

# RulesProjBuildBot

## Rules Project: BuildBot

(part of [RulesProjectPlan](#))

For active rule development, it's important to get rules mass-checked very quickly.

Loren outlined the system used in SARE:

- rule developer sends mail to mailing list
- various other participants run scripts that automatically extract certain attachments posted to the list
- turn those into rules files
- lint them
- run a mass-check immediately with just the rules in that file
- post results including hit freqs and false positives matches
- masscheck requester asks for false positive verification based on report

For active rule development, this is obviously quite useful! If you can't run mass-check locally for whatever reason, it offers a way to do this using other people's corpora in almost-real-time.

[JustinMason](#): We had been considering a pretty direct copy of this. However I think I've thought of an alternative that improves on it, in my opinion...

## The Preflight Mass-checking BuildBot

We set up a [BuildBot](#). It is dedicated to performing mass-checks in the zone, in a chroot, solely on the rules found in the "rulesrc/sandbox" tree. It uses a small(ish) corpus so that mass-checks run quickly.

Mail-submitted rules: if a mail arrives from a known good source at a certain address, and contains a text/plain attachment, that attachment is extracted and passed through "spamassassin --lint". If it passes, it's checked in as "rulesrc/sandbox/mailed/latest.cf", triggering the preflight buildbot to start its mass-checks.

Results are visible immediately through the normal buildbot web UI. They are presented as 'freqs' output (as usual).

Because mail-submitted rules are OK, we have to be a little more careful than normal. For example, plugins are ignored in mail-submitted rules. Nothing happens if --lint fails. In addition, the preflight mass-check script runs with very paranoid ulimits, to trap CPU-load problems.

Why use [BuildBot](#)?

- Good web UI for "builds in progress"; you can monitor progress as it happens
- Designed to do "build whenever necessary, but not more than necessary", ie it solves the load issues caused by continuous integration
- Every mass-check output and every 'freqs' output will get a HTTP permalink, which allows side discussion to "point at" test results
- Integrated with SVN for version control and history tracking, so we can easily find test results that correspond to a mailed-in rules file
- Allows us to run mass-checks securely, in a buildbot slave running in a segmented chroot jail

In terms of results of mail-in rule checking; there's no automated check-ins into the "sandboxes" or "core" from this. Instead rules considered suitable for use are *manually* checked into the "sandbox" area by one of the committers who has privs to do that. With luck, they'll go into the core based on the automated testing described in [RulesProjStreamlining](#). This is a rule-QA system, not an alternative to "svn commit".

## Issues

*The mailed-in rules are checked into SVN temporarily, therefore will be publicly viewable.*

[JustinMason](#): In my opinion, we should go with this for now anyway. The use of SVN makes it hard to make it privately-viewable only, but at the same time offers some of the biggest advantages – such as keeping archive copies of the submitted rules after the mass-check has completed, providing network transparency, and ensuring that submitted rules get mass-checked (eventually).