RuleSandboxes

Rule Sandboxes

Every committer has a 'sandbox' area in the rules source tree ('rulesrc'). This is an area for rules under development.

'Sandbox' rules are expected to break, fall over, conflict with existing rules, contain syntax errors, etc. – the compiler should catch those. It's a low-risk development area, in contrast to the "safe" published rulesets.

Repository Organization

The "rulesrc" tree looks like this:

- rulesrc/core/ = standard rules directory, built into SpamAssassin by default;
- rulesrc/extra/<directory>/ = extra rulesets, not distributed with "core" SpamAssassin;
- rulesrc/sandbox/<username>/ = the sandboxes.

Building

When you run "make" from the masses/ directory, the compiler will run, and performs a few basic "compilation" tasks:

- it copies the 'core' ruleset from rulesrc/core/ into "rules/".
- it'll also look through all rules files in rulesrc/sandbox/*/ , and copies rules found to "rules/70_sandbox.cf" as testing rules.

"Testing rules" are given a *T_* prefix, to make it clear (both to users and the code) that they're rules in testing. They also get a little rudimentary lint- and syntax-checking; if they fail, they're not copied.

If their names collide with existing rules seen elsewhere, they will be auto-renamed to avoid the collision.

Extras/

Some initial thoughts on typical 'extra' rulesets:

- non-spam-oriented rules, such as the anti-virus-bounce ruleset
- non-English-language rulesets
- 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

The main thing is that it's a place to put rules that match the ruleset's publishing criteria (whatever they may be) without putting them in the SpamAssassin core.

Getting Started

1. Create a new sandbox in rulesrc/sandbox (e.g. rulesrc/sandbox/duncf):

```
mkdir rulesrc/sandbox/whatever
svn add rulesrc/sandbox/whatever
```

2. Create as many files in this directory as I wish, containing as many rules as I wish, named "NN_anything.cf" (where "NN" is 2 digits).

```
vi rulesrc/sandbox/whatever/NN_anything.cf
```

3. Check those files into SVN:

```
svn add rulesrc/sandbox/whatever/NN_anything.cf
cd rulesrc
svn commit
```

4. Watch as the preflight mass-checker starts checking them immediately at the PreflightBuildBot (link).

The rules will be renamed to include a T_ prefix. (This lets the engine know that it's a rule under test, so it gets a minimal score; it's also easier to tell them apart from the "core" ruleset in the nightly mass-checks.)

5. Wait until the mass-check results are in, and look at the results at the RuleQaApp (link). (The link to the results of that exact mass-check is included in the output from the PreflightBuildBot's final step, if you're wondering.)

6. (Optionally) wait for a nightly mass-check. (Again, the results will show up on the RuleQaApp (link), specifically under the "Nightly" section of the mass-check list.)

7. Tweak until I'm satisfied with my rule.

8. ???

9. Profit!

(To expand – those last two steps require some thinking about how we're to do the promotion criteria, which is as yet undone.)

Moving Rules To Core

Once rules are deemed 'good enough', according to whatever the promotion criteria are, they can be promoted from the sandbox into core. TODO. We haven't quite solidified the promotion criteria for rules yet... see RulesProjPromotion for ongoing discussion.

Editing Another Developer's Sandbox

Editing another developer's sandbox rules is generally frowned upon. However, there are instances where this is necessary and permitted:

1. Individual developers may allow more access to files in their sandbox by adding comments to the beginning of the file.

2. If content in the sandbox has been promoted into the core codebase or rules, it is treated like any other portion of the code.

3. If a developer's sandbox has had no changes for more than 6 months, it may be considered abandoned. After sending an email to the original developer, if no replies are received for 2 weeks, updates can be made by other developers. Note, it is still best practice to notify the original developer that you're making the change and the revision control comments for the update should also clearly state the reason for the update.

4. In the case of any problem that is considered time-sensitive and not covered above, any patch with votes to commit following RTC, i.e. requiring three votes and no vetoes before committing, can be immediately applied.

(Note: this is documentation; if you want to read the historical planning docs, see RulesProjectPlan and RulesProjSandboxes.)