One of the best features of Snort is its rule engine and language. Snort’s rule engine provides an extensive language that enables you to write your own rules, allowing you to extend it to meet the needs of your own network.

A Snort rule can be broken down into two basic parts, the rule header and options for the rule. The rule header contains the action to perform, the protocol that the rule applies to, and the source and destination addresses and ports. The rule options allow you to create a descriptive message to associate with the rule, as well as check a variety of other packet attributes by making use of Snort’s extensive library of plug-ins.

Here’s the general form of a Snort rule:

```
   action proto src_ip src_port direction dst_ip dst_port (options)
```

When a packet comes in, its source and destination IP addresses and ports are then compared to the rules in the ruleset. If any of them are applicable to the packet, then the options are compared to the packet. If all of these comparisons return a match, then the specified action is taken.

Snort provides several built-in actions that you can use when crafting your rules. To simply log the packet that matches a rule, use the `log` action. The `alert` action generates an alert using the method specified in your configuration file or on the command line, in addition to logging the packet. One nice feature is that you can have very general rules and then create exceptions by writing a rule that uses the `pass` action. This works especially well when you are using the rules distributed with Snort, but are frequently getting false positives for some of them. If this happens and it’s not a security risk to ignore them, you can simply write a `pass` rule for it.

The last two built-in rule actions are used together to dynamically modify Snort’s ruleset at runtime. These are the `activate` and `dynamic` actions. Rules that use the `dynamic` action are just like a `log` rule, except they will be considered only after they have been enabled by an `activate` rule. To accomplish this, Snort enforces the use of the `activates` and `activated_by` rule options in order to know what `dynamic` rules to enable once an `activate` rule has been triggered. In addition, `dynamic` rules are required to specify a `count` option in order for Snort to limit how many packets the rule will record.

For instance, if you wanted to start recording packets once an exploit of a SSH daemon on 192.168.1.21 was noticed, you could use a couple of rules similar to these:
activate tcp any any -> 192.168.1.21 22 (content:"/bin/sh"; activates:1; \msg:"Possible SSH buffer overflow"; )
dynamic tcp any any -> 192.168.1.21 22 (activated_by:1; count:100;)

These two rules aren’t completely foolproof, but if someone were to run an exploit with shell code against an SSH daemon, it would most likely send the string /bin/sh in the clear in order to spawn a shell on the system being attacked. In addition, since SSH is encrypted, strings like that wouldn’t be sent to the daemon under normal circumstances. Once the first rule is triggered, it will activate the second one, which will record 100 packets and then stop. This is useful, since you might be able to catch the intruder downloading or installing a root kit within those first few packets and be able to analyze the compromised system much more quickly.

You can also define custom rule actions, in addition to those that Snort has built-in. This is done with the ruletype keyword:

    ruletype redalert
    {
      type alert
      output alert_syslog: LOG_AUTH LOG_ALERT
      output database: log, mysql, user=snort dbname=snort host=localhost
    }

This custom rule action tells Snort that it behaves like the alert rule action, but specifies that the alerts should be sent to the syslog daemon, while the packet will be logged to a database. When defining a custom action, you can use any of Snort’s output plug-ins, just as you would if you were configuring them as your primary output method.

Snort’s detection engine supports several protocols. The proto field is used to specify what protocol your rule applies to. Valid values for this field are ip, icmp, tcp, and udp.

The next fields in a Snort rule are used to specify the source and destination IP addresses and ports of the packet, as well as the direction the packet is traveling. Snort can accept a single IP or a list of addresses. When specifying a list of IP address, you should separate each one with a comma and then enclose the list within square brackets, like this:

    [192.168.1.1,192.168.1.45,10.1.1.24]

When doing this, be careful not to use any whitespace. You can also specify ranges of IP addresses using CIDR notation, or even include CIDR ranges within lists. Snort also allows you to apply the logical NOT operator (!) to an IP address or CIDR range to specify that the rule should match all but that address or range of addresses.
**Write Your Own Snort Rules**

As with IP addresses, Snort can accept single ports as well as ranges. To specify a range, use a colon character to separate the lower bound from the upper bound. For example, if you wanted to specify all ports from 1 to 1024, you would do it like this:

1:1024

You can also apply the NOT operator to a port, and you can specify a range of ports without an upper or lower bound.

For instance, if you only wanted to examine ports greater than 1024, you would do it this way:

1024:

Similarly, you could specify ports less than 1024 by doing this:

:1024

If you do not care about the IP address or port, you can simply specify any.

Moving on, the direction field is used to tell Snort which IP address and port is the source and which pair is the destination. In earlier versions of Snort you could use either -> or <- to specify the direction. However, the <- operator was removed since you can make either one equivalent to the other by just switching the IP addresses and port numbers. Snort does have another direction operator in addition to ->, though. Specifying <-> as the direction tells Snort that you want the rule to apply bidirectionally. This is especially useful when using log rules or dynamic rules, since you can log both sides of the TCP stream rather than just one direction.

The next part of the rule includes the options. This part lets you specify many other attributes to check against. Each option is implemented through a Snort plug-in. When a rule that specifies an option is triggered, Snort will run through the option’s corresponding plug-in to perform the check against the packet. Snort has over 40 plug-ins—too many to cover in detail in this hack. Here are some of the more useful ones.

The single most useful option is msg. This option allows you to specify a custom message that will be logged in the alert when a packet matching the rule is detected. Without it, most alerts wouldn’t make much sense at first glance. This option takes a string enclosed in quotes as its argument.

For example, this specifies a logical message whenever Snort notices any traffic that is sent from 192.168.1.35:

```
alert tcp 192.168.1.35 any -> any any (msg:"Traffic from 192.168.1.35");
```

Be sure not to include any escaped quotes within the string. Snort’s parser is a simple one and does not support escaping characters.
Another useful option is `content`, which allows you to search a packet for a sequence of characters or hexadecimal values. If you are searching for a string, you can just put it in quotes. In addition, if you want it to do a case-insensitive search, you can add `nocase`; to the end of all your options. However, if you are looking for a sequence of hexadecimal digits, you must enclose them in `|` characters.

This rule will trigger when it sees the digit 0x90:

```
alert tcp any any -> any any (msg: "Possible exploit"; content: "|90|";)
```

This digit is the hexadecimal equivalent of the NOP instruction on the x86 architecture and is often seen in exploit code since it can be used to make buffer overflow exploits easier to write.

The `offset` and `depth` options can be used in conjunction with the `content` option to limit the searched portion of the data payload to a specific range of bytes.

If you wanted to limit content matches for NOP instructions to between bytes 40 and 75 of the data portion of a packet, you could modify the previously shown rule to look like this:

```
alert tcp any any -> any any (msg: "Possible exploit"; content: "|90|"; offset:40; depth:75;)
```

You can also match against packets that do not contain the specified sequence by prefixing it with a `!`. In addition, many shell code payloads can be very large compared to the normal amount of data carried in a packet sent to a particular service. You can check the size of a packet’s data payload by using the `dsize` option. This option takes a number as an argument. In addition, you can specify an upper bound by using the `<` operator, or you can choose a lower bound by using the `>` operator. Upper and lower bounds can be expressed with `<>`.

For example:

```
alert tcp any any -> any any (msg: "Possible exploit"; content: "|90|"; offset:40; depth:75; dsize: >6000;)
```

This modifies the previous rule to match only if the data payload’s size is greater than 6000 bytes, in addition to the other options criteria.

To check the TCP flags of a packet, Snort provides the `flags` option. This option is especially useful for detecting portscans that employ various invalid flag combinations.

For example, this rule will detect when the SYN and FIN flags are set at the same time:

```
alart any any -> any any (flags: SF,12; msg: "Possible SYN FIN scan";)
```
Write Your Own Snort Rules

Valid flags are S for SYN, F for FIN, R for RST, P for PSH, A for ACK, and U for URG. In addition, Snort lets you check the values of the two reserved flag bits. You can specify these by using either 1 or 2. You can also match packets that have no flags set by using 0. There are also several operators that the flags option will accept. You can prepend either a *, +, or ! to the flags, to match on all the flags plus any others, any of the flags, or only if none of the flags are set, respectively.

One of the best features of Snort is that it provides many plug-ins that can be used in the options field of a rule. The options discussed here should get you off to a good start. However, if you want to write more complex rules, consult Snort’s excellent rule documentation, which contains full descriptions and examples for each of Snort’s rule options. The Snort User’s Manual is available at http://www.snort.org/docs/writing_rules/.

This material has been adapted from Network Security Hacks by Andrew Lockhart, published by O'Reilly Media, Inc. Copyright O'Reilly Media, Inc., 2004. All rights reserved. To purchase this or other O'Reilly publications, click here.