First of all, in order to write efficient rules, it is important to understand how the Snort detection engine handles Snort rules. Snort 2.0 introduced the fast pattern matcher, which accelerates the detection process. The basic idea is that the detection engine organizes rules with a two-level labeling system to select some rules out of all loaded rules for detection (please note that Snort loads all rules on startup). The top-level label is a protocol-port number pair and the second-level label is a content (or uricontent) match.

The second-level label is chosen using the principle of first and longest non-negative content match. In the example below, for instance, Rule 1's second-level label is “writing”.

[Rule 1]

alert tcp $HOME_NET any -> $EXTERNAL_NET 80 (msg:”Bad Search”; content:”writing”; content:”bad”; content:”pig”; content:”rules”; content:!”snort”; sid:1;)

As of version 2.8.3, Snort supports the “fast_pattern” content modifier so that you can choose which content match will be used as the second-level label (e.g. Rule 2's second level label is “pig” thanks to this modifier).

[Rule 2]

alert tcp $HOME_NET any -> $EXTERNAL_NET 80 (msg:”Bad Search”; content:”writing”; content:”bad”; content:”pig”; fast_pattern; content:”rules”; content:!”snort”; sid:2;)

Let me explain the two-level labeling system in an even simpler fashion. Snort has big boxes, labeled with a protocol-port number pair, and these big boxes contain many bins that are labeled with fast content matches as well as an unlabeled bin. When Snort launches, it puts rules into the labeled bins if the rules have at least one content match, otherwise they are put into the unlabeled bin. Then, Snort uses these labeled containers to choose which rules are to be used to examine network traffic.

For example, Snort has a box with the “http/80” label, and the box has bins with “bad”, “good”, “writing”, “erasing”, and “pig” labels, as well as an unlabeled bin. Figure 1 shows which rules are in the labeled (or unlabeled) bins.

[Figure 1: Labeled boxes with labeled bins]

****** [“http/80”] **********

* “bad” bin: 12, 13, 14

* “good” bin: 10, 11

* “writing” bin: 1, 3

* “erasing” bin: 4, 8

* “pig” bin: 2

* unlabeled bin: 9, 15

****

If Alice searches for “writing bad pig rules” in a web browser, Snort will use the rules with sids 1, 2, 3, 9, 12, 13, 14, 15. Note that the rules in the unlabeled bin are always used, so it is important for a rule to have at least one content match to keep it out of the unlabeled bin. Also, it is more efficient for rules to have a long and specific fast pattern content match (second-level label) that is unlikely to be used often. For instance, rules in bin labeled “a” are much more likely to be used than those in a bin labeled “pneumonoultramicroscopicsilicovolcanoconiosis”.

After Snort chooses the rules to be used for detection, it parses those rules into option lists (as of Snort 2.8.2, Snort parses the rules into a tree structure to make detection even faster, but the principle of writing good rules remains the same). In the following example, Snort parses Rule 3 into Table 1.

[Rule 3]

alert tcp $EXTERNAL_NET 80 -> $HOME_NET any (msg:”Bad Result”; flow:to_client,established; flowbits:isset,bad_search; content:”bacon”; content:”|90 90 90 90 90|”; sid:3;)

[Table 1: Parsed Rule 3]

flow:to_client,established

flowbits:isset,bad_search

content:”bacon”

content:”|90 90 90 90 90|”

Snort goes through this option list in order, so it is a good idea to use less expensive checks such as dsize, flow and flowbits first to weed out mismatches and avoid further processing. After testing target traffic against this option list, the source/destination addresses and port numbers of a rule are checked.

Writing Snort rules is quite easy, but writing good Snort rules can be tricky. Try to keep these principles in mind, they will help with traffic processing.