New Features in SpamAssassin 3.2.0

For Large-Scale Receivers

Justin Mason
MAAWG Dublin, June 2007
Intro

• One of SpamAssassin's development team
• Wanted SA 3.2.0 to be faster
• Wrote a few of these features, kept a close eye on others
• Will do a slide or 3 on each feature
Feature: “sa-compile”

- SpamAssassin rulesets are specified in configuration files on the server.
- Compiled to Perl bytecode at runtime.
- SpamAssassin's "body" ruleset is the slowest (about 60-65% of the runtime).
- Would be great to speed this up.
How SpamAssassin body rules work

```plaintext
foreach line (lines in rendered message) {
    if (line contains /pattern_1/) {
        got_hit("RULE1"); last;
    }
}

foreach line (lines in rendered message) {
    if (line contains /pattern_2/) {
        got_hit("RULE2"); last;
    }
    ...
}
```
This is surprisingly efficient!

- due to efficiency in perl's regular expression implementation
- and due to the fact that emails are very short in general
- especially when HTML is parsed beforehand
However, it can be improved

- in particular, matching those regular expressions in parallel would help...

- Many commercial products based on opensource SpamAssassin do this, in various ways

- It'd be nice to see it in open-source
re2c

- compiles set of (basic) regexps into C code which implements a parallel-matching DFA state machine
  » compile to native code, with “cc -O2”

- Matt Sergeant contributed "re2xs", which converts (basic) Perl regexps into input for "re2c" and generates a Perl XS module
The plugin

- re2xs adapted into a new SA plugin and a user interface script for administrators:
  - Mail::SpamAssassin::Plugin::Rule2XSBody
  - sa-compile

- run "sa-compile" after adding new rules or updating an existing ruleset; it'll take a minute to compile the regular expressions into a parallel-matching DFA for you
Not a total replacement

- re2c regexps quite different from Perl regexps
  - so we have to follow every potential match with a "double-check" using the full perl regexp
- Some regexps are just too complex, so we're left with a small leftover legacy set
  - (~ 40% of the default "body" ruleset)
Real-world results

- 10% to 20% speedup on a mixed corpus of real spam and non-spam mails
- Faster if you add additional SARE rulesets (24% in my test)
- Runtime went from 51.2 seconds to 38.9 seconds

» (measured using SpamAssassin's "mass-check" mass scan tool)
How to use it

• Edit `/etc/mail/spamassassin/v320.pre`

• Remove the "#" from this "loadplugin" line:
  
  ▶ # Rule2XSBody - speedup by compilation of ruleset to native code
  ▶ # loadplugin Mail::SpamAssassin::Plugin::Rule2XSBody

• Run "sa-compile" as root

• Restart the "spamd" server, Amavisd-new, etc.
Feature: short-circuiting

• SpamAssassin used to run all rules before giving a spam/nonspam diagnosis

• obviously, some spam is "super-spammy"

• can be marked after running only 10% of ruleset

• ideally we should be able to "short-circuit" the scan process if the mail is already marked high enough to be spam
Harder than it seems

• checking to see if we can "short-circuit" like this can itself impose too much of a hit
  » with 1000 rules, performing short-circuit checks after each one is slow

• nonspam mails generally hit only 1 or 2 rules
  » we will eventually have to use all rules when scanning them, anyway
Still harder than it seems

- if we allow s/c to mark a mail as nonspam, then we open a hole that spammers can exploit to get their mails marked as nonspam if we're not careful
  
  » spammers love these holes

- need to be careful about rule ordering: you can't exit early if you may be able to swing back in the opposite direction with a high-scoring rule later
The 3.2.0 approach

• allow the administrator to specify the rules they want to allow to short-circuit the scan

• more intuitive, since the administrator gets to decide which rules are trustworthy enough

• less "magic" happening out of sight behind the scenes
Rule priority

- rule order can be specified in configuration
- "cheap", fast, reliable rules can be set up to run first, and short-circuit if hit (such as spamtrap hits)
- followed by "less cheap" reliable rules (such as DKIM whitelists)
- followed by all the rest
Shortcircuiting example

# local whitelists, or mails via trusted hosts
meta SC_HAM (USER_IN_WHITELIST||USER_IN_DEF_WHITELIST||ALL_TRUSTED)
priority SC_HAM -1000
shortcircuit SC_HAM ham
score SC_HAM -20

# slower, network-based whitelisting
meta SC_NET_HAM (USER_IN_DKIM_WHITELIST||USER_IN_SPF_WHITELIST)
priority SC_NET_HAM -500
shortcircuit SC_NET_HAM ham
score SC_NET_HAM -20

# run Spamhaus tests early, and shortcircuit if they fire
meta SC_SPAMHAUS (RCVD_IN_XBL||RCVD_IN_SBL||RCVD_IN_PBL)
priority SC_SPAMHAUS -400
shortcircuit SC_SPAMHAUS spam
score SC_SPAMHAUS 20
Results

- On my (small, vanity-domain) server, it's resulted in an average of 20% less time spent scanning

- Mails that short-circuited as "spam" completed scans in an average of 0.2 seconds; as "ham", in an average of 0.5s

Feature: “msa_networks”

- Dynablock rules cause false positives for some ISPs with dynamic address pools.
- Mails from dynamic users arrive from the pool via a trusted Mail Submission Agent, which authenticates them.
- However, SpamAssassin can't tell that the MSA authed the user, so a dynablock rule fires (incorrectly).
We try to recognise MSA authentication

- some MTAs record this in a “Received” header (RFC 3848, defining “Received: with ESMTPSA” etc., especially useful)

- some don't record it at all in headers :

- hence “msa_networks”: specify the IP address (ranges) where your MSAs live

- SpamAssassin will assume that any message via those is from a trusted host, since your MSA authenticated the user
Feature: backscatter ruleset

• “backscatter” = bounces, in response to spam sent using a fake address at your domain

• you had nothing to do with it, but the remote MTA still sends you:
  » "user unknown" bounces
  » "your mail was probably spam!" bounces
  » "your mail had a virus!" bounces
  » challenge/response challenges

• volume can be as high as spam itself :(

© 2007 MailChannels Corporation
Add a ruleset to detect it

- based on Tim Jackson's "bogus-virus-warnings.cf" ruleset
- much extended, and made a core part of SpamAssassin
- added whitelisting of "good" relays, so you can rescue bounces of messages that really were sent by your MTAs
Feature: mod_perl module

- spamd implemented as a mod_perl Apache module
- contributed as a Google Summer of Code project by Radoslaw Zielinski
- Apache includes lots of well-tested, optimized, scalable code to do all the TCP heavy-lifting, so this is more efficient than spamd
mod_perl module, contd.

• this speed comes at a cost: simplified configuration support and no setuid mode

• in the SpamAssassin 3.2.0 release tarball in the “spamd-apache2” directory, if you're interested

• a little bit beta! hasn't received massive real-world deployment yet, so watch out ;)

© 2007 MailChannels Corporation
Feature: Amazon EC2 support

• The “Elastic Compute Cloud” is a virtual server farm operated by Amazon

• incredibly easy to bring up and shut down new virtual "servers" to match demand

• a great way in theory to deal with high load caused by spam storms: start up some servers at EC2, and offload your spam filtering load to there until it dies down
Amazon EC2 support, contd.

- EC2 is billed partly on bandwidth used, so we need to reduce that

- added new features to the spamc/spamd protocol to support this:
  - `-z`: compression
  - `--headers`: return just rewritten headers
  - `--ssl`: SSL encryption

- even without EC2, this is good for cross-internet use of spamd, in general
Feature: sa-update

- tighten up the rule-development life cycle by automatically publishing new rules
  - rules are added to our SVN repository for testing
  - automatically tested against several fresh collections of mail
  - if they pass, they're added to the published set in the next day's updates
- (coming; still working on this, post-release)
That's it!

• Thanks for listening!
• Slides will be blogged at http://taint.org/tag/sa320
• Thanks also to MailChannels
• Questions?