RulesProjSandboxes

Rules Project: Sandboxes

Initially, the rules project sandboxes SVN will consist of the existing (empty) rules directory in Subversion (the CVS replacement used by the ASF). Each committer will have their own sandbox to begin development in an unconstrained manner:

rules/sandbox/<username>/

Every person who has listed their rule set on the Apache SpamAssassin Wiki will be invited once the PMC approves the project; there are some rule sets only listed at SARE or Exit0, but those people are invited to join too, of course. There is absolutely no quality or experience requirement for the sandbox although we may later provide some tools to make it easier to avoid name collisions and such.

It is expected that someone (don't know and don't care who) will eventually write scripts to test, filter, and pull rules automatically into the production rules. I am intentionally deferring decisions around that area, though.

What does providing a sandbox for everyone do?

- easy to join (you just have to sign a CLA and get an @apache.org account)
- no expectation of... well, much anything; no quality or experience requirement for the sandbox
- easy for us to import rules (manually or automatically) into main rule body
- easy to move forward with further development around automatic updates and all of the other (hard) ideas we've talked about, but I really want to keep this dirt simple.
- ability to help direct future development of the rules project (as it extends beyond sandboxes, sandboxes will remain just sandboxes, of course).
- can produce multiple "output rule sets" in the long run: conservative, aggressive, sub-areas: bounces, drug rules, etc.
- uses SVN and therefore has version control

In other words, this solves the main part of our "rules problem" – the hurdle of getting rules "over the wall". No longer will we need individual bugs for rule submissions, or need to go to 3 different sites to look for rule ideas, etc. Many of our best rules have come from SARE and the Wiki.

Also, it's expected that many of the rules will never go into the main rules body – someone may write rules for a specific type of annoying mail (not even necessarily spam), or maybe someone will be focused on super-aggressive rules for the brave folks out there. We can even produce multiple "output rule sets" in the long run: conservative, aggressive, sub-areas: bounces, drug rules, etc.

Some notes picked out of followup discussion:

It is possible to keep rules 'private', and in your own checkout only, by not checking them into SVN.

If you do want to have the rules visible for collaboration, but not used for automatic mass-checks or promotion, that could be done by just keeping them in a file that doesn't end in ".cf". (SpamAssassin's standard is that they have to end in ".cf" to be considered valid rules files.)

Repository Organization

- rules/core/ = standard rules directory
- rules/sandbox/<username>/ = per-user sandboxes
- rules/extra/<directory>/ = extra rule sets not in core

The proposal is for rules/core/ to become the rules directory for trunk (3.2 and later, via SVN externals which will make their inclusion seamless in the standard SA tree). The sandbox is discussed further in RulesProjMoreInput.

Promotion of rules from sandbox to rules/core/ is discussed in RulesProjPromotion.

(Update: bug 5123 gets rid of rules/core – now that is simply part of the old "rules" directory to keep things simpler.)

Extras/

We'll want to discuss the structure and process behind creating new extras directories further once we reach a critical mass of committers in the rules project; but here's some initial thoughts on typical 'extra' rulesets.

- non-spam-oriented rules, such as the anti-virus-bounce ruleset
- non-English-language rulesets (although see RulesNotEnglish)
- rules that positively identify spam from spamware, but hit <0.25% of spam
- an "aggressive" rules set might include rules that hit with an S/O of only 0.89, but push a lot of spam over the 5.0 threshold without impacting significantly on ham

(ChrisSanterre: Seeing this breakdown of dirs, gave me an idea. Why not set the "aggresiveness" of SA for updates? Like how SARE has ruleset0.cf (no ham hits), ruleset1.cf (few ham, high S/O), etc., with each "level" of rule set file getting slightly more aggressive, risking (though not necessarily seeing) slightly higher FP rates. Users could set some config like supdate=(1-4), with 1 being the most conservative, and 4 being the most aggresive (with the knowledge that more aggresive *could* possibly cause more FPs).

JustinMason: I think for now it's easiest to stick with the 'load aggressive rulesets by name' idea, rather than adding a new configuration variable. For example, aggressiveness is not the only criteria for what rulesets to use; we'd have to include config variables for "I want anti-viral-bounce rulesets", too.)

Overlap Criteria

BobMenschel: The method I used for weeding out SARE rules that overlapped 3.0.0 rules, was to run a full mass-check with overlap analysis, and throw away anything where the overlap is less than 50% (ie: keep only those rules which have "meaningful" overlap). Manually reviewing the remaining (significantly) overlapping rules was fairly easy. The command I use is: perl ./overlap ../rules/tested/$testfile.ham.log ../rules/tested/$testfile.spam.log | grep -v mid= | awk ' NR == 1 { print } ; $2 + 0 == 1.000 && $3 + 0 >= 0.500 { print } ' >../rules/tested/$testfile.overlap.out

DanielQuinlan: 'By "throw away", do you mean put into the bucket that is retained going forward or did you mean to say "greater than 50%"?'

BobMenschel: 'By "throw away anything where the overlap is less than 50%" I meant to discard (exclude from the final file) anything where the overlap was (IMO) insignificant. This would leave those overlaps where RULE_A hit all the emails that RULE_B also hit (100%), and RULE_B hit somewhere between 50% and 100%
of the rules that RULE_A hit.'

JustinMason: Like Daniel, I'm confused here. as far as I can see, you want to keep the rules that do NOT have a high degree of overlap with other rules, and throw out the rules that do (because they're redundant). in other words, you want to throw away when the mutual overlap is greater than some high value (like 95% at a guess).

Again, this is something we can handle further down the line.