WritingRules

Writing and Testing New Rules

This is a straightforward guide to writing your own add-on rules for SpamAssassin. This was originally written by MattKettler.

Why you don't need custom rules

First, this guide assumes you're already familiar with SpamAssassin and already have a working setup. If you don't yet have SpamAssassin running, I'd suggest starting off with the default ruleset. Custom rules are really more for tweaking SpamAssassin to better suit your personal email, and isn't something you generally need to worry about at the start. Most sites run just fine with the default ruleset and no custom rules at all. This guide also assumes you're relatively familiar with basic SpamAssassin configuration options, such as using blacklist_from and whitelist_to. I've tried to keep this guide very simple, and perhaps to a degree that makes it seem too easy. You should realize that this guide is covering material that should be treated as intermediate to advanced configuration, not something you do 2 days after you manage to get SA working for the first time.

Why you might want custom rules

Since custom rules are generally not needed, why would you want to use some? Well, the default SpamAssassin ruleset is tuned to a somewhat generic sampling of mail that comes from the corpus submitters. If the typical kinds of email coming to your network are significantly different than those used in the corpus, you may get a lot of false positives or false negatives. Whitelists, blacklists and tweaking scores of the default rules can help, but might not be sufficient in all cases.

Where to add them

Before we start writing rules, I have a warning for you. Do NOT add your rules to the .cf files in /usr/share/spamassassin. The reason for this warning is simple. When you upgrade SA (and you should do so somewhat regularly), all the existing rules in /usr/share/spamassassin will be deleted and replaced by the new default ruleset. All your hard work and customization will be lost. I'd only put changes in those files if it's a temporary fix for a bug in a rule that you expect will be fixed in the next version of SA. (ie: if there's a typo-fix in CVS, you might edit your .cf to match it). So where do you put your rules if not in /usr/share/spamassassin? Well, there's two places. And your choice of place depends on how your copy of SpamAssassin is set up, and what you want the rules to do:

/etc/mail/spamassassin/local.cf is the place of choice for site-wide application of a rule. Rules placed here get applied no matter what user invokes SpamAssassin.

~/.spamassassin/user_prefs is best if you want to have a rule only run when a particular user runs SA. Bear in mind that this is the user that executes SA, not the user who the mail is addressed To:. If you run SA from a sendmail milter, the only user_prefs that will be used is the one for the user sendmail runs as (usually root). Even if you do have a site-wide configuration, user_prefs may be useful to you by allowing you to add rules to another user account's user_prefs and test them using the command line prior to adding them to your local.cf.

Note: if you use spamd, rules placed in user_prefs will be IGNORED by default. If you add the allow_user_rules option to your local.cf you can get spamd to honor them. However, before you enable it, you should know that this is disabled by default for security reasons. In theory a malicious local user might be able to exploit spamd with a clever regex and gain root permissions. I know of no specific vulnerabilities of this type in spamassassin at this time, but it is a possibility. I'd only turn this on if you trust your local users not to try to hack root.

Ok, enough with the what's, where's, why's and why nots. Let's move on to the mechanics of writing rules.

Writing basic rules

Body rules

For our first rule, let's start with the simplest type of rules, the basic "body" rule. These rules search the body of the message with a regular expression and if it matches, the corresponding score is assigned. Body rules also include the Subject as the first line of the body content. See [DumpTextPlugin]

Let's look at a really basic fictitious rule:

```
body LOCAL_DEMONSTRATION_RULE        /test/
score LOCAL_DEMONSTRATION_RULE 0.1
describe LOCAL_DEMONSTRATION_RULE         This is a simple test rule
```

This rule does a simple case-sensitive search of the body of the email for the string "test" and adds a 0.1 to the score of the email if it finds it. Now, this rule is pretty simple as rules go. It will match "test" but also "testing" and "attest". The describe statement contains the text which will be placed into the verbose report, if verbose reports are used (this is the default setting for the body, in Spamassassin version 2.5x and upwards).

In regular expressions a \b can be used to indicate where a word-break (anything that isn't an alphanumeric character or underscore) must exist for a match. Our rule above can be made to not match "testing" or "attest" like so:

```
body LOCAL_DEMONSTRATION_RULE        /\btest\b/
```

The rule can also be made case-insensitive by adding an i to the end, like this:

```
body LOCAL_DEMONSTRATION_RULE        /\btest\b/i
score LOCAL_DEMONSTRATION_RULE 0.1
```

Now the rule will match any combination of upper or lower case that spells "test" surrounded by word breaks of some form.

Resources on Perl Regex Syntax

Perl Regular expressions are quite flexible and powerful. I could easily write an entire book on the different kinds of syntax you can use. At this point, I've given you a taste of some of the syntax, but if you're not familiar with regular expressions, you can read one of the many tutorials on the web regarding them.

Here's some sites you might want to check out for information on regular expressions:

http://www.english.uga.edu/humcomp/perl/regex2a.html
http://www.perldoc.com/perl5.6/pod/perlre.html
http://www.troubleshooters.com/codecorn/littperl/perlreg.htm
http://directory.google.com/Top/Computers/Programming/Languages/Regular_Expressions/Perl/

You could use your linux box and it's perl documentation:
If you have perl-doc installed you can type at a linux shell prompt:

perldoc perlretut

- for a good "from scratch" tutorial. perldoc perlre
- for users who already have some regexp experience.

Recommended book to learn the Perl programming language:

- Learning Perl, 4th Edition
- By biran d foy, Tom Phoenix, Randal L. Schwartz
- Publisher: O'Reilly

Header rules

Now let's move on to header rules. Header rules let you check a message header for a string. Most commonly these rules check the Subject, From, or To, but they can be written to check any message header, including non-standard ones. Let's pick up our "test" rule and change it into one that checks the subject line.

```
header LOCAL_DEMONSTRATION_SUBJECT        Subject =~ /\btest\b/i
score LOCAL_DEMONSTRATION_SUBJECT        0.1
```

In these rules, the first part before the =~ indicates what the name of the header you want to check is, and the rest is a familiar regular expression. The header name itself is always case-insensitive, so the above rule will match a subject: line containing "test" or a SUBJECT: line containing "test".

Checking the From: line, or any other header, works much the same:

```
header LOCAL_DEMONSTRATION_FROM        From =~ /test\.com/i
score LOCAL_DEMONSTRATION_FROM        0.1
```

Now, that rule is pretty silly, as it doesn't do much that a blacklist_from can't. Usually if you're making a From line rule you'll be doing so to use more sophisticated rules. However, I wanted to illustrate how to make one, and also point out that some punctuation characters are part of the regex syntax, and if you want to use them literally, you need to put a \ in front of them. Normally in a perl regex . is a wildcard, but this example will look for "test.com" and not match "testzcom".

There's also an option to look at all the headers and match if any of them contain the specified regex:

```
header LOCAL_DEMONSTRATION_ALL        ALL =~ /test\.com/i
score LOCAL_DEMONSTRATION_ALL        0.1
```

Not very commonly used, but this feature can also be used to do a case-sensitive check on a header name (it will look at the whole lines, not just the parts after the colon). Note that if you want to use the '^' character here, you must put an m at the end of your line, which will look at the header one line at a time.

```
header LOCAL_DEMONSTRATION_WEIRD_FROM  ALL =~ /^FrOM\:/m
```

Notes about rule scores

A few short words about the behavior of the "score" command.

- rules with a score set to 0 are not evaluated at all
- rules with no score statement will be scored at 1.0, unless 3 or 4 is true
- rules starting with a double underscore are evaluated with no score, and are intended for use in meta rules where you don't want the sub-rules to have a score.
- although intended for the sa development effort, any rule starting with T_ will be treated as a "test" rule and will be run with a score of 0.01 (nearly 0). This can be handy when testing rules so you don't have to create score lines for them if you think you're not going to keep them.

Last in this section I'll leave you with a word about choosing scores. I'd suggest starting off with a very low score that won't impact messages very much, like 0.1. Watch your rule and make sure it fires when you want and isn't firing when you don't want. Then start increasing the score to make it have more effect, but try not to go overboard. You should be very reluctant to have a custom rule with a score over 1.0 unless you're sure it's not going to hit *any* nonspam messages. Also keep in mind that you can write rules to only match on non-spam messages and give them negative scores to try to correct false-positive problems. Strong negative scores should also be treated with a bit of caution, but aren't quite as serious. Generally false positives can cause problems as valuable mail might get skipped over, but false-negatives are a minor nuisance, so you can be a bit more liberal with negative scores.

Advanced Scoring

Score commands can have 1 or 4 parameters. If there is only one parameter (the norm) then that score is used all the time. Example:

```
score LOCAL_DEMONSTRATION_ALL   0.1
```

With four parameters:

- The first parameter applies when the Bayesian classifier and network tests are not in use
- The second parameter applies when the Bayesian classifier is not in use, but the network tests are
- The third parameter applies when the Bayesian classifier is in use, but network tests are not
- The fourth parameter applies when the Bayesian classifier and network tests are both in use

Example:

```
score LOCAL_DEMONSTRATION_ALL   0.1 0.3 0.3. 0.1
```

Advanced rule types (meta, uri, rawbody and friends)

In addition to body and header rules, SA supports several other kinds of rules. In general these are used much less often than the header and body types, but it is still worth having a short introduction to these and what they do.

URI rules

URI rules are very simple, they only match text in the URI's contained in plain text and HTML sections of mail. This is very handy for searching for links containing spam advertised sites.

For example This rule will look for web links to www.example.com/OrderViagra/

```
uri LOCAL_URI_EXAMPLE         /www\.example\.com\/OrderViagra\//
score LOCAL_URI_EXAMPLE        0.1
```

Rawbody rules

Rawbody rules allow you to search the body of the email without certain kinds of preprocessing that SA normally does before trying body rules. In particular HTML tags won't be stripped and line breaks will still be present. This allows you to create rules searching for HTML tags or HTML comments that are signs of spam or nonspam, or particular patterns of line-break.

As an example this rule looks for a HTML comment claiming the message was "created with spamware 1.0":

```
rawbody LOCAL_RAWBODY_EXAMPLE /\<\-\-! created with spamware 1\.0 \-\-\>/
score LOCAL_RAWBODY_EXAMPLE 0.1
```

Meta rules

Meta rules are rules that are boolean or arithmetic combinations of other rules. This allows you to do things like create a meta rule which fires off when both a header and a body rule are true at the same time.The following example uses a boolean check and will add a negative score to emails from news@example.com containing the body text "Monthly Sales Figures"

```
header __LOCAL_FROM_NEWS  From =~ /news\@example\.com/i
body   __LOCAL_SALES_FIGURES         /\bMonthly Sales Figures\b/
meta LOCAL_NEWS_SALES_FIGURES  (__LOCAL_FROM_NEWS && __LOCAL_SALES_FIGURES)
score  LOCAL_NEWS_SALES_FIGURES -1.0
```

Note that the two sub rules start with a double underscore, so they are run and treated as having no score, as per item 3 in section 2.4.

Also note the slash placed before the @ sign. This is important otherwise perl will try to interpret it as an array.

Meta rules can also be arithmetic, but this feature was absent from the original implementation of meta rules in 2.4x. An arithmetic meta rule can be used to tell if more than a certain number of sub rules matched. For example this meta rule will fire if 2 or more of the strings "test1" "test2" and "test3" are found anywhere in the body:

```
body __LOCAL_TEST1         /\btest1\b/
body __LOCAL_TEST2         /\btest2\b/
body __LOCAL_TEST3         /\btest3\b/
meta LOCAL_MULTIPLE_TESTS (( __LOCAL_TEST1 + __LOCAL_TEST2 + __LOCAL_TEST3) > 1)
score LOCAL_MULTIPLE_TESTS 0.1
```

The value of the sub rule in an arithmetic meta rule is the true/false (1/0) value for whether or not the rule hit.

If you want to weight your sub rules differently, you can apply weights to them like you would in any standard equation:

```
meta LOCAL_MULTIPLE_TESTS (((0.8 * __LOCAL_TEST1) + (0.5 * __LOCAL_TEST2)) > 1)
```

Finding more information and examples

See more advanced description and examples: WritingRulesAdvanced

Perhaps the rule you want has already been written: check out the collection of custom rules at SpamAssassinRules.

More information on the different rule-writing keywords that can be used to write more advanced rules, such as rawbody, meta, and uri can be found in the Mail::SpamAssassin::Conf manpage. You can also glean some great examples by reading through the default rules in /usr/share/spamassassin.

If neither of these are of help, try posting on the spamassassin-talk mailing listor check out the unofficial SA rules wiki that several of the more avid custom rule writers use http://www.exit0.us/.

Writing better rules

Ok, so you now know the basics of how to make some simple rules, and where to read up to build more complicated rules. So you're all ready to get started, right? Well, not quite yet. Let me take a minute to give you some advice to make good, or at least better rules than you might make on your first try.

Rule name suggestions

You'll probably want to have an easy way to tell your rules from the distribution set, so what you'll want to do is use some kind of naming convention. The two common ones that people use to distinguish their custom rules are to prefix the rule name with either LOCAL_ or their initials (ie: MK_). This also makes it obvious where to go looking to fix a rule that's causing problems.

Picking good strings to search for

As you can see from the example rules above, picking a text pattern to search on to make a basic rule for SpamAssassin is fairly easy. However it's also easy to create rules which seem good at casual glance but have unintended consequences. Here's some of my generic advice on rule writing.

First, be as specific as you possibly can. In general single word checks make lousy rules as they don't take into account the context the word is used in. In most cases, phrases are much better to look for, unless the word has no use in normal text.

Next, think about possible alternate uses for the word or phrase you came up with. Many phrases that might sound like good rules for porn spam also wind up matching messages regarding health issues. Think about health newsletters, order status messages, and legitimate financial newsletters. Is your phrase likely to be in them? What about personal emails that have a little off-color humor (to the extent that's acceptable to you at least)? Personal love letters? A note from your insurance company or doctor? Don't forget to think about alternate meanings for words, check in a dictionary. You can also try a web search for your phrase or word.

Using the lint feature

It's also very easy to have a typographical error in your rules. A bad typo can cause SpamAssassin to have to silently skip large chunks of your config files before it can make sense of the rules again. To check your rule syntax for errors, run the command line version with the – lint option. Look for syntax errors complaints and other messages of the sort in the output:

```
spamassassin --lint
```

ALWAYS lint your rules. I've made so many typos I can't count them all. Many had bad side effects before I flogged myself into always linting my rules.

If the --lint output doesn't give you enough information, you can try also looking at the debug output. This might give you some better hints as to what files SA is reading, but does generate a lot of output to scroll through.

```
spamassassin --lint -D
```

A brief introduction to using a corpus

You can also do some thorough testing of your rules by building your own corpus of spam and nonspam, or downloading a public one, and testing your rules against that corpus using the tools in the masses/ subdirectory of the spamassassin distribution. I'm not going to go into great detail here as it's a bit on the advanced side, but the basic process is to first use mass_check to generate spam and nonspam logs, then use hit_frequencies to generate statistics on a per-rule basis.

The public corpus is at http://spamassassin.org/publiccorpus/

Automatic rule generation

A script apparently associated with the SoughtRules automatic score generation can be downloaded from trunk:

```
svn checkout http://svn.apache.org/repos/asf/spamassassin/trunk
cd trunk/masses/rule-dev
./seek-phrases-in-corpus ham:dir:~/Maildir/ spam:dir:~/Maildir/.bad.spam-missed/
```

The script contains instructions. You tell it a directory or file containing spam, and a directory or file containing non-spam, and it'll come up with rules that match the spam but not the non-spam.

If you have run MassCheck, you can generate rules for the set of spams that were below the default threshold by doing something like (you'll probably need to change the paths to corpora):

```
cd trunk/masses
awk '$2 < 5' < spam-*.log > missed_spam.ids
./mboxget < missed_spam.ids > missed_spam.mbox
cd rule-dev
./seek-phrases-in-corpus ham:dir:$HOME/masscheckwork/ham/ spam:detect:~/masscheckwork/nightly_mass_check/masses
/missed_spam.mbox  > seek.out
```

Thanks

Thanks to Daniel Quinlan, Michael Moncur, Randy Diffenderfer, and many other members of the SA development team and users of the SA for their patient feedback and suggestions.

Matthew Kettler

Information to be integrated

If you wish to change the score from the default for a single user, add a line like this to that user's ~/.spamassassin/user_prefs file:

```
score NAME_OF_TEST 3.0
```

This would be most appropriate if that one user is (for example) a bankruptcy lawyer, and needs to see all email dealing with bankruptcies. This user's ~/.spamassassin/user_prefs file would then contain score modifications for those tests that otherwise might flag his client's emails.

If you need to change the score for system-wide processing, create this type of line in your /etc/mail/spamassassin/local.cf or /etc/spamassassin/local.cf file depending on your installation (or any *.cf file in that directory). This is needed from time to time when a network black list used by SA stops functioning, and you need to turn off that specific test by setting its score to zero.

Most systems do not allow new rules to be defined in user_prefs files – they need to go into local.cf or at least that file's directory.

See Chris Santerre's guides at http://www.rulesemporium.com/

The SA development team assigns scores to the distribution set of rules when we do a major release using the GeneticAlgorithm. Some details on how we do that are in RescoringProcess.

If you write custom rules, you really should test them using a hand-classified set of mail messages divided into spam and non-spam collections – see HandClassifiedCorpora – and the mass-check tool – see MassCheck. Also, once you have mass-check results, the overlap tool can be used to check to see if a rule is hitting mails that are being hit by another rule: UsingOverlap

If you have questions concerning rules, or want help testing rules against others' corpora, it's appropriate to post your rules (or attempts) to the spamassassin-users list.

If you think your rules might need some tweaking for speed, you can measure their overhead by profiling them as described in ProfilingRulesWithDprof.

Once you've written and tested a few rules, consider sharing them with us at SpamAssassinRules.