RulesProjCompiler

Rules Project: Rules Compiler

In thinking about stuff in RulesProjectPlan, specifically RulesProjPromotion, myself (Justin) and Daniel quickly got bogged down in thinking about how to deal with the following categories of rules in the ruleset:

- automatically-promoted rules (ones copied from sandboxes into the core ruleset based on S/O etc.)
- manually-promoted rules (ones copied based on developer agreement, despite possibly having crappy S/Os etc.)

namely the following problems arise:

- should rules be automatically renamed?
- will automatically-promoted rules have to be copied into the core ruleset as a script-generated file, which is then checked into SVN?
- how will that preserve revision history?
- will we wind up with "svn churn" as the script rebuilds and re-checks-in the file every night?
- will we have to rename/copy/move auto-promoted rules manually, after the fact, to give them decent names, locations etc.?

Given those problems, I suggest that we may be better off using a compiler. In other words, I propose

- all rules files in the rules project directory are considered "source"
- the rules/ dir in the SpamAssassin engine build tree is the output directory
- we have a "rules compiler" script.
- there are a couple of new config directives, aimed at the rules compiler:
    - one indicates whether a rule should be "manually promoted"
    - one indicates the desired output name for the rule
- the rules compiler is run when you run "make" in the engine source
- the source for the rules, and the code itself, is in the rules project tree, and remains in each developer's sandboxes etc., for them to move around and work with as they see fit.
- SVN revision history is therefore preserved.

The compiler runs recursively through the "source" tree, picking out rules. If the rule is listed as "manually promoted" with the relevant directive, it'll be promoted; otherwise that's up to the auto-promotion criteria noted previously.

When a rule is promoted, it's output to a file in the output directory:

- The build directives are omitted
- the lines of rule code are modified to use a new name if:
    - the "rename" command was used
    - or there's a collision with an existing, already-output rule
- otherwise the rule name remains the same.
- descriptions/tflags/etc are copied
- translations are also included in the output, if they exist in the translation directories

The current core ruleset becomes source files for the compiler. All the current rules are tagged as "manually promoted", right off the bat. We can gradually "demote" them to require decent performance, instead, as time goes on.

The major win? We get to preserve revision history.

Finally, an important point – the "source" files are still rules files, in other words they can be used as rules files for a mass-check or for the "spamassassin" script.