an important part of SpamAssassin. It requires training with both spam and ham before it will operate. This training should continue on a regular basis.

Although the Bayesian filter includes the ability to automatically learn from emails it processes, this should not be the only method used to train the Bayesian filter. It is important that wrongly classified emails are not taught to the Bayesian filter, as this will decrease the effectiveness of the filter.