Snort 3 User Manual

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

```
 ,,_     -*> Snort++ <*-
o"  )~   Version 3.0.0-a4 (Build 221) from 2.9.8-383
 ''''    By Martin Roesch & The Snort Team
         http://snort.org/contact#team
         Copyright (C) 2014-2016 Cisco and/or its affiliates. All rights reserved.
         Copyright (C) 1998-2013 Sourcefire, Inc., et al.
```

1 Overview

Snort 3.0 is an updated version of the Snort Intrusion Prevention System (IPS) which features a new design that provides a superset of Snort 2.X functionality with better throughput, detection, scalability, and usability. Some of the key features of Snort 3.0 are:

- Support multiple packet processing threads
- Use a shared configuration and attribute table
- Autodetect services for portless configuration
- Modular design
- Plugin framework with over 200 plugins
- More scalable memory profile
- LuaJIT configuration, loggers, and rule options
- Hyperscan support
- Rewritten TCP handling
- New rule parser and syntax
- Service rules like alert http
- Rule "sticky" buffers

- Way better SO rules
- New HTTP inspector
- New performance monitor
- New time and space profiling
- New latency monitoring and enforcement
- Piglets to facilitate component testing
- Inspection Events
- Automake and Cmake
- Autogenerate reference documentation

Additional features are on the road map:

- Use a shared network map
- Support hardware offload for fast pattern acceleration
- Provide support for DPDK and ODP
- Support pipelining of packet processing
- Support proxy mode
- Multi-tennant support
- Incremental reload
- New serialization of perf data and events
- Enhanced rule processing
- Windows support
- Anomaly detection
- and more!

The remainder of this section provides a high level survey of the inputs, processing, and outputs available with Snort 3.0.

Snort++ is the project that is creating Snort 3.0. In this manual "Snort" or "Snort 3" refers to the 3.0 version and earlier versions will be referred to as "Snort 2" where the distinction is relevant.

1.1 First Steps

Snort can be configured to perform complex packet processing and deep packet inspection but it is best start simply and work up to more interesting tasks. Snort won't do anything you didn't specifically ask it to do so it is safe to just try things out and see what happens. Let's start by just running Snort with no arguments:

```
$ snort
```

That will output usage information including some basic help commands. You should run all of these commands now to see what is available:

```
$ snort -V
$ snort -?
$ snort --help
```

Note that Snort has extensive command line help available so if anything below isn't clear, there is probably a way to get the exact information you need from the command line.

Now let's examine the packets in a capture file (pcap):

```
$ snort -r a.pcap
```

Snort will decode and count the packets in the file and output some statistics. Note that the output excludes non-zero numbers so it is easy to see what is there.

You may have noticed that there are command line options to limit the number of packets examined or set a filter to select particular packets. Now is a good time to experiment with those options.

If you want to see details on each packet, you can dump the packets to console like this:

```
$ snort -r a.pcap -L dump
```

Add the -d option to see the TCP and UDP payload. Now let's switch to live traffic. Replace eth0 in the below command with an available network interface:

```
$ snort -i eth0 -L dump
```

Unless the interface is taken down, Snort will just keep running, so enter Control-C to terminate or use the -n option to limit the number of packets.

Generally it is better to capture the packets for later analysis like this:

```
$ snort -i eth0 -L pcap -n 10
```

Snort will write 10 packets to log.pcap.# where # is a timestamp value. You can read these back with -r and dump to console or pcap with -L. You get the idea.

Note that you can do similar things with other tools like tcpdump or Wireshark however these commands are very useful when you want to check your Snort setup.

The examples above use the default pcap DAQ. Snort supports non-pcap interfaces as well via the DAQ (data acquisition) library. Other DAQs provide additional functionality such as inline operation and/or higher performance. There are even DAQs that support raw file processing (ie without packets), socket processing, and plain text packets. To load external DAQ libraries and see available DAQs or select a particular DAQ use one of these commands:

```
$ snort --daq-dir <path> --daq-list
$ snort --daq-dir <path> --daq <type>
```

Be sure to put the --daq-dir option ahead of the --daq-list option or the external DAQs won't appear in the list.

To leverage intrusion detection features of Snort you will need to provide some configuration details. The next section breaks down what must be done.

1.2 Configuration

Effective configuration of Snort is done via the environment, command line, a Lua configuration file, and a set of rules.

Note that backwards compatibility with Snort 2 was sacrificed to obtain new and improved functionality. While Snort 3 leverages some of the Snort 2 code base, a lot has changed. The configuration of Snort 3 is done with Lua, so your old conf won't work as is. Rules are still text based but with syntax tweaks, so your 2.X rules must be fixed up. However, snort2lua will help you convert your conf and rules to the new format.

1.2.1 Environment

LUA_PATH must be set based on your install:

```
LUA_PATH=$install_prefix/include/snort/lua/\?.lua\;\;
```

SNORT_LUA_PATH must be set to load auxiliary configuration files if you use the default snort.lua. For example:

```
export SNORT_LUA_PATH=$install_prefix/etc/snort
```

1.2.2 Command Line

A simple command line might look like this:

```
snort -c snort.lua -R cool.rules -r some.pcap -A cmg
```

To understand what that does, you can start by just running snort with no arguments by running snort --help. Help for all configuration and rule options is available via a suitable command line. In this case:

-c snort.lua is the main configuration file. This is a Lua script that is executed when loaded.

-R cool.rules contains some detection rules. You can write your own or obtain them from Talos (native 3.0 rules are not yet available from Talos so you must convert them with snort2lua). You can also put your rules directly in your configuration file.

-r some.pcap tells Snort to read network traffic from the given packet capture file. You could instead use -i eth0 to read from a live interface. There many other options available too depending on the DAQ you use.

-A cmg says to output intrusion events in "cmg" format, which has basic header details followed by the payload in hex and text.

Note that you add to and/or override anything in your configuration file by using the --lua command line option. For example:

```
--lua 'ips = { enable_builtin_rules = true }'
```

will load the built-in decoder and inspector rules. In this case, ips is overwritten with the config you see above. If you just want to change the config given in your configuration file you would do it like this:

```
--lua 'ips.enable_builtin_rules = true'
```

1.2.3 Configuration File

The configuration file gives you complete control over how Snort processes packets. Start with the default snort.lua included in the distribution because that contains some key ingredients. Note that most of the configurations look like:

```
stream = { }
```

This means enable the stream module using internal defaults. To see what those are, you could run:

```
snort --help-config stream
```

Snort is organized into a collection of builtin and plugin modules. If a module has parameters, it is configured by a Lua table of the same name. For example, we can see what the active module has to offer with this command:

```
$ snort --help-module active

What: configure responses

Type: basic
```

```
Configuration:

int active.attempts = 0: number of TCP packets sent per response (with
varying sequence numbers) { 0:20 }

string active.device: use 'ip' for network layer responses or 'eth0' etc
for link layer

string active.dst_mac: use format '01:23:45:67:89:ab'

int active.max_responses = 0: maximum number of responses { 0: }

int active.min_interval = 255: minimum number of seconds between
responses { 1: }
```

This says active is a basic module that has several parameters. For each, you will see:

```
type module.name = default: help { range }
```

For example, the active module has a max_responses parameter that takes non-negative integer values and defaults to zero. We can change that in Lua as follows:

```
active = { max_responses = 1 }
```

or:

```
active = { }
active.max_responses = 1
```

If we also wanted to limit retries to at least 5 seconds, we could do:

```
active = { max_responses = 1, min_interval = 5 }
```

1.2.4 Rules

Rules determine what Snort is looking for. They can be put directly in your Lua configuration file with the ips module, on the command line with --lua, or in external files. Generally you will have many rules obtained from various sources such as Talos and loading external files is the way to go so we will summarize that here. Add this to your Lua configuration:

```
ips = { include = 'rules.txt' }
```

to load the external rules file named rules.txt. You can only specify one file this way but rules files can include other rules files with the include statement. In addition you can load rules like:

```
$ sort -c snort.lua -R rules.txt
```

You can use both approaches together.

1.2.5 Converting Your 2.X Configuration

If you have a working 2.X configuration snort2lua makes it easy to get up and running with Snort 3. This tool will convert your configuration and/or rules files automatically. You will want to clean up the results and double check that it is doing exactly what you need.

```
snort2lua -c snort.conf
```

The above command will generate snort.lua based on your 2.X configuration. For more information and options for more sophisticated use cases, see the Snort2Lua section later in the manual.

1.3 Output

Snort can produce quite a lot of data. In the following we will summarize the key aspects of the core output types. Additional data such as from appid is covered later.

1.3.1 Basic Statistics

At shutdown, Snort will output various counts depending on configuration and the traffic processed. Generally, you may see:

- Packet Statistics - this includes data from the DAQ and decoders such as the number of packets received and number of UDP packets.
- Module Statistics - each module tracks activity via a set of peg counts that indicate how many times something was observed or performed. This might include the number of HTTP GET requests processed and the number of TCP reset packets trimmed.
- File Statistics - look here for a breakdown of file type, bytes, signatures.
- Summary Statistics - this includes total runtime for packet processing and the packets per second. Profiling data will appear here as well if configured.

Note that only the non-zero counts are output. Run this to see the available counts:

```
$ snort --help-counts
```

1.3.2 Alerts

If you configured rules, you will need to configure alerts to see the details of detection events. Use the -A option like this:

```
$ snort -c snort.lua -r a.pcap -A cmg
```

There are many types of alert outputs possible. Here is a brief list:

- -A cmg is the same as -A fast -d -e and will show information about the alert along with packet headers and payload.
- -A u2 is the same as -A unified2 and will log events and triggering packets in a binary file that you can feed to other tools for post processing. Note that Snort 3 does not provide the raw packets for alerts on PDUs; you will get the actual buffer that alerted.
- -A csv will output various fields in comma separated value format. This is entirely customizable and very useful for pcap analysis.

To see the available alert types, you can run this command:

```
$ snort --list-plugins | grep logger
```

1.3.3 Files and Paths

Note that output is specific to each packet thread. If you run 4 packet threads with u2 output, you will get 4 different u2 files. The basic structure is:

```
<logdir>/[<run_prefix>][<id#>][<X>]<name>
```

where:

- logdir is set with -l and defaults to ./

- run_prefix is set with --run-prefix else not used
- id# is the packet thread number that writes the file; with one packet thread, id# (zero) is omitted without --id-zero
- X is / if you use --id-subdir, else _ if id# is used
- name is based on module name that writes the file

Additional considerations:

- There is no way to explicitly configure a full path to avoid issues with multiple packet threads.
- All text mode outputs default to stdout

1.3.4 Performance Statistics

Still more data is available beyond the above.

- By configuring the perf_monitor module you can capture a configurable set of peg counts during runtime. This is useful to feed to an external program so you can see what is happening without stopping Snort.
- The profiler module allows you to track time and space used by module and rules. Use this data to tune your system for best performance. The output will show up under Summary Statistics at shutdown.

2 Concepts

This section provides background on essential aspects of Snort's operation.

2.1 Terminology

- **basic module**: a module integrated into Snort that does not come from a plugin.
- **binder**: inspector that maps configuration to traffic
- **builtin rules**: codec and inspector rules for anomalies detected internally.
- **codec**: short for coder / decoder. These plugins are used for basic protocol decoding, anomaly detection, and construction of active responses.
- **data module**: an adjunct configuration plugin for use with certain inspectors.
- **dynamic rules**: plugin rules loaded at runtime. See SO rules.
- **fast pattern**: the content in an IPS rule that must be found by the search engine in order for a rule to be evaluated.
- **fast pattern matcher**: see search engine.
- **hex**: a type of protocol magic that the wizard uses to identify binary protocols.
- **inspector**: plugin that processes packets (similar to the Snort 2 preprocessor)
- **IPS**: intrusion prevention system, like Snort.
- **IPS action**: plugin that allows you to perform custom actions when events are generated. Unlike loggers, these are invoked before thresholding and can be used to control external agents or send active responses.
- **IPS option**: this plugin is the building blocks of IPS rules.
- **logger**: a plugin that performs output of events and packets. Events are thresholded before reaching loggers.

- **module**: the user facing portion of a Snort component. Modules chiefly provide configuration parameters, but may also provide commands, builtin rules, profiling statistics, peg counts, etc. Note that not all modules are plugins and not all plugins have modules.
- **peg count**: the number of times a given event or condition occurs.
- **plugin**: one of several types of software components that can be loaded from a dynamic library when Snort starts up. Some plugins are coupled with the main engine in such a way that they must be built statically, but a newer version can be loaded dynamically.
- **search engine**: a plugin that performs multipattern searching of packets and payload to find rules that should be evaluated. There are currently no specific modules, although there are several search engine plugins. Related configuration is done with the basic detection module. Aka fast pattern matcher.
- **SO rule**: a IPS rule plugin that performs custom detection that can't be done by a text rule. These rules typically do not have associated modules. SO comes from shared object, meaning dynamic library.
- **spell**: a type of protocol magic that the wizard uses to identify ASCII protocols.
- **text rule**: a rule loaded from the configuration that has a header and body. The header specifies action, protocol, source and destination IP addresses and ports, and direction. The body specifies detection and non-detection options.
- **wizard**: inspector that applies protocol magic to determine which inspectors should be bound to traffic absent a port specific binding. See hex and spell.

2.2 Modules

Modules are the building blocks of Snort. They encapsulate the types of data that many components need including parameters, peg counts, profiling, builtin rules, and commands. This allows Snort to handle them generically and consistently. You can learn quite a lot about any given module from the command line. For example, to see what stream_tcp is all about, do this:

```
$ snort --help-config stream_tcp
```

Modules are configured using Lua tables with the same name. So the stream_tcp module is configured with defaults like this:

```
stream_tcp = { }
```

The earlier help output showed that the default session tracking timeout is 30 seconds. To change that to 60 seconds, you can configure it this way:

```
stream_tcp = { session_timeout = 60 }
```

Or this way:

```
stream_tcp = { }
stream_tcp.session_timeout = 60
```

More on parameters is given in the next section.

Other things to note about modules:

- Shutdown output will show the non-zero peg counts for all modules. For example, if stream_tcp did anything, you would see the number of sessions processed among other things.
- Providing the builtin rules allows the documentation to include them automatically and also allows for autogenerating the rules at startup.
- Only a few module provide commands at this point, most notably the snort module.

2.3 Parameters

Parameters are given with this format:

```
type name = default: help { range }
```

The following types are used:

- **addr**: any valid IP4 or IP6 address or CIDR
- **addr_list**: a space separated list of addr values
- **bit_list**: a list of consecutive integer values from 1 to the range maximum
- **bool**: true or false
- **dynamic**: a select type determined by loaded plugins
- **enum**: a string selected from the given range
- **implied**: an IPS rule option that takes no value but means true
- **int**: a whole number in the given range
- **ip4**: an IP4 address or CIDR
- **mac**: an ethernet address with the form 01:02:03:04:05:06
- **multi**: one or more space separated strings from the given range
- **port**: an int in the range 0:65535 indicating a TCP or UDP port number
- **real**: a real number in the given range
- **select**: a string selected from the given range
- **string**: any string with no more than the given length, if any

The parameter name may be adorned in various ways to indicate additional information about the type and use of the parameter:

- For Lua configuration (not IPS rules), if the name ends with [] it is a list item and can be repeated.
- For IPS rules only, names starting with ~ indicate positional parameters. The names of such parameters do not appear in the rule.
- IPS rules may also have a wild card parameter, which is indicated by a *. Only used for metadata that Snort ignores.
- The snort module has command line options starting with a -.

Some additional details to note:

- Table and variable names are case sensitive; use lower case only.
- String values are case sensitive too; use lower case only.
- Numeric ranges may be of the form low:high where low and high are bounds included in the range. If either is omitted, there is no hard bound. E.g. 0: means any x where x >= 0.
- Strings may have a numeric range indicating a length limit; otherwise there is no hard limit.
- bit_list is typically used to store a set of byte, port, or VLAN ID values.

2.4 Plugins

Snort uses a variety of plugins to accomplish much of its processing objectives, including:

- Codec - to decode and encode packets
- Inspector - like Snort 2 preprocessors, for normalization, etc.
- IpsOption - for detection in Snort rules
- IpsAction - for custom actions
- Logger - for handling events
- Mpse - for fast pattern matching
- So - for dynamic rules

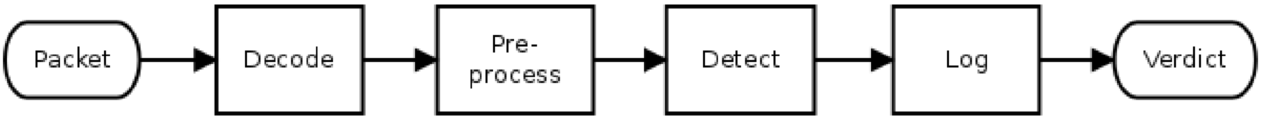
The power of plugins is that they have a very focused purpose and can be created with relative ease. For example, you can extend the rule language by writing your own IpsOption and it will plug in and function just like existing options. The extra directory has examples of each type of plugin.

Most plugins can be built statically or dynamically. By default they are all static. There is no difference in functionality between static or dynamic plugins but the dynamic build generates a slightly lighter weight binary. Either way you can add dynamic plugins with --plugin-path and newer versions will replace older versions, even when built statically.

A single dynamic library may contain more than one plugin. For example, an inspector will typically be packaged together with any associated rule options.

2.5 Operation

Snort is a signature-based IPS, which means that as it receives network packets it reassembles and normalizes the content so that a set of rules can be evaluated to detect the presence of any significant conditions that merit further action. A rough processing flow is as follows:



The steps are:

1. Decode each packet to determine the basic network characteristics such as source and destination addresses and ports. A typical packet might have ethernet containing IP containing TCP containing HTTP (ie eth:ip:tcp:http). The various encapsulating protocols are examined for sanity and anomalies as the packet is decoded. This is essentially a stateless effort.
2. Preprocess each decoded packet using accumulated state to determine the purpose and content of the innermost message. This step may involve reordering and reassembling IP fragments and TCP segments to produce the original application protocol data unit (PDU). Such PDUs are analyzed and normalized as needed to support further processing.
3. Detection is a two step process. For efficiency, most rules contain a specific content pattern that can be searched for such that if no match is found no further processing is necessary. Upon start up, the rules are compiled into pattern groups such that a single, parallel search can be done for all patterns in the group. If any match is found, the full rule is examined according to the specifics of the signature.
4. The logging step is where Snort saves any pertinent information resulting from the earlier steps. More generally, this is where other actions can be taken as well such as blocking the packet.

2.5.1 Snort 2 Processing

The preprocess step in Snort 2 is highly configurable. Arbitrary preprocessors can be loaded dynamically at startup, configured in snort.conf, and then executed at runtime. Basically, the preprocessors are put into a list which is iterated for each packet. Recent versions have tweaked the list handling some, but the same basic architecture has allowed Snort 2 to grow from a sniffer, with no preprocessing, to a full-fledged IPS, with lots of preprocessing.

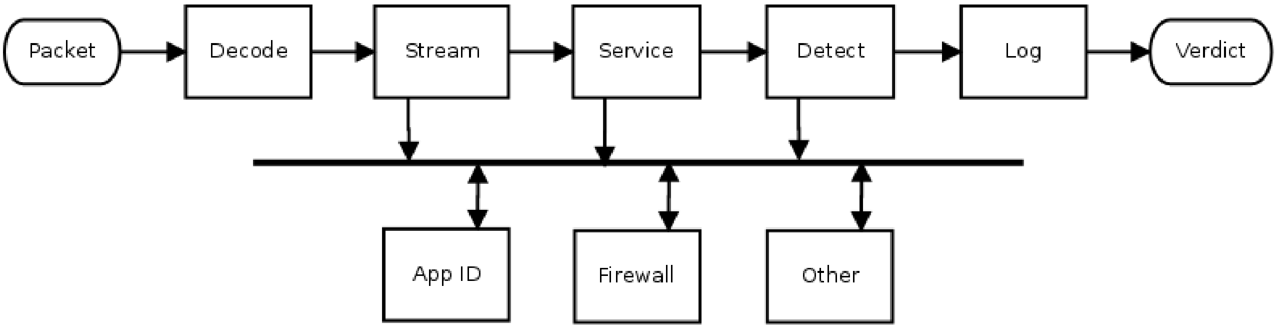
While this "list of plugins" approach has considerable flexibility, it hampers future development when the flow of data from one preprocessor to the next depends on traffic conditions, a common situation with advanced features like application identification. In this case, a preprocessor like HTTP may be extracting and normalizing data that ultimately is not used, or appID may be repeatedly checking for data that is just not available.

Callbacks help break out of the preprocess straitjacket. This is where one preprocessor supplies another with a function to call when certain data is available. Snort has started to take this approach to pass some HTTP and SIP preprocessor data to appID. However, it remains a peripheral feature and still requires the production of data that may not be consumed.

2.5.2 Snort 3 Processing

One of the goals of Snort 3 is to provide a more flexible framework for packet processing by implementing an event-driven approach. Another is to produce data only when needed to minimize expensive normalizations. However, the basic packet processing provides very similar functionality.

The basic processing steps Snort 3 takes are similar to Snort 2 as seen in the following diagram. The preprocess step employs specific inspector types instead of a generalized list, but the basic procedure includes stateless packet decoding, TCP stream reassembly, and service specific analysis in both cases. (Snort 3 provides hooks for arbitrary inspectors, but they are not central to basic flow processing and are not shown.)



However, Snort 3 also provides a more flexible mechanism than callback functions. By using inspection events, it is possible for an inspector to supply data that other inspectors can process. This is known as the observer pattern or publish-subscribe pattern.

Note that the data is not actually published. Instead, access to the data is published, and that means that subscribers can access the raw or normalized version(s) as needed. Normalizations are done only on the first access, and subsequent accesses get the previously normalized data. This results in just in time (JIT) processing.

A basic example of this in action is provided by the extra data_log plugin. It is a passive inspector, ie it does nothing until it receives the data it subscribed for (*other* in the above diagram). By adding the following to your snort.lua configuration, you will get a simple URI logger.

```
data_log = { key = 'http_raw_uri' }
```

Inspection events coupled with pluggable inspectors provide a very flexible framework for implementing new features. And JIT buffer stuffers allow Snort to work smarter, not harder. These capabilities will be leveraged more and more as Snort development continues.

2.6 Rules

Rules tell Snort how to detect interesting conditions, such as an attack, and what to do when the condition is detected. Here is an example rule:

```
alert tcp any any -> 192.168.1.1 80 ( msg:"A ha!"; content:"attack"; sid:1; )
```

The structure is:

```
action proto source dir dest ( body )
```

Where:

action - tells Snort what to do when a rule "fires", ie when the signature matches. In this case Snort will log the event. It can also do thing like block the flow when running inline.

proto - tells Snort what protocol applies. This may be ip, icmp, tcp, udp, http, etc.

source - specifies the sending IP address and port, either of which can be the keyword any, which is a wildcard.

dir - must be either unidirectional as above or bidirectional indicated by <>.

dest - similar to source but indicates the receiving end.

body - detection and other information contained in parenthesis.

There are many rule options available to construct as sophisticated a signature as needed. In this case we are simply looking for the "attack" in any TCP packet. A better rule might look like this:

```
alert http
(
    msg:"Gotcha!";
    flow:established, to_server;
    http_uri:"attack";
    sid:2;
)
```

Note that these examples have a sid option, which indicates the signature ID. In general rules are specified by gid:sid:rev notation, where gid is the generator ID and rev is the revision of the rule. By default, text rules are gid 1 and shared-object (SO) rules are gid 3. The various components within Snort that generate events have 1XX gids, for example the decoder is gid 116. You can list the internal gids and sids with these commands:

```
$ snort --list-gids
$ snort --list-builtin
```

For details on these and other options, see the reference section.

2.7 Pattern Matching

Snort evaluates rules in a two-step process which includes a fast pattern search and full evaluation of the signature. More details on this process follow.

2.7.1 Rule Groups

When Snort starts or reloads configuration, rules are grouped by protocol, port and service. For example, all TCP rules using the HTTP_PORTS variable will go in one group and all service HTTP rules will go in another group. These rule groups are compiled into multipattern search engines (MPSE) which are designed to search for all patterns with just a single pass through a given packet or buffer. You can select the algorithm to use for fast pattern searches with search_engine.search_method which defaults to *ac_bnfa*, which balances speed and memory. For a faster search at the expense of significantly more memory, use *ac_full*. For best performance and reasonable memory, download the hyperscan source from Intel.

2.7.2 Fast Patterns

Fast patterns are content strings that have the fast_pattern option or which have been selected by Snort automatically to be used as a fast pattern. Snort will by default choose the longest pattern in the rule since that is likely to be most unique. That is not always the case so add fast_pattern to the appropriate content option for best performance. The ideal fast pattern is one which, if found, is very likely to result in a rule match. Fast patterns that match frequently for unrelated traffic will cause Snort to work hard with little to show for it.

Certain contents are not eligible to be used as fast patterns. Specifically, if a content is negated, then if it is also relative to another content, case sensitive, or has non-zero offset or depth, then it is not eligible to be used as a fast pattern.

2.7.3 Rule Evaluation

For each fast pattern match, the corresponding rule(s) are evaluated left-to-right. Rule evaluation requires checking each detection option in a rule and is a fairly costly process which is why fast patterns are so important. Rule evaluation aborts on the first non-matching option.

When rule evaluation takes place, the fast pattern may or may not need to be searched for a second time. Note that this differs from Snort 2 which provided the fast_pattern:only option to designate such cases. This was removed because it is difficult for the rule writer get it right.

3 Tutorial

The section will walk you through building and running Snort. It is not exhaustive but, once you master this material, you should be able to figure out more advanced usage.

3.1 Dependencies

Required:

- autotools or cmake to build from source
- daq from http://www.snort.org for packet IO
- g++ >= 4.8 or other recent C++11 compiler
- dnet from https://github.com/dugsong/libdnet.git for network utility functions
- hwloc from https://www.open-mpi.org/projects/hwloc/ for CPU affinity management
- LuaJIT from http://luajit.org for configuration and scripting
- OpenSSL from https://www.openssl.org/source/ for SHA and MD5 file signatures, the protected_content rule option, and SSL service detection
- pcap from http://www.tcpdump.org for tcpdump style logging
- pcre from http://www.pcre.org for regular expression pattern matching
- pkgconfig from https://www.freedesktop.org/wiki/Software/pkg-config/ to locate build dependencies
- zlib from http://www.zlib.net for decompression (>= 1.2.8 recommended)

Optional:

- lzma >= 5.1.2 from http://tukaani.org/xz/ for decompression of SWF and PDF files
- hyperscan from https://github.com/01org/hyperscan to build new and improved regex and (coming soon) fast pattern support

- cpputest from http://cpputest.github.io to run additional unit tests with make check
- asciidoc from http://www.methods.co.nz/asciidoc/ to build the HTML manual
- dblatex from http://dblatex.sourceforge.net to build the pdf manual (in addition to asciidoc)
- w3m from http://sourceforge.net/projects/w3m/ to build the plain text manual
- source-highlight from http://www.gnu.org/software/src-highlite/ to generate the dev guide
- safec from https://sourceforge.net/projects/safeclib/ for runtime bounds checks on certain legacy C-library calls.

3.2 Building

- Optionally built features are listed in the reference section.
- Create an install path:

  ```
  export my_path=/path/to/snorty
  mkdir -p $my_path
  ```

- If you are using a github clone with autotools, do this:

  ```
  autoreconf -isvf
  ```

- Now do one of the following:

  a. To build with autotools, simply do the usual from the top level directory:

     ```
     ./configure --prefix=$my_path
     make -j 8
     make install
     ```

  b. To build with cmake and make, run configure_cmake.sh. It will automatically create and populate a new subdirectory named *build*.

     ```
     ./configure_cmake.sh --prefix=$my_path
     cd build
     make -j 8
     make install
     ln -s $my_path/conf $my_path/etc
     ```

  c. You can also specify a cmake project generator:

     ```
     ./configure_cmake.sh --generator=Xcode --prefix=$my_path
     ```

  d. Or use ccmake directly to configure and generate from an arbitrary build directory like one of these:

     ```
     ccmake -G Xcode /path/to/Snort++/tree
     open snort.xcodeproj

     ccmake -G "Eclipse CDT4 - Unix Makefiles" /path/to/Snort++/tree
     run eclipse and do File > Import > Existing Eclipse Project
     ```

- To build with g++ on OS X where clang is installed, do this first:

  ```
  export CXX=g++
  ```

3.3 Running

First set up the environment:

```
export LUA_PATH=$my_path/include/snort/lua/\?.lua\;\;
export SNORT_LUA_PATH=$my_path/etc/snort/
```

Then give it a go:

- Get some help:

  ```
  $my_path/bin/snort --help
  $my_path/bin/snort --help-module suppress
  $my_path/bin/snort --help-config | grep thread
  ```

- Examine and dump a pcap:

  ```
  $my_path/bin/snort -r <pcap>
  $my_path/bin/snort -L dump -d -e -q -r <pcap>
  ```

- Verify config, with or w/o rules:

  ```
  $my_path/bin/snort -c $my_path/etc/snort/snort.lua
  $my_path/bin/snort -c $my_path/etc/snort/snort.lua -R $my_path/etc/snort/sample. ↩
      rules
  ```

- Run IDS mode. To keep it brief, look at the first n packets in each file:

  ```
  $my_path/bin/snort -c $my_path/etc/snort/snort.lua -R $my_path/etc/snort/sample. ↩
      rules \
       -r <pcap> -A alert_test -n 100000
  ```

- Let’s suppress 1:2123. We could edit the conf or just do this:

  ```
  $my_path/bin/snort -c $my_path/etc/snort/snort.lua -R $my_path/etc/snort/sample. ↩
      rules \
       -r <pcap> -A alert_test -n 100000 --lua "suppress = { { gid = 1, sid = 2123 } ↩
           }"
  ```

- Go whole hog on a directory with multiple packet threads:

  ```
  $my_path/bin/snort -c $my_path/etc/snort/snort.lua -R $my_path/etc/snort/sample. ↩
      rules \
       --pcap-filter \*.pcap --pcap-dir <dir> -A alert_fast -n 1000 --max-packet- ↩
          threads 8
  ```

For more examples, see the usage section.

3.4 Tips

One of the goals of Snort 3 is to make it easier to configure your sensor. Here is a summary of tips and tricks you may find useful.

General Use

- Snort tries hard not to error out too quickly. It will report multiple semantic errors.

- Snort always assumes the simplest mode of operation. Eg, you can omit the -T option to validate the conf if you don’t provide a packet source.
- Warnings are not emitted unless --warn-* is specified. --warn-all enables all warnings, and --pedantic makes such warnings fatal.
- You can process multiple sources at one time by using the -z or --max-threads option.
- To make it easy to find the important data, zero counts are not output at shutdown.
- Load plugins from the command line with --plugin-path /path/to/install/lib.
- You can process multiple sources at one time by using the -z or --max-threads option.
- Unit tests are configured with --enable-unit-tests. They can then be run with snort --catch-test [tags]|all.

Lua Configuration

- Configure the wizard and default bindings will be created based on configured inspectors. No need to explicitly bind ports in this case.
- You can override or add to your Lua conf with the --lua command line option.
- The Lua conf is a live script that is executed when loaded. You can add functions, grab environment variables, compute values, etc.
- You can also rename symbols that you want to disable. For example, changing normalizer to Xnormalizer (an unknown symbol) will disable the normalizer. This can be easier than commenting in some cases.
- By default, symbols unknown to Snort are silently ignored. You can generate warnings for them with --warn-unknown. To ignore such symbols, export them in the environment variable SNORT_IGNORE.

Writing and Loading Rules

Snort rules allow arbitrary whitespace. Multi-line rules make it easier to structure your rule for clarity. There are multiple ways to add comments to your rules:

- The # character starts a comment to end of line. In addition, all lines between #begin and #end are comments.
- The rem option allows you to write a comment that is conveyed with the rule.
- C style multi-line comments are allowed, which means you can comment out portions of a rule while testing it out by putting the options between /* and */.

There are multiple ways to load rules too:

- Set ips.rules or ips.include.
- include statements can be used in rules files.
- Use -R to load a rules file.
- Use --stdin-rules with command line redirection.
- Use --lua to specify one or more rules as a command line argument.

Output Files

To make it simple to configure outputs when you run with multiple packet threads, output files are not explicitly configured. Instead, you can use the options below to format the paths:

```
<logdir>/[<run_prefix>][<id#>][<X>]<name>
```

- logdir is set with -l and defaults to ./
- run_prefix is set with --run-prefix else not used
- id# is the packet thread number that writes the file; with one packet thread, id# (zero) is omitted without --id-zero
- X is / if you use --id-subdir, else _ if id# is used
- name is based on module name that writes the file
- all text mode outputs default to stdout

3.5 Help

```
Snort has several options to get more help:

-? list command line options (same as --help)
--help this overview of help
--help-commands [<module prefix>] output matching commands
--help-config [<module prefix>] output matching config options
--help-counts [<module prefix>] output matching peg counts
--help-module <module> output description of given module
--help-modules list all available modules with brief help
--help-plugins list all available plugins with brief help
--help-options [<option prefix>] output matching command line options
--help-signals dump available control signals
--list-buffers output available inspection buffers
--list-builtin [<module prefix>] output matching builtin rules
--list-gids [<module prefix>] output matching generators
--list-modules [<module type>] list all known modules
--list-plugins list all known modules
--show-plugins list module and plugin versions

--help* and --list* options preempt other processing so should be last on the
command line since any following options are ignored.  To ensure options like
--markup and --plugin-path take effect, place them ahead of the help or list
options.

Options that filter output based on a matching prefix, such as --help-config
won't output anything if there is no match.  If no prefix is given, everything
matches.

Report bugs to bugs@snort.org.
```

3.6 Common Errors

FATAL: snort_config is required

- add this line near top of file:

  ```
  require('snort_config')
  ```

PANIC: unprotected error in call to Lua API (cannot open snort_defaults.lua: No such file or directory)

- export SNORT_LUA_PATH to point to any dofiles

ERROR can't find xyz

- if xyz is the name of a module, make sure you are not assigning a scalar where a table is required (e.g. xyz = 2 should be xyz = { }).

ERROR can't find x.y

- module x does not have a parameter named y. check --help-module x for available parameters.

ERROR invalid x.y = z

- the value z is out of range for x.y. check --help-config x.y for the range allowed.

ERROR: x = { y = z } is in conf but is not being applied

- make sure that x = { } isn't set later because it will override the earlier setting. same for x.y.

FATAL: can't load lua/errors.lua: lua/errors.lua:68: = expected near ';'

- this is a syntax error reported by Lua to Snort on line 68 of errors.lua.

ERROR: rules(2) unknown rule keyword: find.

- this was due to not including the --script-path.

WARNING: unknown symbol x

- if you any variables, you can squelch such warnings by setting them in an environment variable SNORT_IGNORE. to ignore x, y, and z:

  ```
  export SNORT_IGNORE="x y z"
  ```

3.7 Gotchas

- A nil key in a table will not caught. Neither will a nil value in a table. Neither of the following will cause errors, nor will they actually set http_server.post_depth:

  ```
  http_server = { post_depth }
  http_server = { post_depth = undefined_symbol }
  ```

- It is not an error to set a value multiple times. The actual value applied may not be the last in the table either. It is best to avoid such cases.

  ```
  http_server =
  {
      post_depth = 1234,
      post_depth = 4321
  }
  ```

- Snort can't tell you the exact filename or line number of a semantic error but it will tell you the fully qualified name.
- The dump DAQ will not work with multiple threads unless you use --daq-var file=/dev/null. This will be fixed in at some point to use the Snort log directory, etc.
- Variables are currently processed in an order determined by the Lua hash table which is effectively random. That means you will need to use Lua string concatenation to ensure Snort doesn't try to use a variable before it is defined (even when it is defined ahead of use in the file):

```
-- this may fail:
MY_SERVERS = [[ 172.20.0.0/16 172.21.0.0/16 ]]
EXTERNAL_NET = '!$MY_SERVERS'

-- this will work:
MY_SERVERS = [[ 172.20.0.0/16 172.21.0.0/16 ]]
EXTERNAL_NET = '!' .. MY_SERVERS
```

- configure will use clang `by default if it is installed.To compile with g` instead:

  ```
  export CXX=g++
  ```

- If you build with hyperscan on OS X and see:

  ```
  dyld: Library not loaded: @rpath/libhs.4.0.dylib

  when you try to run src/snort, export DYLD_LIBRARY_PATH with the path to
  libhs.  You can also do:

  install_name_tool -change @rpath/libhs.4.0.dylib \
      /path-to/libhs.4.0.dylib src/snort
  ```

4 Usage

For the following examples "$my_path" is assumed to be the path to the Snort install directory. Additionally, it is assumed that "$my_path/bin" is in your PATH.

4.1 Environment

LUA_PATH is used directly by Lua to load and run required libraries. SNORT_LUA_PATH is used by Snort to load supplemental configuration files.

```
export LUA_PATH=$my_path/include/snort/lua/\?.lua\;\;
export SNORT_LUA_PATH=$my_path/etc/snort
```

4.2 Help

Print the help summary:

```
snort --help
```

Get help on a specific module ("stream", for example):

```
snort --help-module stream
```

Get help on the "-A" command line option:

```
snort --help-options A
```

Grep for help on threads:

```
snort --help-config | grep thread
```

Output help on "rule" options in AsciiDoc format:

```
snort --markup --help-options rule
```

Note
Snort stops reading command-line options after the "--help-**" and "--list-**" options, so any other options should be placed before them.

4.3 Sniffing and Logging

Read a pcap:

```
snort -r /path/to/my.pcap
```

Dump the packets to stdout:

```
snort -r /path/to/my.pcap -L dump
```

Dump packets with application data and layer 2 headers

```
snort -r /path/to/my.pcap -L dump -d -e
```

Note
Command line options must be specified separately. "snort -de" won’t work. You can still concatenate options and their arguments, however, so "snort -Ldump" will work.

Dump packets from all pcaps in a directory:

```
snort --pcap-dir /path/to/pcap/dir --pcap-filter ’*.pcap’ -L dump -d -e
```

Log packets to a directory:

```
snort --pcap-dir /path/to/pcap/dir --pcap-filter ’*.pcap’ -L dump -l /path/to/log/ ↩
   dir
```

4.4 Configuration

Validate a configuration file:

```
snort -c $my_path/etc/snort/snort.lua
```

Validate a configuration file and a separate rules file:

```
snort -c $my_path/etc/snort/snort.lua -R $my_path/etc/snort/sample.rules
```

Read rules from stdin and validate:

```
snort -c $my_path/etc/snort/snort.lua --stdin-rules < $my_path/etc/snort/sample. ↩
   rules
```

Enable warnings for Lua configurations and make warnings fatal:

```
snort -c $my_path/etc/snort/snort.lua --warn-all --pedantic
```

Tell Snort where to look for additional Lua scripts:

```
snort --script-path /path/to/script/dir
```

4.5 IDS mode

Run Snort in IDS mode, reading packets from a pcap:

```
snort -c $my_path/etc/snort/snort.lua -r /path/to/my.pcap
```

Log any generated alerts to the console using the "-A" option:

```
snort -c $my_path/etc/snort/snort.lua -r /path/to/my.pcap -A alert_full
```

Add or modify a configuration from the command line using the "--lua" option:

```
snort -c $my_path/etc/snort/snort.lua -r /path/to/my.pcap -A cmg \
    --lua 'ips = { enable_builtin_rules = true }'
```

Note
The "--lua" option can be specified multiple times.

Run Snort in IDS mode on an entire directory of pcaps, processing each input source on a separate thread:

```
snort -c $my_path/etc/snort/snort.lua --pcap-dir /path/to/pcap/dir \
    --pcap-filter '*.pcap' --max-packet-threads 8
```

Run Snort on 2 interfaces, eth0 and eth1:

```
snort -c $my_path/etc/snort/snort.lua -i "eth0 eth1" -z 2 -A cmg
```

Run Snort inline with the afpacket DAQ:

```
snort -c $my_path/etc/snort/snort.lua --daq afpacket -i "eth0:eth1" \
    -A cmg
```

4.6 Plugins

Load external plugins and use the "ex" alert:

```
snort -c $my_path/etc/snort/snort.lua \
    --plugin-path $my_path/lib/snort_extra \
    -A alert_ex -r /path/to/my.pcap
```

Test the LuaJIT rule option *find* loaded from stdin:

```
snort -c $my_path/etc/snort/snort.lua \
    --script-path $my_path/lib/snort_extra \
    --stdin-rules -A cmg -r /path/to/my.pcap << END
alert tcp any any -> any 80 (
    sid:3; msg:"found"; content:"GET";
    find:"pat='HTTP/1%.%d'" ; )
END
```

4.7 Output Files

To make it simple to configure outputs when you run with multiple packet threads, output files are not explicitly configured. Instead, you can use the options below to format the paths:

```
<logdir>/[<run_prefix>][<id#>][<X>]<name>
```

Log to unified in the current directory:

```
snort -c $my_path/etc/snort/snort.lua -r /path/to/my.pcap -A unified2
```

Log to unified in the current directory with a different prefix:

```
snort -c $my_path/etc/snort/snort.lua -r /path/to/my.pcap -A unified2 \
    --run-prefix take2
```

Log to unified in /tmp:

```
snort -c $my_path/etc/snort/snort.lua -r /path/to/my.pcap -A unified2 -l /tmp
```

Run 4 packet threads and log with thread number prefix (0-3):

```
snort -c $my_path/etc/snort/snort.lua --pcap-dir /path/to/pcap/dir \
    --pcap-filter '*.pcap' -z 4 -A unified2
```

Run 4 packet threads and log in thread number subdirs (0-3):

```
snort -c $my_path/etc/snort/snort.lua --pcap-dir /path/to/pcap/dir \
    --pcap-filter '*.pcap' -z 4 -A unified2 --id-subdir
```

Note

subdirectories are created automatically if required. Log filename is based on module name that writes the file. All text mode outputs default to stdout. These options can be combined.

4.8 DAQ Alternatives

Process hext packets from stdin:

```
snort -c $my_path/etc/snort/snort.lua \
    --daq-dir $my_path/lib/snort/daqs --daq hext -i tty << END
$packet 10.1.2.3 48620 -> 10.9.8.7 80
"GET / HTTP/1.1\r\n"
"Host: localhost\r\n"
"\r\n"
END
```

Process raw ethernet from hext file:

```
snort -c $my_path/etc/snort/snort.lua \
    --daq-dir $my_path/lib/snort/daqs --daq hext \
    --daq-var dlt=1 -r <hext-file>
```

Process a directory of plain files (ie non-pcap) with 4 threads with 8K buffers:

```
snort -c $my_path/etc/snort/snort.lua \
    --daq-dir $my_path/lib/snort/daqs --daq file \
    --pcap-dir path/to/files -z 4 -s 8192
```

Bridge two TCP connections on port 8000 and inspect the traffic:

```
snort -c $my_path/etc/snort/snort.lua \
    --daq-dir $my_path/lib/snort/daqs --daq socket
```

4.9 Logger Alternatives

Dump TCP stream payload in hext mode:

```
snort -c $my_path/etc/snort/snort.lua -L hext
```

Output timestamp, pkt_num, proto, pkt_gen, dgm_len, dir, src_ap, dst_ap, rule, action for each alert:

```
snort -c $my_path/etc/snort/snort.lua -A csv
```

Output the old test format alerts:

```
snort -c $my_path/etc/snort/snort.lua \
    --lua "alert_csv = { fields = 'pkt_num gid sid rev', separator = '\t' }"
```

4.10 Shell

You must build with --enable-shell to make the command line shell available.

Enable shell mode:

```
snort --shell <args>
```

You will see the shell mode command prompt, which looks like this:

```
o")~
```

(The prompt can be changed with the SNORT_PROMPT environment variable.)

You can pause immediately after loading the configuration and again before exiting with:

```
snort --shell --pause <args>
```

In that case you must issue the resume() command to continue. Enter quit() to terminate Snort or detach() to exit the shell. You can list the available commands with help().

To enable local telnet access on port 12345:

```
snort --shell -j 12345 <args>
```

The command line interface is still under development. Suggestions are welcome.

4.11 Signals

Note
The following examples assume that Snort is currently running and has a process ID of <pid>.

Modify and Reload Configuration:

```
echo 'suppress = { { gid = 1, sid = 2215 } }' >> $my_path/etc/snort/snort.lua
kill -hup <pid>
```

Dump stats to stdout:

```
kill -usr1 <pid>
```

Shutdown normally:

```
kill -term <pid>
```

Exit without flushing packets:

```
kill -quit <pid>
```

List available signals:

```
snort --help-signals
```

Note
The available signals may vary from platform to platform.

5 Features

This section explains how to use key features of Snort.

5.1 Binder

One of the fundamental differences between Snort 2 and Snort 3 concerns configuration related to networks and ports. Here is a brief review of Snort 2 configuration for network and service related components:

- Snort’s configuration has a default policy and optional policies selected by VLAN or network (with config binding).
- Each policy contains a user defined set of preprocessor configurations.
- Each preprocessor has a default configuration and some support non-default configurations selected by network.
- Most preprocessors have port configurations.
- The default policy may also contain a list of ports to ignore.

In Snort 3, the above configurations are done in a single module called the binder. Here is an example:

```
binder =
{
    -- allow all tcp port 22:
    -- (similar to Snort 2 config ignore_ports)
    { when = { proto = ’tcp’, ports = ’22’ }, use = { action = ’allow’ } },

-- select a config file by vlan
-- (similar to Snort 2 config binding by vlan)
{ when = { vlans = ’1024’ }, use = { file = ’vlan.lua’ } },

-- use a non-default HTTP inspector for port 8080:
-- (similar to a Snort 2 targeted preprocessor config)
{ when = { nets = ’192.168.0.0/16’, proto = ’tcp’, ports = ’8080’ },
  use = { name = ’alt_http’, type = ’http_inspect’ } },
-- use the default inspectors:
-- (similar to a Snort 2 default preprocessor config)
{ when = { proto = 'tcp' }, use = { type = 'stream_tcp' } },
{ when = { service = 'http' }, use = { type = 'http_inspect' } },

    -- figure out which inspector to run automatically:
    { use = { type = 'wizard' } }
}
```

Bindings are evaluated when a session starts and again if and when service is identified on the session. Essentially, the bindings are a list of when-use rules evaluated from top to bottom. The first matching network and service configurations are applied. binder.when can contain any combination of criteria and binder.use can specify an action, config file, or inspector configuration.

5.2 DCE Inspectors

The main purpose of these inspector are to perform SMB desegmentation and DCE/RPC defragmentation to avoid rule evasion using these techniques.

5.2.1 Overview

The following transports are supported for DCE/RPC: SMB, TCP, and UDP. New rule options have been implemented to improve performance, reduce false positives and reduce the count and complexity of DCE/RPC based rules.

Different from Snort 2, the DCE-RPC preprocessor is split into three inspectors - one for each transport: dce_smb, dce_tcp, dce_udp. This includes the configuration as well as the inspector modules. The Snort 2 server configuration is now split between the inspectors. Options that are meaningful to all inspectors, such as policy and defragmentation, are copied into each inspector configuration. The address/port mapping is handled by the binder. Autodetect functionality is replaced by wizard curses.

5.2.2 Quick Guide

A typical dcerpce configuration looks like this:

```
binder =
{
   {
       when =
       {
           proto = 'tcp',
           ports = '139 445 1025',
        },
       use =
       {
           type = 'dce_smb',
       },
    },
    {
       when =
       {
           proto = 'tcp',
           ports = '135 2103',
       },
       use =
       {
           type = 'dce_tcp',
       },
    },
    {
       when =
       {
           proto = 'udp',
           ports = '1030',
       },
       use =
       {
           type = 'dce_udp',
       },
    }
}

dce_smb = { }

dce_tcp = { }

dce_udp = { }
```

In this example, it defines smb, tcp and udp inspectors based on port. All the configurations are default.

5.2.3 Target Based

There are enough important differences between Windows and Samba versions that a target based approach has been implemented. Some important differences:

- Named pipe instance tracking
- Accepted SMB commands
- AndX command chaining
- Transaction tracking
- Multiple Bind requests
- DCE/RPC Fragmented requests - Context ID
- DCE/RPC Fragmented requests - Operation number
- DCE/RPC Stub data byte order

Because of those differences, each inspector can be configured to different policy. Here are the list of policies supported:

- WinXP (default)
- Win2000
- WinVista
- Win2003
- Win2008
- Win7
- Samba
- Samba-3.0.37
- Samba-3.0.22
- Samba-3.0.20

5.2.4 Reassembling

Both SMB inspector and TCP inspector support reassemble. Reassemble threshold specifies a minimum number of bytes in the DCE/RPC desegmentation and defragmentation buffers before creating a reassembly packet to send to the detection engine. This option is useful in inline mode so as to potentially catch an exploit early before full defragmentation is done. A value of 0 s supplied as an argument to this option will, in effect, disable this option. Default is disabled.

5.2.5 SMB

SMB inspector is one of the most complex inspectors. In addition to supporting rule options and lots of inspector rule events, it also supports file processing for both SMB version 1, 2, and 3.

Finger Print Policy

In the initial phase of an SMB session, the client needs to authenticate with a SessionSetupAndX. Both the request and response to this command contain OS and version information that can allow the inspector to dynamically set the policy for a session which allows for better protection against Windows and Samba specific evasions.

File Inspection

SMB inspector supports file inspection. A typical configuration looks like this:

```
binder =
{
   {
       when =
       {
           proto = 'tcp',
           ports = '139 445',
       },
       use =
       {
           type = 'dce_smb',
       },
   },
}

dce_smb =
{
    smb_file_inspection = 'on',
    smb_file_depth = 0,
 }

file_id =
{
    enable_type = true,
    enable_signature = true,
    enable_capture = true,
    file_rules = magics,
}
```

First, define a binder to map tcp port 139 and 445 to smb. Then, enable file inspection in smb inspection and set the file depth as unlimited. Lastly, enable file inspector to inspect file type, calculate file signature, and capture file. The details of file inspector are explained in file processing section.

SMB inspector does inspection of normal SMB file transfers. This includes doing file type and signature through the file processing as well as setting a pointer for the "file_data" rule option. Note that the "file_depth" option only applies to the maximum amount of file data for which it will set the pointer for the "file_data" rule option. For file type and signature it will use the value configured for the file API. If "only" is specified, the inspector will only do SMB file inspection, i.e. it will not do any DCE/RPC tracking or inspection. If "on" is specified with no arguments, the default file depth is 16384 bytes. An argument of -1 to "file-depth" disables setting the pointer for "file_data", effectively disabling SMB file inspection in rules. An argument of 0 to "file_depth" means unlimited. Default is "off", i.e. no SMB file inspection is done in the inspector.

5.2.6 TCP

dce_tcp inspector supports defragementation, reassembling, and policy that is similar to SMB.

5.2.7 UDP

dce_udp is a very simple inspector that only supports defragementation

5.2.8 Rule Options

New rule options are supported by enabling the dcerpc2 inspectors:

- dce_iface
- dce_opnum
- dce_stub_data

New modifiers to existing byte_test and byte_jump rule options:

- byte_test: dce
- byte_jump: dce

dce_iface

For DCE/RPC based rules it has been necessary to set flow-bits based on a client bind to a service to avoid false positives. It is necessary for a client to bind to a service before being able to make a call to it. When a client sends a bind request to the server, it can, however, specify one or more service interfaces to bind to. Each interface is represented by a UUID. Each interface UUID is paired with a unique index (or context id) that future requests can use to reference the service that the client is making a call to. The server will respond with the interface UUIDs it accepts as valid and will allow the client to make requests to those services. When a client makes a request, it will specify the context id so the server knows what service the client is making a request to. Instead of using flow-bits, a rule can simply ask the inspector, using this rule option, whether or not the client has bound to a specific interface UUID and whether or not this client request is making a request to it. This can eliminate false positives where more than one service is bound to successfully since the inspector can correlate the bind UUID to the context id used in the request. A DCE/RPC request can specify whether numbers are represented as big endian or little endian. The representation of the interface UUID is different depending on the endianness specified in the DCE/RPC previously requiring two rules - one for big endian and one for little endian. The inspector eliminates the need for two rules by normalizing the UUID. An interface contains a version. Some versions of an interface may not be vulnerable to a certain exploit. Also, a DCE/RPC request can be broken up into 1 or more fragments. Flags (and a field in the connectionless header) are set in the DCE/RPC header to indicate whether the fragment is the first, a middle or the last fragment. Many checks for data in the DCE/RPC request are only relevant if the DCE/RPC request is a first fragment (or full request), since subsequent fragments will contain data deeper into the DCE/RPC request. A rule which is looking for data, say 5 bytes into the request (maybe it’s a length field), will be looking at the wrong data on a fragment other than the first, since the beginning of subsequent fragments are already offset some length from the beginning of the request. This can be a source of false positives in fragmented DCE/RPC traffic. By default it is reasonable to only evaluate if the request is a first fragment (or full request). However, if the "any_frag" option is used to specify evaluating on all fragments.

Examples:

```
dce_iface: 4b324fc8-1670-01d3-1278-5a47bf6ee188;
dce_iface: 4b324fc8-1670-01d3-1278-5a47bf6ee188,<2;
dce_iface: 4b324fc8-1670-01d3-1278-5a47bf6ee188,any_frag;
dce_iface: 4b324fc8-1670-01d3-1278-5a47bf6ee188,=1,any_frag;
```

This option is used to specify an interface UUID. Optional arguments are an interface version and operator to specify that the version be less than (<), greater than (>), equal to (=) or not equal to (!) the version specified. Also, by default the rule will only be evaluated for a first fragment (or full request, i.e. not a fragment) since most rules are written to start at the beginning of a request. The "any_frag" argument says to evaluate for middle and last fragments as well. This option requires tracking client Bind and Alter Context requests as well as server Bind Ack and Alter Context responses for connection-oriented DCE/RPC in the inspector. For each Bind and Alter Context request, the client specifies a list of interface UUIDs along with a handle (or context id) for each interface UUID that will be used during the DCE/RPC session to reference the interface. The server response indicates which interfaces it will allow the client to make requests to - it either accepts or rejects the client's wish to bind to a certain interface. This tracking is required so that when a request is processed, the context id used in the request can be correlated with the interface UUID it is a handle for.

hexlong and hexshort will be specified and interpreted to be in big endian order (this is usually the default way an interface UUID will be seen and represented). As an example, the following Messenger interface UUID as taken off the wire from a little endian Bind request:

```
|f8 91 7b 5a 00 ff d0 11 a9 b2 00 c0 4f b6 e6 fc|
```

must be written as:

```
5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc
```

The same UUID taken off the wire from a big endian Bind request:

```
|5a 7b 91 f8 ff 00 11 d0 a9 b2 00 c0 4f b6 e6 fc|
```

must be written the same way:

```
5a7b91f8-ff00-11d0-a9b2-00c04fb6e6fc
```

This option matches if the specified interface UUID matches the interface UUID (as referred to by the context id) of the DCE/RPC request and if supplied, the version operation is true. This option will not match if the fragment is not a first fragment (or full request) unless the "any_frag" option is supplied in which case only the interface UUID and version need match. Note that a defragmented DCE/RPC request will be considered a full request.

Using this rule option will automatically insert fast pattern contents into the fast pattern matcher. For UDP rules, the interface UUID, in both big and little endian format will be inserted into the fast pattern matcher. For TCP rules, (1) if the rule option "flow:to_server|from_client" is used, |05 00 00| will be inserted into the fast pattern matcher, (2) if the rule option "flow:from_server|to_client" is used, |05 00 02| will be inserted into the fast pattern matcher and (3) if the flow isn't known, |05 00| will be inserted into the fast pattern matcher. Note that if the rule already has content rule options in it, the best (meaning longest) pattern will be used. If a content in the rule uses the fast_pattern rule option, it will unequivocally be used over the above mentioned patterns.

dce_opnum

The opnum represents a specific function call to an interface. After is has been determined that a client has bound to a specific interface and is making a request to it (see above - dce_iface) usually we want to know what function call it is making to that service. It is likely that an exploit lies in the particular DCE/RPC function call.

Examples:

```
dce_opnum: 15;
dce_opnum: 15-18;
dce_opnum: 15,18-20;
dce_opnum: 15,17,20-22;
```

This option is used to specify an opnum (or operation number), opnum range or list containing either or both opnum and/or opnum-range. The opnum of a DCE/RPC request will be matched against the opnums specified with this option. This option matches if any one of the opnums specified match the opnum of the DCE/RPC request.

dce_stub_data

Since most DCE/RPC based rules had to do protocol decoding only to get to the DCE/RPC stub data, i.e. the remote procedure call or function call data, this option will alleviate this need and place the cursor at the beginning of the DCE/RPC stub data. This reduces the number of rule option checks and the complexity of the rule.

This option takes no arguments.

Example:

```
dce_stub_data;
```

This option is used to place the cursor (used to walk the packet payload in rules processing) at the beginning of the DCE/RPC stub data, regardless of preceding rule options. There are no arguments to this option. This option matches if there is DCE/RPC stub data.

The cursor is moved to the beginning of the stub data. All ensuing rule options will be considered "sticky" to this buffer. The first rule option following dce_stub_data should use absolute location modifiers if it is position-dependent. Subsequent rule options should use a relative modifier if they are meant to be relative to a previous rule option match in the stub data buffer. Any rule option that does not specify a relative modifier will be evaluated from the start of the stub data buffer. To leave the stub data buffer and return to the main payload buffer, use the "pkt_data" rule option.

byte_test and byte_jump

A DCE/RPC request can specify whether numbers are represented in big or little endian. These rule options will take as a new argument "dce" and will work basically the same as the normal byte_test/byte_jump, but since the DCE/RPC inspector will know the endianness of the request, it will be able to do the correct conversion.

Examples:

```
byte_test: 4,>,35000,0,relative,dce;
byte_test: 2,!=,2280,-10,relative,dce;
```

When using the "dce" argument to a byte_test, the following normal byte_test arguments will not be allowed: "big", "little", "string", "hex", "dec" and "oct".

Examples:

```
byte_jump:4,-4,relative,align,multiplier 2,post_offset -4,dce;
```

When using the dce argument to a byte_jump, the following normal byte_jump arguments will not be allowed: "big", "little", "string", "hex", "dec", "oct" and "from_beginning"

5.3 File Processing

With the volume of malware transferred through network increasing, network file inspection becomes more and more important. This feature will provide file type identification, file signature creation, and file capture capabilities to help users deal with those challenges.

5.3.1 Overview

There are two parts of file services: file APIs and file policy. File APIs provides all the file inspection functionalities, such as file type identification, file signature calculation, and file capture. File policy provides users ability to control file services, such as enable/disable/configure file type identification, file signature, or file capture.

In addition to all capabilities from Snort 2, we support customized file policy along with file event log.

- Supported protocols: HTTP, SMTP, IMAP, POP3, FTP, and SMB.
- Supported file signature calculation: SHA256

5.3.2 Quick Guide

A very simple configuration has been included in lua/snort.lua file. A typical file configuration looks like this:

```
dofile('magic.lua')

my_file_policy =
{
    {  when = { file_type_id = 0 }, use = { verdict = 'log', enable_file_signature ↩
         = true, enable_file_capture = true } }
    {  when = { file_type_id = 22 }, use = { verdict = 'log',  ↩
        enable_file_signature = true } },
    {  when = { sha256 = " ↩
        F74DC976BC8387E7D4FC0716A069017A0C7ED13F309A523CC41A8739CCB7D4B6" }, use =  ↩
        { verdict = 'block'} },
}

file_id =
{
    enable_type = true,
    enable_signature = true,
    enable_capture = true,
    file_rules = magics,
    trace_type = true,
    trace_signature = true,
    trace_stream = true,
    file_policy = my_file_policy,
 }

file_log =
{
    log_pkt_time = true,
    log_sys_time = false,
}
```

There are 3 steps to enable file processing:

- First, you need to include the file magic rules.
- Then, define the file policy and configure the inspector
- At last, enable file_log to get detailed information about file event

5.3.3 Pre-packaged File Magic Rules

A set of file magic rules is packaged with Snort. They can be located at "lua/file_magic.lua". To use this feature, it is recommended that these pre-packaged rules are used; doing so requires that you include the file in your Snort configuration as such (already in snort.lua):

```
dofile('magic.lua')
```

Example:

```
{ type = "GIF", id = 62, category = "Graphics", rev = 1,
  magic = { { content = "| 47 49 46 38 37 61 |",offset = 0 } } },

{ type = "GIF", id = 63, category = "Graphics", rev = 1,
  magic = { { content = "| 47 49 46 38 39 61 |",offset = 0 } } },
```

The previous two rules define GIF format, because two file magics are different. File magics are specifed by content and offset, which look at content at particular file offset to identify the file type. In this case, two magics look at the beginning of the file. You can use character if it is printable or hex value in between "|".

5.3.4 File Policy

You can enabled file type, file signature, or file capture by configuring file_id. In addition, you can enable trace to see file stream data, file type, and file signature information.

Most importantly, you can configure a file policy that can block/alert some file type or an individual file based on SHA. This allows you build a file blacklist or whitelist.

Example:

```
file_policy =
{
    {  when = { file_type_id = 22 }, use = { verdict = 'log',  ↩
       enable_file_signature = true } },
    {  when = { sha256 = " ↩
       F74DC976BC8387E7D4FC0716A069017A0C7ED13F309A523CC41A8739CCB7D4B6" }, use =  ↩
       { verdict = 'block'} },
    {  when = { file_type_id = 0 }, use = { verdict = 'log', enable_file_signature ↩
        = true, enable_file_capture = true } }
}
```

In this example, it enables this policy:

- For PDF files, they will be logged with signatures.
- For the file matching this SHA, it will be blocked
- For all file types identified, they will be logged with signature, and also captured onto log folder.

5.3.5 File Capture

File can be captured and stored to log folder. We use SHA as file name instead of actual file name to avoid conflicts. You can capture either all files, some file type, or a particular file based on SHA.

You can enable file capture through this config:

```
enable_capture = true,
```

or enable it for some file or file type in your file policy:

```
{  when = { file_type_id = 22 }, use = { verdict = 'log', enable_file_capture = ↩
   true } },
```

The above rule will enable PDF file capture.

5.3.6 File Events

File inspect preprocessor also works as a dynamic output plugin for file events. It logs basic information about file. The log file is in the same folder as other log files with name starting with "file.log".

Example:

```
file_log = { log_pkt_time = true, log_sys_time = false }
```

All file events will be logged in packet time, system time is not logged.

File event example:

```
08/14-19:14:19.100891  10.22.75.72:33734 -> 10.22.75.36:80,
[Name: "malware.exe"] [Verdict: Block] [Type: MSEXE]
[SHA: 6F26E721FDB1AAFD29B41BCF90196DEE3A5412550615A856DAE8E3634BCE9F7A]
[Size: 1039328]
```

5.4 HTTP Inspector

One of the major undertakings for Snort 3 is developing a completely new HTTP inspector. You can configure it by adding:

```
http_inspect = {}
```

to your snort.lua configuration file. Or you can read it in the source code under src/service_inspectors/http_inspect.

The classic HTTP preprocessor is still available in the alpha release under extra. It has been renamed http_server. Be sure not to configure both old and new HTTP inspectors at the same time.

So why a new HTTP inspector?

For starters it is object-oriented. That's good for us because we maintain this software. But it should also be really nice for open-source developers. You can make meaningful changes and additions to HTTP processing without having to understand the whole thing. In fact much of the new HTTP inspector's knowledge of HTTP is centralized in a series of tables where it can be easily reviewed and modified. Many significant changes can be made just by updating these tables.

Http_inspect is the first inspector written specifically for the new Snort 3 architecture. That provides access to one of the very best features of Snort 3: purely PDU-based inspection. The classic preprocessor processes HTTP messages, but even while doing so it is constantly aware of IP packets and how they divide up the TCP data stream. The same HTTP message might be processed differently depending on how the sender (bad guy) divided it up into IP packets.

Http_inspect is free of this burden and can focus exclusively on HTTP. That makes it much simpler, easier to test, and less prone to false positives. It also greatly reduces the opportunity for adversaries to probe the inspector for weak spots by adjusting packet boundaries to disguise bad behavior.

Dealing solely with HTTP messages also opens the door for developing major new features. The http_inspect design supports true stateful processing. Want to ask questions that involve both the client request and the server response? Or different requests in the same session? These things are possible.

Another new feature on the horizon is HTTP/2 analysis. HTTP/2 derives from Google's SPDY project and is in the process of being standardized. Despite the name, it is better to think of HTTP/2 not as a newer version of HTTP/1.1, but rather a separate protocol layer that runs under HTTP/1.1 and on top of TLS or TCP. It's a perfect fit for the new Snort 3 architecture because a new HTTP/2 inspector would naturally output HTTP/1.1 messages but not any underlying packets. Exactly what http_inspect wants to input.

Http_inspect is taking a very different approach to HTTP header fields. The classic preprocessor divides all the HTTP headers following the start line into cookies and everything else. It normalizes the two pieces using a generic process and puts them in buffers that one can write rules against. There is some limited support for examining individual headers within the inspector but it is very specific.

The new concept is that every header should be normalized in an appropriate and specific way and individually made available for the user to write rules against it. If for example a header is supposed to be a date then normalization means put that date in a standard format.

5.5 Performance Monitor

The new and improved performance monitor! Is your sensor being bogged down by too many flows? perf_monitor! Why are certain TCP segments being dropped without hitting a rule? perf_monitor! Why is a sensor leaking water? Not perf_monitor, check with stream…

5.5.1 Overview

The Snort performance monitor is the built-in utility for monitoring system and traffic statistics. All statistics are separated by processing thread. perf_monitor supports several trackers for monitoring such data:

5.5.2 Base Tracker

The base tracker is used to gather running statistics about Snort and its running modules. All Snort modules gather, at the very least, counters for the number of packets reaching it. Most supplement these counts with those for domain specific functions, such as http_inspect's number of GET requests seen.

Statistics are gathered live and can be reported at regular intervals. The stats reported correspond only to the interval in question and are reset at the beginning of each interval.

These are the same counts displayed when Snort shuts down, only sorted amongst the discrete intervals in which they occurred.

Base differs from prior implementations in Snort in that all stats gathered are only raw counts, allowing the data to be evaluated as needed. Additionally, base is entirely pluggable. Data from new Snort plugins can be added to the existing stats either automatically or, if specified, by name and function.

All plugins and counters can be enabled or disabled individually, allowing for only the data that is actually desired instead of overly verbose performance logs.

To enable everything:

```
perf_monitor = { modules = {} }
```

To enable everything within a module:

```
perf_monitor =
{
    modules =
    {
        {
            name = 'stream_tcp',
            pegs = [[ ]]
        },
    }
}
```

To enable specific counts within modules:

```
perf_monitor =
{
    modules =
    {
        {
            name = 'stream_tcp',
            pegs = [[ overlaps gaps ]]
        },
    }
```

Note: Event stats from prior Snorts are now located within base statistics.

5.5.3 Flow Tracker

Flow tracks statistics regarding traffic and L3/L4 protocol distributions. This data can be used to build a profile of traffic for inspector tuning and for identifying where Snort may be stressed.

To enable:

```
perf_monitor = { flow = true }
```

5.5.4 FlowIP Tracker

FlowIP provides statistics for individual hosts within a network. This data can be used for identifying communication habits, such as generating large or small amounts of data, opening a small or large number of sessions, and tendency to send smaller or larger IP packets.

To enable:

```
perf_monitor = { flow_ip = true }
```

5.5.5 CPU Tracker

This tracker monitors the CPU and wall time spent by a given processing thread.

To enable:

```
perf_monitor = { cpu = true }
```

5.6 Sensitive Data Filtering

The `sd_pattern` IPS option provides detection and filtering of Personally Identifiable Information (PII). This information includes credit card numbers, U.S. Social Security numbers, and email addresses. A rich regular expression syntax is available for defining your own PII.

5.6.1 Hyperscan

The `sd_pattern` rule option is powered by the open source Hyperscan library from Intel. It provides a regex grammar which is mostly PCRE compatible. To learn more about Hyperscan see http://01org.github.io/hyperscan/dev-reference/

5.6.2 Syntax

Snort provides `sd_pattern` as IPS rule option with no additional inspector overhead. The Rule option takes the following syntax.

```
sd_pattern: "<pattern>"[, threshold <count>];
```

Pattern

Pattern is the most important and is the only required parameter to `sd_pattern`. It supports 3 built in patterns which are configured by name: "credit_card", "us_social" and "us_social_nodashes", as well as user defined regular expressions of the Hyperscan dialect (see http://01org.github.io/hyperscan/dev-reference/compilation.html#pattern-support).

```
sd_pattern:"credit_card";
```

When configured, Snort will replace the pattern *credit_card* with the built in pattern. In addition to pattern matching, Snort will validate that the matched digits will pass the Luhn-check algorithm. Currently the only pattern that performs extra verification.

```
sd_pattern:"us_social";
sd_pattern:"us_social_nodashes";
```

These special patterns will also be replaced with a built in pattern. Naturally, "us_social" is a pattern of 9 digits separated by −'s in the canonical form.

```
sd_pattern:"\b\w+@ourdomain\.com\b"
```

This is a user defined pattern which matches what is most likely email addresses for the site "ourdomain.com". The pattern is a PCRE compatible regex, *b* matches a word boundary (whitespace, end of line, non-word characters) and *w*+ matches one or more word characters. \. matches a literal ..

The above pattern would match "a@ourdomain.com", "aa@ourdomain.com" but would not match `1@ourdomain.com` `ab12@ourdomain.com` or `@ourdomain.com`.

Note: This is just an example, this pattern is not suitable to detect many correctly formatted emails.

Threshold

Threshold is an optional parameter allowing you to change built in default value (default value is *1*). The following two instances are identical. The first will assume the default value of *1* the second declaration explicitly sets the threshold to *1*.

```
sd_pattern:"This rule requires 1 match";
sd_pattern:"This rule requires 1 match", threshold 1;
```

That's pretty easy, but here is one more example anyway.

```
sd_pattern:"This is a string literal", threshold 300;
```

This example requires 300 matches of the pattern "This is a string literal" to qualify as a positive match. That is, if the string only occurred 299 times in a packet, you will not see an event.

Obfuscating Credit Cards and Social Security Numbers

Snort provides discreet logging for the built in patterns "credit_card", "us_social" and "us_social_nodashes". Enabling `output.obfuscate_pii` makes Snort obfuscate the suspect packet payload which was matched by the patterns. This configuration is disabled by default.

```
output =
{
    obfuscate_pii = true
}
```

5.6.3 Example

A complete Snort IPS rule

```
alert tcp ( sid:1; msg:"Credit Card"; sd_pattern:"credit_card"; )
```

Logged output when running Snort in "cmg" alert format.

```
02/25-21:19:05.125553 [**] [1:1:0] "Credit Card" [**] [Priority: 0] {TCP}  ←↩
   10.1.2.3:48620 -> 10.9.8.7:8
02:01:02:03:04:05 -> 02:09:08:07:06:05 type:0x800 len:0x46
10.1.2.3:48620 -> 10.9.8.7:8 TCP TTL:64 TOS:0x0 ID:14 IpLen:20 DgmLen:56
***A**** Seq: 0xB2  Ack: 0x2  Win: 0x2000  TcpLen: 20
- - - raw[16] - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
58 58 58 58 58 58 58 58 58 58 58 58 39 32 39 34             XXXXXXXXXXXX9294
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

5.6.4 Caveats

1. Snort currently requires setting the fast pattern engine to use "hyperscan" in order for `sd_pattern` ips option to function correctly.

   ```
   search_engine = { search_method = 'hyperscan' }
   ```

2. Log obfuscation is only applicable to CMG and Unified2 logging formats.
3. Log obfuscation doesn't support user defined PII patterns. It is currently only supported for the built in patterns for Credit Cards and US Social Security numbers.
4. Log obfuscation doesn't work with stream rebuilt packet payloads. (This is a known bug).

5.7 Wizard

Using the wizard enables port-independent configuration and the detection of malware command and control channels. If the wizard is bound to a session, it peeks at the initial payload to determine the service. For example, *GET* would indicate HTTP and *HELO* would indicate SMTP. Upon finding a match, the service bindings are reevaluated so the session can be handed off to the appropriate inspector. The wizard is still under development; if you find you need to tweak the defaults please let us know.

Additional Details:

- If the wizard and one or more service inspectors are configured w/o explicitly configuring the binder, default bindings will be generated which should work for most common cases.
- Also note that while Snort 2 bindings can only be configured in the default policy, each Snort 3 policy can contain a binder leading to an arbitrary hierarchy.
- The entire configuration can be reloaded and hot-swapped during run-time via signal or command in both Snort 2 and Snort 3. Ultimately, Snort 3 will support commands to update the binder on the fly, thus enabling incremental reloads of individual inspectors.
- Both Snort 2 and Snort 3 support server specific configurations via a hosts table (XML in Snort 2 and Lua in Snort 3). The table allows you to map network, protocol, and port to a service and policy. This table can be reloaded and hot-swapped separately from the config file.
- You can find the specifics on the binder, wizard, and hosts tables in the manual or command line like this: snort --help-module binder, etc.

6 Basic Modules

Internal modules which are not plugins are termed "basic". These include configuration for core processing.

6.1 active

What: configure responses

Type: basic

Configuration:

- int **active.attempts** = 0: number of TCP packets sent per response (with varying sequence numbers) { 0:20 }
- string **active.device**: use *ip* for network layer responses or *eth0* etc for link layer
- string **active.dst_mac**: use format *01:23:45:67:89:ab*
- int **active.max_responses** = 0: maximum number of responses { 0: }
- int **active.min_interval** = 255: minimum number of seconds between responses { 1: }

6.2 alerts

What: configure alerts

Type: basic

Configuration:

- bool **alerts.alert_with_interface_name** = false: include interface in alert info (fast, full, or syslog only)
- bool **alerts.default_rule_state** = true: enable or disable ips rules
- int **alerts.detection_filter_memcap** = 1048576: set available bytes of memory for detection_filters { 0: }
- int **alerts.event_filter_memcap** = 1048576: set available bytes of memory for event_filters { 0: }
- string **alerts.order** = pass drop alert log: change the order of rule action application
- int **alerts.rate_filter_memcap** = 1048576: set available bytes of memory for rate_filters { 0: }
- string **alerts.reference_net**: set the CIDR for homenet (for use with -l or -B, does NOT change $HOME_NET in IDS mode)
- bool **alerts.stateful** = false: don’t alert w/o established session (note: rule action still taken)
- string **alerts.tunnel_verdicts**: let DAQ handle non-allow verdicts for GTP|Teredo|6in4|4in6 traffic

6.3 attribute_table

What: configure hosts loading

Type: basic

Configuration:

- int **attribute_table.max_hosts** = 1024: maximum number of hosts in attribute table { 32:207551 }
- int **attribute_table.max_services_per_host** = 8: maximum number of services per host entry in attribute table { 1:65535 }
- int **attribute_table.max_metadata_services** = 8: maximum number of services in rule metadata { 1:256 }

6.4 classifications

What: define rule categories with priority

Type: basic

Configuration:

- string **classifications[].name**: name used with classtype rule option
- int **classifications[].priority** = 1: default priority for class { 0: }
- string **classifications[].text**: description of class

6.5 daq

What: configure packet acquisition interface

Type: basic

Configuration:

- string **daq.module_dirs[].str**: string parameter
- string **daq.input_spec**: input specification
- string **daq.module**: DAQ module to use
- string **daq.variables[].str**: string parameter
- int **daq.instances[].id**: instance ID (required) { 0: }
- string **daq.instances[].input_spec**: input specification
- string **daq.instances[].variables[].str**: string parameter
- int **daq.snaplen**: set snap length (same as -s) { 0:65535 }
- bool **daq.no_promisc** = false: whether to put DAQ device into promiscuous mode

Peg counts:

- **daq.pcaps**: total files and interfaces processed
- **daq.received**: total packets received from DAQ
- **daq.analyzed**: total packets analyzed from DAQ
- **daq.dropped**: packets dropped
- **daq.filtered**: packets filtered out
- **daq.outstanding**: packets unprocessed
- **daq.injected**: active responses or replacements
- **daq.allow**: total allow verdicts
- **daq.block**: total block verdicts
- **daq.replace**: total replace verdicts
- **daq.whitelist**: total whitelist verdicts
- **daq.blacklist**: total blacklist verdicts

- **daq.ignore**: total ignore verdicts
- **daq.internal blacklist**: packets blacklisted internally due to lack of DAQ support
- **daq.internal whitelist**: packets whitelisted internally due to lack of DAQ support
- **daq.skipped**: packets skipped at startup
- **daq.idle**: attempts to acquire from DAQ without available packets

6.6 decode

What: general decoder rules

Type: basic

Rules:

- **116:450** (decode) bad IP protocol
- **116:293** (decode) two or more IP (v4 and/or v6) encapsulation layers present
- **116:459** (decode) fragment with zero length
- **116:150** (decode) loopback IP
- **116:151** (decode) same src/dst IP
- **116:449** (decode) unassigned/reserved IP protocol
- **116:472** (decode) too many protocols present

6.7 detection

What: configure general IPS rule processing parameters

Type: basic

Configuration:

- int **detection.asn1** = 256: maximum decode nodes { 1: }
- bool **detection.pcre_enable** = true: disable pcre pattern matching
- int **detection.pcre_match_limit** = 1500: limit pcre backtracking, -1 = max, 0 = off { -1:1000000 }
- int **detection.pcre_match_limit_recursion** = 1500: limit pcre stack consumption, -1 = max, 0 = off { -1:10000 }

Peg counts:

- **detection.analyzed**: packets sent to detection
- **detection.hard evals**: non-fast pattern rule evaluations
- **detection.raw searches**: fast pattern searches in raw packet data
- **detection.cooked searches**: fast pattern searches in cooked packet data
- **detection.pkt searches**: fast pattern searches in packet data
- **detection.alt searches**: alt fast pattern searches in packet data
- **detection.key searches**: fast pattern searches in key buffer

- **detection.header searches**: fast pattern searches in header buffer
- **detection.body searches**: fast pattern searches in body buffer
- **detection.file searches**: fast pattern searches in file buffer
- **detection.alerts**: alerts not including IP reputation
- **detection.total alerts**: alerts including IP reputation
- **detection.logged**: logged packets
- **detection.passed**: passed packets
- **detection.match limit**: fast pattern matches not processed
- **detection.queue limit**: events not queued because queue full
- **detection.log limit**: events queued but not logged
- **detection.event limit**: events filtered
- **detection.alert limit**: events previously triggered on same PDU

6.8 event_filter

What: configure thresholding of events

Type: basic

Configuration:

- int **event_filter[].gid** = 1: rule generator ID { 0: }
- int **event_filter[].sid** = 1: rule signature ID { 0: }
- enum **event_filter[].type**: 1st count events | every count events | once after count events { limit | threshold | both }
- enum **event_filter[].track**: filter only matching source or destination addresses { by_src | by_dst }
- int **event_filter[].count** = 0: number of events in interval before tripping; -1 to disable { -1: }
- int **event_filter[].seconds** = 0: count interval { 0: }
- string **event_filter[].ip**: restrict filter to these addresses according to track

6.9 event_queue

What: configure event queue parameters

Type: basic

Configuration:

- int **event_queue.max_queue** = 8: maximum events to queue { 1: }
- int **event_queue.log** = 3: maximum events to log { 1: }
- enum **event_queue.order_events** = content_length: criteria for ordering incoming events { priority|content_length }
- bool **event_queue.process_all_events** = false: process just first action group or all action groups

6.10 file_id

What: configure file identification

Type: basic

Configuration:

- int **file_id.type_depth** = 1460: stop type ID at this point { 0: }
- int **file_id.signature_depth** = 10485760: stop signature at this point { 0: }
- int **file_id.block_timeout** = 86400: stop blocking after this many seconds { 0: }
- int **file_id.lookup_timeout** = 2: give up on lookup after this many seconds { 0: }
- bool **file_id.block_timeout_lookup** = false: block if lookup times out
- int **file_id.capture_memcap** = 100: memcap for file capture in megabytes { 0: }
- int **file_id.capture_max_size** = 1048576: stop file capture beyond this point { 0: }
- int **file_id.capture_min_size** = 0: stop file capture if file size less than this { 0: }
- int **file_id.capture_block_size** = 32768: file capture block size in bytes { 8: }
- int **file_id.max_files_cached** = 65536: maximal number of files cached in memory { 8: }
- bool **file_id.enable_type** = false: enable type ID
- bool **file_id.enable_signature** = false: enable signature calculation
- bool **file_id.enable_capture** = false: enable file capture
- int **file_id.show_data_depth** = 100: print this many octets { 0: }
- int **file_id.file_rules[].rev** = 0: rule revision { 0: }
- string **file_id.file_rules[].msg**: information about the file type
- string **file_id.file_rules[].type**: file type name
- int **file_id.file_rules[].id** = 0: file type id { 0: }
- string **file_id.file_rules[].category**: file type category
- string **file_id.file_rules[].version**: file type version
- string **file_id.file_rules[].magic[].content**: file magic content
- int **file_id.file_rules[].magic[].offset** = 0: file magic offset { 0: }
- int **file_id.file_policy[].when.file_type_id** = 0: unique ID for file type in file magic rule { 0: }
- string **file_id.file_policy[].when.sha256**: SHA 256
- enum **file_id.file_policy[].use.verdict** = unknown: what to do with matching traffic { unknown | log | stop | block | reset }
- bool **file_id.file_policy[].use.enable_file_type** = false: true/false → enable/disable file type identification
- bool **file_id.file_policy[].use.enable_file_signature** = false: true/false → enable/disable file signature
- bool **file_id.file_policy[].use.enable_file_capture** = false: true/false → enable/disable file capture
- bool **file_id.trace_type** = false: enable runtime dump of type info
- bool **file_id.trace_signature** = false: enable runtime dump of signature info

- bool **file_id.trace_stream** = false: enable runtime dump of file data

Peg counts:

- **file_id.total files**: number of files processed
- **file_id.total file data**: number of file data bytes processed
- **file_id.cache failures**: number of file cache add failures

6.11 high_availability

What: implement flow tracking high availability

Type: basic

Configuration:

- bool **high_availability.enable** = false: enable high availability
- bool **high_availability.daq_channel** = false: enable use of daq data plane channel
- bit_list **high_availability.ports**: side channel message port list { 65535 }
- real **high_availability.min_age** = 1.0: minimum session life before HA updates { 0.0:100.0 }
- real **high_availability.min_sync** = 1.0: minimum interval between HA updates { 0.0:100.0 }

Peg counts:

6.12 host_cache

What: configure hosts

Type: basic

Configuration:

- int **host_cache[].size**: size of host cache

Peg counts:

- **host_cache.lru cache adds**: lru cache added new entry
- **host_cache.lru cache replaces**: lru cache replaced existing entry
- **host_cache.lru cache prunes**: lru cache pruned entry to make space for new entry
- **host_cache.lru cache find hits**: lru cache found entry in cache
- **host_cache.lru cache find misses**: lru cache did not find entry in cache
- **host_cache.lru cache removes**: lru cache found entry and removed it
- **host_cache.lru cache clears**: lru cache clear API calls

6.13 host_tracker

What: configure hosts

Type: basic

Configuration:

- addr **host_tracker[].ip** = 0.0.0.0/32: hosts address / cidr
- enum **host_tracker[].frag_policy**: defragmentation policy { first | linux | bsd | bsd_right | last | windows | solaris }
- enum **host_tracker[].tcp_policy**: tcp reassembly policy { first | last | linux | old_linux | bsd | macos | solaris | irix | hpux11 | hpux10 | windows | win_2003 | vista | proxy }
- string **host_tracker[].services[].name**: service identifier
- enum **host_tracker[].services[].proto** = tcp: ip protocol { tcp | udp }
- port **host_tracker[].services[].port**: port number

Peg counts:

- **host_tracker.service adds**: host service adds
- **host_tracker.service finds**: host service finds
- **host_tracker.service removes**: host service removes

6.14 hosts

What: configure hosts

Type: basic

Configuration:

- addr **hosts[].ip** = 0.0.0.0/32: hosts address / cidr
- enum **hosts[].frag_policy**: defragmentation policy { first | linux | bsd | bsd_right | last | windows | solaris }
- enum **hosts[].tcp_policy**: tcp reassembly policy { first | last | linux | old_linux | bsd | macos | solaris | irix | hpux11 | hpux10 | windows | win_2003 | vista | proxy }
- string **hosts[].services[].name**: service identifier
- enum **hosts[].services[].proto** = tcp: ip protocol { tcp | udp }
- port **hosts[].services[].port**: port number

6.15 ips

What: configure IPS rule processing

Type: basic

Configuration:

- bool **ips.enable_builtin_rules** = false: enable events from builtin rules w/o stubs
- int **ips.id** = 0: correlate unified2 events with configuration { 0:65535 }
- string **ips.include**: legacy snort rules and includes
- enum **ips.mode**: set policy mode { tap | inline | inline-test }
- string **ips.rules**: snort rules and includes

6.16 latency

What: packet and rule latency monitoring and control

Type: basic

Configuration:

- int **latency.packet.max_time** = 500: set timeout for packet latency thresholding (usec) { 0: }
- bool **latency.packet.fastpath** = false: fastpath expensive packets (max_time exceeded)
- enum **latency.packet.action** = none: event action if packet times out and is fastpathed { none | alert | log | alert_and_log }
- int **latency.rule.max_time** = 500: set timeout for rule evaluation (usec) { 0: }
- bool **latency.rule.suspend** = false: temporarily suspend expensive rules
- int **latency.rule.suspend_threshold** = 5: set threshold for number of timeouts before suspending a rule { 1: }
- int **latency.rule.max_suspend_time** = 30000: set max time for suspending a rule (ms, 0 means permanently disable rule) { 0: }
- enum **latency.rule.action** = none: event action for rule latency enable and suspend events { none | alert | log | alert_and_log }

Rules:

- **134:1** (latency) rule tree suspended due to latency
- **134:2** (latency) rule tree re-enabled after suspend timeout
- **134:3** (latency) packet fastpathed due to latency

Peg counts:

- **latency.total packets**: total packets monitored
- **latency.total usecs**: total usecs elapsed
- **latency.max usecs**: maximum usecs elapsed
- **latency.packet timeouts**: packets that timed out
- **latency.total rule evals**: total rule evals monitored
- **latency.rule eval timeouts**: rule evals that timed out
- **latency.rule tree enables**: rule tree re-enables

6.17 memory

What: memory management configuration

Type: basic

Configuration:

- int **memory.cap** = 0: set the per-packet-thread cap on memory (bytes, 0 to disable) { 0: }
- bool **memory.soft** = false: always succeed in allocating memory, even if above the cap
- int **memory.threshold** = 0: set the per-packet-thread threshold for preemptive cleanup actions (percent, 0 to disable) { 0: }

6.18 network

What: configure basic network parameters

Type: basic

Configuration:

- multi **network.checksum_drop** = none: drop if checksum is bad { all | ip | noip | tcp | notcp | udp | noudp | icmp | noicmp | none }
- multi **network.checksum_eval** = none: checksums to verify { all | ip | noip | tcp | notcp | udp | noudp | icmp | noicmp | none }
- bool **network.decode_drops** = false: enable dropping of packets by the decoder
- int **network.id** = 0: correlate unified2 events with configuration { 0:65535 }
- int **network.min_ttl** = 1: alert / normalize packets with lower ttl / hop limit (you must enable rules and / or normalization also) { 1:255 }
- int **network.new_ttl** = 1: use this value for responses and when normalizing { 1:255 }
- int **network.layers** = 40: the maximum number of protocols that Snort can correctly decode { 3:255 }
- int **network.max_ip6_extensions** = 0: the maximum number of IP6 options Snort will process for a given IPv6 layer before raising 116:456 (0 = unlimited) { 0:255 }
- int **network.max_ip_layers** = 0: the maximum number of IP layers Snort will process for a given packet before raising 116:293 (0 = unlimited) { 0:255 }

6.19 output

What: configure general output parameters

Type: basic

Configuration:

- bool **output.dump_chars_only** = false: turns on character dumps (same as -C)
- bool **output.dump_payload** = false: dumps application layer (same as -d)
- bool **output.dump_payload_verbose** = false: dumps raw packet starting at link layer (same as -X)
- bool **output.log_ipv6_extra_data** = false: log IPv6 source and destination addresses as unified2 extra data records
- int **output.event_trace.max_data** = 0: maximum amount of packet data to capture { 0:65535 }
- bool **output.quiet** = false: suppress non-fatal information (still show alerts, same as -q)
- string **output.logdir** = .: where to put log files (same as -l)
- bool **output.obfuscate** = false: obfuscate the logged IP addresses (same as -O)
- bool **output.obfuscate_pii** = false: mask all but the last 4 characters of credit card and social security numbers
- bool **output.show_year** = false: include year in timestamp in the alert and log files (same as -y)
- int **output.tagged_packet_limit** = 256: maximum number of packets tagged for non-packet metrics { 0: }
- bool **output.verbose** = false: be verbose (same as -v)

6.20 packets

What: configure basic packet handling

Type: basic

Configuration:

- bool **packets.address_space_agnostic** = false: determines whether DAQ address space info is used to track fragments and connections
- string **packets.bpf_file**: file with BPF to select traffic for Snort
- int **packets.limit** = 0: maximum number of packets to process before stopping (0 is unlimited) { 0: }
- int **packets.skip** = 0: number of packets to skip before before processing { 0: }
- bool **packets.vlan_agnostic** = false: determines whether VLAN info is used to track fragments and connections

6.21 process

What: configure basic process setup

Type: basic

Configuration:

- string **process.chroot**: set chroot directory (same as -t)
- string **process.threads[].cpuset**: pin the associated thread to this cpuset
- int **process.threads[].thread** = 0: set cpu affinity for the <cur_thread_num> thread that runs { 0: }
- bool **process.daemon** = false: fork as a daemon (same as -D)
- bool **process.dirty_pig** = false: shutdown without internal cleanup
- string **process.set_gid**: set group ID (same as -g)
- string **process.set_uid**: set user ID (same as -u)
- string **process.umask**: set process umask (same as -m)
- bool **process.utc** = false: use UTC instead of local time for timestamps

6.22 profiler

What: configure profiling of rules and/or modules

Type: basic

Configuration:

- bool **profiler.modules.show** = true: show module time profile stats
- int **profiler.modules.count** = 0: limit results to count items per level (0 = no limit) { 0: }
- enum **profiler.modules.sort** = total_time: sort by given field { none | checks | avg_check | total_time }
- int **profiler.modules.max_depth** = -1: limit depth to max_depth (-1 = no limit) { -1: }
- bool **profiler.memory.show** = true: show module memory profile stats
- int **profiler.memory.count** = 0: limit results to count items per level (0 = no limit) { 0: }

- enum **profiler.memory.sort** = total_used: sort by given field { none | allocations | total_used | avg_allocation }
- int **profiler.memory.max_depth** = -1: limit depth to max_depth (-1 = no limit) { -1: }
- bool **profiler.rules.show** = true: show rule time profile stats
- int **profiler.rules.count** = 0: print results to given level (0 = all) { 0: }
- enum **profiler.rules.sort** = total_time: sort by given field { none | checks | avg_check | total_time | matches | no_matches | avg_match | avg_no_match }

6.23 rate_filter

What: configure rate filters (which change rule actions)

Type: basic

Configuration:

- int **rate_filter[].gid** = 1: rule generator ID { 0: }
- int **rate_filter[].sid** = 1: rule signature ID { 0: }
- enum **rate_filter[].track** = by_src: filter only matching source or destination addresses { by_src | by_dst | by_rule }
- int **rate_filter[].count** = 1: number of events in interval before tripping { 0: }
- int **rate_filter[].seconds** = 1: count interval { 0: }
- enum **rate_filter[].new_action** = alert: take this action on future hits until timeout { log | pass | alert | drop | block | reset }
- int **rate_filter[].timeout** = 1: count interval { 0: }
- string **rate_filter[].apply_to**: restrict filter to these addresses according to track

6.24 references

What: define reference systems used in rules

Type: basic

Configuration:

- string **references[].name**: name used with reference rule option
- string **references[].url**: where this reference is defined

6.25 rule_state

What: enable/disable specific IPS rules

Type: basic

Configuration:

- int **rule_state.gid** = 0: rule generator ID { 0: }
- int **rule_state.sid** = 0: rule signature ID { 0: }
- bool **rule_state.enable** = true: enable or disable rule in all policies

6.26 search_engine

What: configure fast pattern matcher

Type: basic

Configuration:

- int **search_engine.bleedover_port_limit** = 1024: maximum ports in rule before demotion to any-any port group { 1: }
- bool **search_engine.bleedover_warnings_enabled** = false: print warning if a rule is demoted to any-any port group
- bool **search_engine.enable_single_rule_group** = false: put all rules into one group
- bool **search_engine.debug** = false: print verbose fast pattern info
- bool **search_engine.debug_print_nocontent_rule_tests** = false: print rule group info during packet evaluation
- bool **search_engine.debug_print_rule_group_build_details** = false: print rule group info during compilation
- bool **search_engine.debug_print_rule_groups_uncompiled** = false: prints uncompiled rule group information
- bool **search_engine.debug_print_rule_groups_compiled** = false: prints compiled rule group information
- int **search_engine.max_pattern_len** = 0: truncate patterns when compiling into state machine (0 means no maximum) { 0: }
- int **search_engine.max_queue_events** = 5: maximum number of matching fast pattern states to queue per packet
- bool **search_engine.inspect_stream_inserts** = false: inspect reassembled payload - disabling is good for performance, bad for detection
- dynamic **search_engine.search_method** = ac_bnfa: set fast pattern algorithm - choose available search engine { ac_banded | ac_bnfa | ac_full | ac_sparse | ac_sparse_bands | ac_std | hyperscan }
- bool **search_engine.search_optimize** = true: tweak state machine construction for better performance
- bool **search_engine.show_fast_patterns** = false: print fast pattern info for each rule
- bool **search_engine.split_any_any** = false: evaluate any-any rules separately to save memory

Peg counts:

- **search_engine.max queued**: maximum fast pattern matches queued for further evaluation
- **search_engine.total flushed**: fast pattern matches discarded due to overflow
- **search_engine.total inserts**: total fast pattern hits
- **search_engine.total unique**: total unique fast pattern hits
- **search_engine.non-qualified events**: total non-qualified events
- **search_engine.qualified events**: total qualified events

6.27 side_channel

What: implement the side-channel asynchronous messaging subsystem

Type: basic

Configuration:

- bit_list **side_channel.ports**: side channel message port list { 65535 }
- string **side_channel.connectors[].connector**: connector handle
- string **side_channel.connector**: connector handle

Peg counts:

6.28 snort

What: command line configuration and shell commands

Type: basic

Configuration:

- string **snort.-?**: <option prefix> output matching command line option quick help (same as --help-options) { (optional) }
- string **snort.-A**: <mode> set alert mode: none, cmg, or alert_*
- addr **snort.-B** = 255.255.255.255/32: <mask> obfuscated IP addresses in alerts and packet dumps using CIDR mask
- implied **snort.-C**: print out payloads with character data only (no hex)
- string **snort.-c**: <conf> use this configuration
- implied **snort.-D**: run Snort in background (daemon) mode
- implied **snort.-d**: dump the Application Layer
- implied **snort.-e**: display the second layer header info
- implied **snort.-f**: turn off fflush() calls after binary log writes
- int **snort.-G**: <0xid> (same as --logid) { 0:65535 }
- string **snort.-g**: <gname> run snort gid as <gname> group (or gid) after initialization
- implied **snort.-H**: make hash tables deterministic
- string **snort.-i**: <iface>... list of interfaces
- port **snort.-j**: <port> to listen for telnet connections
- enum **snort.-k** = all: <mode> checksum mode; default is all { all|noip|notcp|noudp|noicmp|none }
- string **snort.-L**: <mode> logging mode (none, dump, pcap, or log_*)
- string **snort.-l**: <logdir> log to this directory instead of current directory
- implied **snort.-M**: log messages to syslog (not alerts)
- int **snort.-m**: <umask> set umask = <umask> { 0: }
- int **snort.-n**: <count> stop after count packets { 0: }
- implied **snort.-O**: obfuscate the logged IP addresses
- implied **snort.-Q**: enable inline mode operation
- implied **snort.-q**: quiet mode - Don’t show banner and status report
- string **snort.-R**: <rules> include this rules file in the default policy
- string **snort.-r**: <pcap>... (same as --pcap-list)
- string **snort.-S**: <x=v> set config variable x equal to value v
- int **snort.-s** = 1514: <snap> (same as --snaplen); default is 1514 { 68:65535 }
- implied **snort.-T**: test and report on the current Snort configuration
- string **snort.-t**: <dir> chroots process to <dir> after initialization
- implied **snort.-U**: use UTC for timestamps

- string **snort.-u**: <uname> run snort as <uname> or <uid> after initialization
- implied **snort.-V**: (same as --version)
- implied **snort.-v**: be verbose
- implied **snort.-W**: lists available interfaces
- implied **snort.-X**: dump the raw packet data starting at the link layer
- implied **snort.-x**: same as --pedantic
- implied **snort.-y**: include year in timestamp in the alert and log files
- int **snort.-z** = 1: <count> maximum number of packet threads (same as --max-packet-threads); 0 gets the number of CPU cores reported by the system; default is 1 { 0: }
- implied **snort.--alert-before-pass**: process alert, drop, sdrop, or reject before pass; default is pass before alert, drop,…
- string **snort.--bpf**: <filter options> are standard BPF options, as seen in TCPDump
- string **snort.--c2x**: output hex for given char (see also --x2c)
- implied **snort.--create-pidfile**: create PID file, even when not in Daemon mode
- string **snort.--daq**: <type> select packet acquisition module (default is pcap)
- string **snort.--daq-dir**: <dir> tell snort where to find desired DAQ
- implied **snort.--daq-list**: list packet acquisition modules available in optional dir, default is static modules only
- string **snort.--daq-var**: <name=value> specify extra DAQ configuration variable
- implied **snort.--dirty-pig**: don’t flush packets on shutdown
- implied **snort.--dump-builtin-rules**: [<module prefix>] output stub rules for selected modules
- implied **snort.--dump-dynamic-rules**: output stub rules for all loaded rules libraries
- string **snort.--dump-defaults**: [<module prefix>] output module defaults in Lua format { (optional) }
- implied **snort.--dump-version**: output the version, the whole version, and only the version
- implied **snort.--enable-inline-test**: enable Inline-Test Mode Operation
- implied **snort.--help**: list command line options
- string **snort.--help-commands**: [<module prefix>] output matching commands { (optional) }
- string **snort.--help-config**: [<module prefix>] output matching config options { (optional) }
- string **snort.--help-counts**: [<module prefix>] output matching peg counts { (optional) }
- string **snort.--help-module**: <module> output description of given module
- implied **snort.--help-modules**: list all available modules with brief help
- string **snort.--help-options**: <option prefix> output matching command line option quick help (same as -?) { (optional) }
- implied **snort.--help-plugins**: list all available plugins with brief help
- implied **snort.--help-signals**: dump available control signals
- implied **snort.--id-subdir**: create/use instance subdirectories in logdir instead of instance filename prefix
- implied **snort.--id-zero**: use id prefix / subdirectory even with one packet thread
- implied **snort.--list-buffers**: output available inspection buffers

- string **snort.--list-builtin**: <module prefix> output matching builtin rules { (optional) }
- string **snort.--list-gids**: [<module prefix>] output matching generators { (optional) }
- string **snort.--list-modules**: [<module type>] list all known modules of given type { (optional) }
- implied **snort.--list-plugins**: list all known plugins
- string **snort.--lua**: <chunk> extend/override conf with chunk; may be repeated
- int **snort.--logid**: <0xid> log Identifier to uniquely id events for multiple snorts (same as -G) { 0:65535 }
- implied **snort.--markup**: output help in asciidoc compatible format
- int **snort.--max-packet-threads** = 1: <count> configure maximum number of packet threads (same as -z) { 0: }
- implied **snort.--nostamps**: don't include timestamps in log file names
- implied **snort.--nolock-pidfile**: do not try to lock Snort PID file
- implied **snort.--pause**: wait for resume/quit command before processing packets/terminating
- string **snort.--pcap-file**: <file> file that contains a list of pcaps to read - read mode is implied
- string **snort.--pcap-list**: <list> a space separated list of pcaps to read - read mode is implied
- string **snort.--pcap-dir**: <dir> a directory to recurse to look for pcaps - read mode is implied
- string **snort.--pcap-filter**: <filter> filter to apply when getting pcaps from file or directory
- int **snort.--pcap-loop**: <count> read all pcaps <count> times; 0 will read until Snort is terminated { -1: }
- implied **snort.--pcap-no-filter**: reset to use no filter when getting pcaps from file or directory
- implied **snort.--pcap-reload**: if reading multiple pcaps, reload snort config between pcaps
- implied **snort.--pcap-show**: print a line saying what pcap is currently being read
- implied **snort.--pedantic**: warnings are fatal
- string **snort.--plugin-path**: <path> where to find plugins
- implied **snort.--process-all-events**: process all action groups
- string **snort.--rule**: <rules> to be added to configuration; may be repeated
- implied **snort.--rule-to-hex**: output so rule header to stdout for text rule on stdin
- implied **snort.--rule-to-text**: output plain so rule header to stdout for text rule on stdin
- string **snort.--run-prefix**: <pfx> prepend this to each output file
- string **snort.--script-path**: <path> to a luajit script or directory containing luajit scripts
- implied **snort.--shell**: enable the interactive command line
- implied **snort.--piglet**: enable piglet test harness mode
- implied **snort.--show-plugins**: list module and plugin versions
- int **snort.--skip**: <n> skip 1st n packets { 0: }
- int **snort.--snaplen** = 1514: <snap> set snaplen of packet (same as -s) { 68:65535 }
- implied **snort.--stdin-rules**: read rules from stdin until EOF or a line starting with END is read
- implied **snort.--treat-drop-as-alert**: converts drop, sdrop, and reject rules into alert rules during startup
- implied **snort.--treat-drop-as-ignore**: use drop, sdrop, and reject rules to ignore session traffic when not inline

- string **snort.--catch-test**: comma separated list of cat unit test tags or *all*
- implied **snort.--version**: show version number (same as -V)
- implied **snort.--warn-all**: enable all warnings
- implied **snort.--warn-conf**: warn about configuration issues
- implied **snort.--warn-daq**: warn about DAQ issues, usually related to mode
- implied **snort.--warn-flowbits**: warn about flowbits that are checked but not set and vice-versa
- implied **snort.--warn-hosts**: warn about host table issues
- implied **snort.--warn-plugins**: warn about issues that prevent plugins from loading
- implied **snort.--warn-rules**: warn about duplicate rules and rule parsing issues
- implied **snort.--warn-scripts**: warn about issues discovered while processing Lua scripts
- implied **snort.--warn-symbols**: warn about unknown symbols in your Lua config
- implied **snort.--warn-vars**: warn about variable definition and usage issues
- int **snort.--x2c**: output ASCII char for given hex (see also --c2x)
- string **snort.--x2s**: output ASCII string for given byte code (see also --x2c)

Commands:

- **snort.show_plugins**(): show available plugins
- **snort.dump_stats**(): show summary statistics
- **snort.rotate_stats**(): roll perfmonitor log files
- **snort.reload_config**(filename): load new configuration
- **snort.reload_hosts**(filename): load a new hosts table
- **snort.pause**(): suspend packet processing
- **snort.resume**(): continue packet processing
- **snort.detach**(): exit shell w/o shutdown
- **snort.quit**(): shutdown and dump-stats
- **snort.help**(): this output

Peg counts:

- **snort.local commands**: total local commands processed
- **snort.remote commands**: total remote commands processed
- **snort.signals**: total signals processed
- **snort.conf reloads**: number of times configuration was reloaded
- **snort.attribute table reloads**: number of times hosts table was reloaded
- **snort.attribute table hosts**: total number of hosts in table

6.29 suppress

What: configure event suppressions

Type: basic

Configuration:

- int **suppress[].gid** = 0: rule generator ID { 0: }
- int **suppress[].sid** = 0: rule signature ID { 0: }
- enum **suppress[].track**: suppress only matching source or destination addresses { by_src | by_dst }
- string **suppress[].ip**: restrict suppression to these addresses according to track

7 Codec Modules

Codec is short for coder / decoder. These modules are used for basic protocol decoding, anomaly detection, and construction of active responses.

7.1 arp

What: support for address resolution protocol

Type: codec

Rules:

- **116:109** (arp) truncated ARP

7.2 auth

What: support for IP authentication header

Type: codec

Rules:

- **116:465** (auth) truncated authentication header
- **116:466** (auth) bad authentication header length

7.3 ciscometadata

What: support for cisco metadata

Type: codec

Rules:

- **116:468** (ciscometadata) truncated Cisco Metadata header
- **116:469** (ciscometadata) invalid Cisco Metadata option length
- **116:470** (ciscometadata) invalid Cisco Metadata option type
- **116:471** (ciscometadata) invalid Cisco Metadata SGT

7.4 erspan2

What: support for encapsulated remote switched port analyzer - type 2

Type: codec

Rules:

- **116:462** (erspan2) ERSpan header version mismatch
- **116:463** (erspan2) captured length < ERSpan type2 header length

7.5 erspan3

What: support for encapsulated remote switched port analyzer - type 3

Type: codec

Rules:

- **116:464** (erspan3) captured < ERSpan type3 header length

7.6 esp

What: support for encapsulating security payload

Type: codec

Configuration:

- bool **esp.decode_esp** = false: enable for inspection of esp traffic that has authentication but not encryption

Rules:

- **116:294** (esp) truncated encapsulated security payload header

7.7 eth

What: support for ethernet protocol (DLT 1) (DLT 51)

Type: codec

Rules:

- **116:424** (eth) truncated ethernet header

7.8 fabricpath

What: support for fabricpath

Type: codec

Rules:

- **116:467** (fabricpath) truncated FabricPath header

7.9 gre

What: support for generic routing encapsulation

Type: codec

Rules:

- **116:160** (gre) GRE header length > payload length
- **116:161** (gre) multiple encapsulations in packet
- **116:162** (gre) invalid GRE version
- **116:163** (gre) invalid GRE header
- **116:164** (gre) invalid GRE v.1 PPTP header
- **116:165** (gre) GRE trans header length > payload length

7.10 gtp

What: support for general-packet-radio-service tunnelling protocol

Type: codec

Rules:

- **116:297** (gtp) two or more GTP encapsulation layers present
- **116:298** (gtp) GTP header length is invalid

7.11 icmp4

What: support for Internet control message protocol v4

Type: codec

Rules:

- **116:105** (icmp4) ICMP header truncated
- **116:106** (icmp4) ICMP timestamp header truncated
- **116:107** (icmp4) ICMP address header truncated
- **116:250** (icmp4) ICMP original IP header truncated
- **116:251** (icmp4) ICMP version and original IP header versions differ
- **116:252** (icmp4) ICMP original datagram length < original IP header length
- **116:253** (icmp4) ICMP original IP payload < 64 bits
- **116:254** (icmp4) ICMP original IP payload > 576 bytes
- **116:255** (icmp4) ICMP original IP fragmented and offset not 0
- **116:415** (icmp4) ICMP4 packet to multicast dest address
- **116:416** (icmp4) ICMP4 packet to broadcast dest address
- **116:418** (icmp4) ICMP4 type other

- **116:434** (icmp4) ICMP ping Nmap
- **116:435** (icmp4) ICMP icmpenum v1.1.1
- **116:436** (icmp4) ICMP redirect host
- **116:437** (icmp4) ICMP redirect net
- **116:438** (icmp4) ICMP traceroute ipopts
- **116:439** (icmp4) ICMP source quench
- **116:440** (icmp4) broadscan smurf scanner
- **116:441** (icmp4) ICMP destination unreachable communication administratively prohibited
- **116:442** (icmp4) ICMP destination unreachable communication with destination host is administratively prohibited
- **116:443** (icmp4) ICMP destination unreachable communication with destination network is administratively prohibited
- **116:451** (icmp4) ICMP path MTU denial of service attempt
- **116:452** (icmp4) Linux ICMP header DOS attempt
- **116:426** (icmp4) truncated ICMP4 header

Peg counts:

- **icmp4.bad checksum**: non-zero icmp checksums

7.12 icmp6

What: support for Internet control message protocol v6

Type: codec

Rules:

- **116:427** (icmp6) truncated ICMP6 header
- **116:431** (icmp6) ICMPv6 type not decoded
- **116:432** (icmp6) ICMPv6 packet to multicast address
- **116:285** (icmp6) ICMPv6 packet of type 2 (message too big) with MTU field < 1280
- **116:286** (icmp6) ICMPv6 packet of type 1 (destination unreachable) with non-RFC 2463 code
- **116:287** (icmp6) ICMPv6 router solicitation packet with a code not equal to 0
- **116:288** (icmp6) ICMPv6 router advertisement packet with a code not equal to 0
- **116:289** (icmp6) ICMPv6 router solicitation packet with the reserved field not equal to 0
- **116:290** (icmp6) ICMPv6 router advertisement packet with the reachable time field set > 1 hour
- **116:457** (icmp6) ICMPv6 packet of type 1 (destination unreachable) with non-RFC 4443 code
- **116:460** (icmp6) ICMPv6 node info query/response packet with a code greater than 2

Peg counts:

- **icmp6.bad checksum (ip4)**: nonzero ipcm4 checksums
- **icmp6.bad checksum (ip6)**: nonzero ipcm6 checksums

7.13 igmp

What: support for Internet group management protocol

Type: codec

Rules:

- **116:455** (igmp) DOS IGMP IP options validation attempt

7.14 ipv4

What: support for Internet protocol v4

Type: codec

Rules:

- **116:1** (ipv4) not IPv4 datagram
- **116:2** (ipv4) IPv4 header length < minimum
- **116:3** (ipv4) IPv4 datagram length < header field
- **116:4** (ipv4) IPv4 options found with bad lengths
- **116:5** (ipv4) truncated IPv4 options
- **116:6** (ipv4) IPv4 datagram length > captured length
- **116:404** (ipv4) IPv4 packet with zero TTL
- **116:405** (ipv4) IPv4 packet with bad frag bits (both MF and DF set)
- **116:407** (ipv4) IPv4 packet frag offset + length exceed maximum
- **116:408** (ipv4) IPv4 packet from *current net* source address
- **116:409** (ipv4) IPv4 packet to *current net* dest address
- **116:410** (ipv4) IPv4 packet from multicast source address
- **116:411** (ipv4) IPv4 packet from reserved source address
- **116:412** (ipv4) IPv4 packet to reserved dest address
- **116:413** (ipv4) IPv4 packet from broadcast source address
- **116:414** (ipv4) IPv4 packet to broadcast dest address
- **116:428** (ipv4) IPv4 packet below TTL limit
- **116:430** (ipv4) IPv4 packet both DF and offset set
- **116:448** (ipv4) IPv4 reserved bit set
- **116:444** (ipv4) IPv4 option set
- **116:425** (ipv4) truncated IPv4 header

Peg counts:

- **ipv4.bad checksum**: nonzero ip checksums

7.15 ipv6

What: support for Internet protocol v6

Type: codec

Rules:

- **116:270** (ipv6) IPv6 packet below TTL limit
- **116:271** (ipv6) IPv6 header claims to not be IPv6
- **116:272** (ipv6) IPv6 truncated extension header
- **116:273** (ipv6) IPv6 truncated header
- **116:274** (ipv6) IPv6 datagram length < header field
- **116:275** (ipv6) IPv6 datagram length > captured length
- **116:276** (ipv6) IPv6 packet with destination address ::0
- **116:277** (ipv6) IPv6 packet with multicast source address
- **116:278** (ipv6) IPv6 packet with reserved multicast destination address
- **116:279** (ipv6) IPv6 header includes an undefined option type
- **116:280** (ipv6) IPv6 address includes an unassigned multicast scope value
- **116:281** (ipv6) IPv6 header includes an invalid value for the *next header* field
- **116:282** (ipv6) IPv6 header includes a routing extension header followed by a hop-by-hop header
- **116:283** (ipv6) IPv6 header includes two routing extension headers
- **116:292** (ipv6) IPv6 header has destination options followed by a routing header
- **116:291** (ipv6) IPV6 tunneled over IPv4, IPv6 header truncated, possible Linux kernel attack
- **116:295** (ipv6) IPv6 header includes an option which is too big for the containing header
- **116:296** (ipv6) IPv6 packet includes out-of-order extension headers
- **116:429** (ipv6) IPv6 packet has zero hop limit
- **116:453** (ipv6) ISATAP-addressed IPv6 traffic spoofing attempt
- **116:458** (ipv6) bogus fragmentation packet, possible BSD attack
- **116:461** (ipv6) IPv6 routing type 0 extension header
- **116:456** (ipv6) too many IPv6 extension headers

7.16 mpls

What: support for multiprotocol label switching

Type: codec

Configuration:

- bool **mpls.enable_mpls_multicast** = false: enables support for MPLS multicast
- bool **mpls.enable_mpls_overlapping_ip** = false: enable if private network addresses overlap and must be differentiated by MPLS label(s)

- int **mpls.max_mpls_stack_depth** = -1: set MPLS stack depth { -1: }
- enum **mpls.mpls_payload_type** = ip4: set encapsulated payload type { eth | ip4 | ip6 }

Rules:

- **116:170** (mpls) bad MPLS frame
- **116:171** (mpls) MPLS label 0 appears in non-bottom header
- **116:172** (mpls) MPLS label 1 appears in bottom header
- **116:173** (mpls) MPLS label 2 appears in non-bottom header
- **116:174** (mpls) MPLS label 3 appears in header
- **116:175** (mpls) MPLS label 4, 5,.. or 15 appears in header
- **116:176** (mpls) too many MPLS headers

Peg counts:

- **mpls.total packets**: total mpls labeled packets processed
- **mpls.total bytes**: total mpls labeled bytes processed

7.17 pgm

What: support for pragmatic general multicast

Type: codec

Rules:

- **116:454** (pgm) PGM nak list overflow attempt

7.18 pppoe

What: support for point-to-point protocol over ethernet

Type: codec

Rules:

- **116:120** (pppoe) bad PPPOE frame detected

7.19 tcp

What: support for transmission control protocol

Type: codec

Rules:

- **116:45** (tcp) TCP packet length is smaller than 20 bytes
- **116:46** (tcp) TCP data offset is less than 5
- **116:47** (tcp) TCP header length exceeds packet length
- **116:54** (tcp) TCP options found with bad lengths

- **116:55** (tcp) truncated TCP options
- **116:56** (tcp) T/TCP detected
- **116:57** (tcp) obsolete TCP options found
- **116:58** (tcp) experimental TCP options found
- **116:59** (tcp) TCP window scale option found with length > 14
- **116:400** (tcp) XMAS attack detected
- **116:401** (tcp) Nmap XMAS attack detected
- **116:419** (tcp) TCP urgent pointer exceeds payload length or no payload
- **116:420** (tcp) TCP SYN with FIN
- **116:421** (tcp) TCP SYN with RST
- **116:422** (tcp) TCP PDU missing ack for established session
- **116:423** (tcp) TCP has no SYN, ACK, or RST
- **116:433** (tcp) DDOS shaft SYN flood
- **116:446** (tcp) TCP port 0 traffic
- **116:402** (tcp) DOS NAPTHA vulnerability detected
- **116:403** (tcp) SYN to multicast address

Peg counts:

- **tcp.bad checksum (ip4)**: nonzero tcp over ip checksums
- **tcp.bad checksum (ip6)**: nonzero tcp over ipv6 checksums

7.20 udp

What: support for user datagram protocol

Type: codec

Configuration:

- bool **udp.deep_teredo_inspection** = false: look for Teredo on all UDP ports (default is only 3544)
- bool **udp.enable_gtp** = false: decode GTP encapsulations
- bit_list **udp.gtp_ports** = 2152 3386: set GTP ports { 65535 }

Rules:

- **116:95** (udp) truncated UDP header
- **116:96** (udp) invalid UDP header, length field < 8
- **116:97** (udp) short UDP packet, length field > payload length
- **116:98** (udp) long UDP packet, length field < payload length
- **116:406** (udp) invalid IPv6 UDP packet, checksum zero
- **116:445** (udp) large UDP packet (> 4000 bytes)

- **116:447** (udp) UDP port 0 traffic

Peg counts:

- **udp.bad checksum (ip4)**: nonzero udp over ipv4 checksums
- **udp.bad checksum (ip6)**: nonzero udp over ipv6 checksums

7.21 vlan

What: support for local area network

Type: codec

Rules:

- **116:130** (vlan) bad VLAN frame
- **116:131** (vlan) bad LLC header
- **116:132** (vlan) bad extra LLC info

8 Inspector Modules

These modules perform a variety of functions, including analysis of protocols beyond basic decoding.

8.1 appid

What: application and service identification

Type: inspector

Configuration:

- string **appid.conf**: RNA configuration file
- int **appid.memcap** = 0: disregard - not implemented { 0: }
- bool **appid.log_stats** = false: enable logging of appid statistics
- int **appid.app_stats_period** = 300: time period for collecting and logging appid statistics { 0: }
- int **appid.app_stats_rollover_size** = 20971520: max file size for appid stats before rolling over the log file { 0: }
- int **appid.app_stats_rollover_time** = 86400: max time period for collection appid stats before rolling over the log file { 0: }
- string **appid.app_detector_dir**: directory to load appid detectors from
- int **appid.instance_id** = 0: instance id - need more details for what this is { 0: }
- bool **appid.debug** = false: enable appid debug logging
- bool **appid.dump_ports** = false: enable dump of appid port information
- string **appid.thirdparty_appid_dir**: directory to load thirdparty appid detectors from
- addr **appid.session_log_filter.src_ip** = 0.0.0.0/32: source ip address in CIDR format
- addr **appid.session_log_filter.dst_ip** = 0.0.0.0/32: destination ip address in CIDR format
- port **appid.session_log_filter.src_port**: source port { 1: }

- port **appid.session_log_filter.dst_port**: destination port { 1: }
- string **appid.session_log_filter.protocol**: ip protocol
- bool **appid.session_log_filter.log_all_sessions** = false: enable logging for all appid sessions

Peg counts:

- **appid.packets**: count of packets received
- **appid.processed packets**: count of packets processed
- **appid.ignored packets**: count of packets ignored
- **appid.aim clients**: count of aim clients discovered
- **appid.battlefield flows**: count of battle field flows discovered
- **appid.bgp flows**: count of bgp flows discovered
- **appid.bit clients**: count of bittorrent clients discovered
- **appid.bit flows**: count of bittorrent flows discovered
- **appid.bittracker clients**: count of bittorrent tracker clients discovered
- **appid.bootp flows**: count of bootp flows discovered
- **appid.dcerpc tcp flows**: count of dce rpc flows over tcp discovered
- **appid.dcerpc udp flows**: count of dce rpc flows over udp discovered
- **appid.direct connect flows**: count of direct connect flows discovered
- **appid.dns tcp flows**: count of dns flows over tcp discovered
- **appid.dns udp flows**: count of dns flows over udp discovered
- **appid.ftp flows**: count of ftp flows discovered
- **appid.ftps flows**: count of ftps flows discovered
- **appid.http flows**: count of http flows discovered
- **appid.imap flows**: count of imap service flows discovered
- **appid.imaps flows**: count of imap TLS service flows discovered
- **appid.irc flows**: count of irc service flows discovered
- **appid.kerberos clients**: count of kerberos clients discovered
- **appid.kerberos flows**: count of kerberos service flows discovered
- **appid.kerberos users**: count of kerberos users discovered
- **appid.lpr flows**: count of lpr service flows discovered
- **appid.mdns flows**: count of mdns service flows discovered
- **appid.msn clients**: count of msn clients discovered
- **appid.mysql flows**: count of mysql service flows discovered
- **appid.netbios dgm flows**: count of netbios-dgm service flows discovered
- **appid.netbios ns flows**: count of netbios-ns service flows discovered

- **appid.netbios ssn flows**: count of netbios-ssn service flows discovered
- **appid.nntp flows**: count of nntp flows discovered
- **appid.ntp flows**: count of ntp flows discovered
- **appid.pop flows**: count of pop service flows discovered
- **appid.radius flows**: count of radius flows discovered
- **appid.rexec flows**: count of rexec flows discovered
- **appid.rfb flows**: count of rfb flows discovered
- **appid.rlogin flows**: count of rlogin flows discovered
- **appid.rpc flows**: count of rpc flows discovered
- **appid.rshell flows**: count of rshell flows discovered
- **appid.rsync flows**: count of rsync service flows discovered
- **appid.rtmp flows**: count of rtmp flows discovered
- **appid.rtp clients**: count of rtp clients discovered
- **appid.sip clients**: count of SIP clients discovered
- **appid.sip flows**: count of SIP flows discovered
- **appid.smtp aol clients**: count of AOL smtp clients discovered
- **appid.smtp applemail clients**: count of Apple Mail smtp clients discovered
- **appid.smtp eudora clients**: count of Eudora smtp clients discovered
- **appid.smtp eudora pro clients**: count of Eudora Pro smtp clients discovered
- **appid.smtp evolution clients**: count of Evolution smtp clients discovered
- **appid.smtp kmail clients**: count of KMail smtp clients discovered
- **appid.smtp lotus notes clients**: count of Lotus Notes smtp clients discovered
- **appid.smtp microsoft outlook clients**: count of Microsoft Outlook smtp clients discovered
- **appid.smtp microsoft outlook express clients**: count of Microsoft Outlook Express smtp clients discovered
- **appid.smtp microsoft outlook imo clients**: count of Microsoft Outlook IMO smtp clients discovered
- **appid.smtp mutt clients**: count of Mutt smtp clients discovered
- **appid.smtp thunderbird clients**: count of Thunderbird smtp clients discovered
- **appid.smtp flows**: count of smtp flows discovered
- **appid.smtps flows**: count of smtps flows discovered
- **appid.snmp flows**: count of snmp flows discovered
- **appid.ssh clients**: count of ssh clients discovered
- **appid.ssh flows**: count of ssh flows discovered
- **appid.ssl flows**: count of ssl flows discovered
- **appid.telnet flows**: count of telnet flows discovered
- **appid.tftp flows**: count of tftp flows discovered

- **appid.timbuktu flows**: count of timbuktu flows discovered
- **appid.tns clients**: count of tns clients discovered
- **appid.tns flows**: count of tns flows discovered
- **appid.vnc clients**: count of vnc clients discovered
- **appid.yahoo messenger clients**: count of Yahoo Messenger clients discovered

8.2 arp_spoof

What: detect ARP attacks and anomalies

Type: inspector

Configuration:

- ip4 **arp_spoof.hosts[].ip**: host ip address
- mac **arp_spoof.hosts[].mac**: host mac address

Rules:

- **112:1** (arp_spoof) unicast ARP request
- **112:2** (arp_spoof) ethernet/ARP mismatch request for source
- **112:3** (arp_spoof) ethernet/ARP mismatch request for destination
- **112:4** (arp_spoof) attempted ARP cache overwrite attack

Peg counts:

- **arp_spoof.packets**: total packets

8.3 back_orifice

What: back orifice detection

Type: inspector

Rules:

- **105:1** (back_orifice) BO traffic detected
- **105:2** (back_orifice) BO client traffic detected
- **105:3** (back_orifice) BO server traffic detected
- **105:4** (back_orifice) BO Snort buffer attack

Peg counts:

- **back_orifice.packets**: total packets

8.4 binder

What: configure processing based on CIDRs, ports, services, etc.

Type: inspector

Configuration:

- int **binder[].when.policy_id** = 0: unique ID for selection of this config by external logic { 0: }
- bit_list **binder[].when.ifaces**: list of interface indices { 255 }
- bit_list **binder[].when.vlans**: list of VLAN IDs { 4095 }
- addr_list **binder[].when.nets**: list of networks
- enum **binder[].when.proto**: protocol { any | ip | icmp | tcp | udp | user | file }
- bit_list **binder[].when.ports**: list of ports { 65535 }
- enum **binder[].when.role** = any: use the given configuration on one or any end of a session { client | server | any }
- string **binder[].when.service**: override default configuration
- enum **binder[].use.action** = inspect: what to do with matching traffic { reset | block | allow | inspect }
- string **binder[].use.file**: use configuration in given file
- string **binder[].use.service**: override automatic service identification
- string **binder[].use.type**: select module for binding
- string **binder[].use.name**: symbol name (defaults to type)

Peg counts:

- **binder.packets**: initial bindings
- **binder.resets**: reset bindings
- **binder.blocks**: block bindings
- **binder.allows**: allow bindings
- **binder.inspects**: inspect bindings

8.5 dce_smb

What: dce over smb inspection

Type: inspector

Configuration:

- bool **dce_smb.disable_defrag** = false: Disable DCE/RPC defragmentation
- int **dce_smb.max_frag_len** = 65535: Maximum fragment size for defragmentation { 1514:65535 }
- int **dce_smb.reassemble_threshold** = 0: Minimum bytes received before performing reassembly { 0:65535 }
- enum **dce_smb.smb_fingerprint_policy** = none: Target based SMB policy to use { none | client | server | both }
- enum **dce_smb.policy** = WinXP: Target based policy to use { Win2000 | WinXP | WinVista | Win2003 | Win2008 | Win7 | Samba | Samba-3.0.37 | Samba-3.0.22 | Samba-3.0.20 }

- int **dce_smb.smb_max_chain** = 3: SMB max chain size { 0:255 }
- int **dce_smb.smb_max_compound** = 3: SMB max compound size { 0:255 }
- multi **dce_smb.valid_smb_versions** = all: Valid SMB versions { v1 | v2 | all }
- enum **dce_smb.smb_file_inspection** = off: SMB file inspection { off | on | only }
- int **dce_smb.smb_file_depth** = 16384: SMB file depth for file data { -1: }
- string **dce_smb.smb_invalid_shares**: SMB shares to alert on
- bool **dce_smb.smb_legacy_mode** = false: inspect only SMBv1

Rules:

- **133:2** (dce_smb) SMB - bad NetBIOS session service session type
- **133:3** (dce_smb) SMB - bad SMB message type
- **133:4** (dce_smb) SMB - bad SMB Id (not \xffSMB for SMB1 or not \xfeSMB for SMB2)
- **133:5** (dce_smb) SMB - bad word count or structure size
- **133:6** (dce_smb) SMB - bad byte count
- **133:7** (dce_smb) SMB - bad format type
- **133:8** (dce_smb) SMB - bad offset
- **133:9** (dce_smb) SMB - zero total data count
- **133:10** (dce_smb) SMB - NetBIOS data length less than SMB header length
- **133:12** (dce_smb) SMB - remaining NetBIOS data length less than command byte count
- **133:13** (dce_smb) SMB - remaining NetBIOS data length less than command data size
- **133:14** (dce_smb) SMB - remaining total data count less than this command data size
- **133:15** (dce_smb) SMB - total data sent (STDu64) greater than command total data expected
- **133:16** (dce_smb) SMB - byte count less than command data size (STDu64)
- **133:17** (dce_smb) SMB - invalid command data size for byte count
- **133:18** (dce_smb) SMB - excessive tree connect requests with pending tree connect responses
- **133:19** (dce_smb) SMB - excessive read requests with pending read responses
- **133:20** (dce_smb) SMB - excessive command chaining
- **133:21** (dce_smb) SMB - multiple chained tree connect requests
- **133:22** (dce_smb) SMB - multiple chained tree connect requests
- **133:23** (dce_smb) SMB - chained/compounded login followed by logoff
- **133:24** (dce_smb) SMB - chained/compounded tree connect followed by tree disconnect
- **133:25** (dce_smb) SMB - chained/compounded open pipe followed by close pipe
- **133:26** (dce_smb) SMB - invalid share access
- **133:27** (dce_smb) connection oriented DCE/RPC - invalid major version
- **133:28** (dce_smb) connection oriented DCE/RPC - invalid minor version

- **133:29** (dce_smb) connection-oriented DCE/RPC - invalid PDU type
- **133:30** (dce_smb) connection-oriented DCE/RPC - fragment length less than header size
- **133:32** (dce_smb) connection-oriented DCE/RPC - no context items specified
- **133:33** (dce_smb) connection-oriented DCE/RPC -no transfer syntaxes specified
- **133:34** (dce_smb) connection-oriented DCE/RPC - fragment length on non-last fragment less than maximum negotiated fragment transmit size for client
- **133:35** (dce_smb) connection-oriented DCE/RPC - fragment length greater than maximum negotiated fragment transmit size
- **133:36** (dce_smb) connection-oriented DCE/RPC - alter context byte order different from bind
- **133:37** (dce_smb) connection-oriented DCE/RPC - call id of non first/last fragment different from call id established for fragmented request
- **133:38** (dce_smb) connection-oriented DCE/RPC - opnum of non first/last fragment different from opnum established for fragmented request
- **133:39** (dce_smb) connection-oriented DCE/RPC - context id of non first/last fragment different from context id established for fragmented request
- **133:44** (dce_smb) SMB - invalid SMB version 1 seen
- **133:45** (dce_smb) SMB - invalid SMB version 2 seen
- **133:46** (dce_smb) SMB - invalid user, tree connect, file binding
- **133:47** (dce_smb) SMB - excessive command compounding
- **133:48** (dce_smb) SMB - zero data count
- **133:50** (dce_smb) SMB - maximum number of outstanding requests exceeded
- **133:51** (dce_smb) SMB - outstanding requests with same MID
- **133:52** (dce_smb) SMB - deprecated dialect negotiated
- **133:53** (dce_smb) SMB - deprecated command used
- **133:54** (dce_smb) SMB - unusual command used
- **133:55** (dce_smb) SMB - invalid setup count for command
- **133:56** (dce_smb) SMB - client attempted multiple dialect negotiations on session
- **133:57** (dce_smb) SMB - client attempted to create or set a file’s attributes to readonly/hidden/system
- **133:58** (dce_smb) SMB - file offset provided is greater than file size specified
- **133:59** (dce_smb) SMB - next command specified in SMB2 header is beyond payload boundary

Peg counts:

- **dce_smb.events**: total events
- **dce_smb.PDUs**: total connection-oriented PDUs
- **dce_smb.Binds**: total connection-oriented binds
- **dce_smb.Bind acks**: total connection-oriented binds acks
- **dce_smb.Alter contexts**: total connection-oriented alter contexts
- **dce_smb.Alter context responses**: total connection-oriented alter context responses

- **dce_smb.Bind naks**: total connection-oriented bind naks
- **dce_smb.Requests**: total connection-oriented requests
- **dce_smb.Responses**: total connection-oriented responses
- **dce_smb.Cancels**: total connection-oriented cancels
- **dce_smb.Orphaned**: total connection-oriented orphaned
- **dce_smb.Faults**: total connection-oriented faults
- **dce_smb.Auth3s**: total connection-oriented auth3s
- **dce_smb.Shutdowns**: total connection-oriented shutdowns
- **dce_smb.Rejects**: total connection-oriented rejects
- **dce_smb.MS RPC/HTTP PDUs**: total connection-oriented MS requests to send RPC over HTTP
- **dce_smb.Other requests**: total connection-oriented other requests
- **dce_smb.Other responses**: total connection-oriented other responses
- **dce_smb.Request fragments**: total connection-oriented request fragments
- **dce_smb.Response fragments**: total connection-oriented response fragments
- **dce_smb.Client max fragment size**: connection-oriented client maximum fragment size
- **dce_smb.Client min fragment size**: connection-oriented client minimum fragment size
- **dce_smb.Client segs reassembled**: total connection-oriented client segments reassembled
- **dce_smb.Client frags reassembled**: total connection-oriented client fragments reassembled
- **dce_smb.Server max fragment size**: connection-oriented server maximum fragment size
- **dce_smb.Server min fragment size**: connection-oriented server minimum fragment size
- **dce_smb.Server segs reassembled**: total connection-oriented server segments reassembled
- **dce_smb.Server frags reassembled**: total connection-oriented server fragments reassembled
- **dce_smb.Sessions**: total smb sessions
- **dce_smb.Packets**: total smb packets
- **dce_smb.Ignored bytes**: total ignored bytes
- **dce_smb.Client segs reassembled**: total smb client segments reassembled
- **dce_smb.Server segs reassembled**: total smb server segments reassembled
- **dce_smb.Max outstanding requests**: total smb maximum outstanding requests
- **dce_smb.Files processed**: total smb files processed
- **dce_smb.SMBv2 create**: total number of SMBv2 create packets seen
- **dce_smb.SMBv2 write**: total number of SMBv2 write packets seen
- **dce_smb.SMBv2 read**: total number of SMBv2 read packets seen
- **dce_smb.SMBv2 set info**: total number of SMBv2 set info packets seen
- **dce_smb.SMBv2 tree connect**: total number of SMBv2 tree connect packets seen
- **dce_smb.SMBv2 tree disconnect**: total number of SMBv2 tree disconnect packets seen
- **dce_smb.SMBv2 close**: total number of SMBv2 close packets seen

8.6 dce_tcp

What: dce over tcp inspection

Type: inspector

Configuration:

- bool **dce_tcp.disable_defrag** = false: Disable DCE/RPC defragmentation
- int **dce_tcp.max_frag_len** = 65535: Maximum fragment size for defragmentation { 1514:65535 }
- int **dce_tcp.reassemble_threshold** = 0: Minimum bytes received before performing reassembly { 0:65535 }
- enum **dce_tcp.policy** = WinXP: Target based policy to use { Win2000 | WinXP | WinVista | Win2003 | Win2008 | Win7 | Samba | Samba-3.0.37 | Samba-3.0.22 | Samba-3.0.20 }

Rules:

- **133:27** (dce_tcp) connection oriented DCE/RPC - invalid major version
- **133:28** (dce_tcp) connection oriented DCE/RPC - invalid minor version
- **133:29** (dce_tcp) connection-oriented DCE/RPC - invalid PDU type
- **133:30** (dce_tcp) connection-oriented DCE/RPC - fragment length less than header size
- **133:32** (dce_tcp) connection-oriented DCE/RPC - no context items specified
- **133:33** (dce_tcp) connection-oriented DCE/RPC -no transfer syntaxes specified
- **133:34** (dce_tcp) connection-oriented DCE/RPC - fragment length on non-last fragment less than maximum negotiated fragment transmit size for client
- **133:35** (dce_tcp) connection-oriented DCE/RPC - fragment length greater than maximum negotiated fragment transmit size
- **133:36** (dce_tcp) connection-oriented DCE/RPC - alter context byte order different from bind
- **133:37** (dce_tcp) connection-oriented DCE/RPC - call id of non first/last fragment different from call id established for fragmented request
- **133:38** (dce_tcp) connection-oriented DCE/RPC - opnum of non first/last fragment different from opnum established for fragmented request
- **133:39** (dce_tcp) connection-oriented DCE/RPC - context id of non first/last fragment different from context id established for fragmented request

Peg counts:

- **dce_tcp.events**: total events
- **dce_tcp.PDUs**: total connection-oriented PDUs
- **dce_tcp.Binds**: total connection-oriented binds
- **dce_tcp.Bind acks**: total connection-oriented binds acks
- **dce_tcp.Alter contexts**: total connection-oriented alter contexts
- **dce_tcp.Alter context responses**: total connection-oriented alter context responses
- **dce_tcp.Bind naks**: total connection-oriented bind naks
- **dce_tcp.Requests**: total connection-oriented requests

- **dce_tcp.Responses**: total connection-oriented responses
- **dce_tcp.Cancels**: total connection-oriented cancels
- **dce_tcp.Orphaned**: total connection-oriented orphaned
- **dce_tcp.Faults**: total connection-oriented faults
- **dce_tcp.Auth3s**: total connection-oriented auth3s
- **dce_tcp.Shutdowns**: total connection-oriented shutdowns
- **dce_tcp.Rejects**: total connection-oriented rejects
- **dce_tcp.MS RPC/HTTP PDUs**: total connection-oriented MS requests to send RPC over HTTP
- **dce_tcp.Other requests**: total connection-oriented other requests
- **dce_tcp.Other responses**: total connection-oriented other responses
- **dce_tcp.Request fragments**: total connection-oriented request fragments
- **dce_tcp.Response fragments**: total connection-oriented response fragments
- **dce_tcp.Client max fragment size**: connection-oriented client maximum fragment size
- **dce_tcp.Client min fragment size**: connection-oriented client minimum fragment size
- **dce_tcp.Client segs reassembled**: total connection-oriented client segments reassembled
- **dce_tcp.Client frags reassembled**: total connection-oriented client fragments reassembled
- **dce_tcp.Server max fragment size**: connection-oriented server maximum fragment size
- **dce_tcp.Server min fragment size**: connection-oriented server minimum fragment size
- **dce_tcp.Server segs reassembled**: total connection-oriented server segments reassembled
- **dce_tcp.Server frags reassembled**: total connection-oriented server fragments reassembled
- **dce_tcp.tcp sessions**: total tcp sessions
- **dce_tcp.tcp packets**: total tcp packets

8.7 dce_udp

What: dce over udp inspection

Type: inspector

Configuration:

- bool **dce_udp.disable_defrag** = false: Disable DCE/RPC defragmentation
- int **dce_udp.max_frag_len** = 65535: Maximum fragment size for defragmentation { 1514:65535 }

Rules:

- **133:40** (dce_udp) connection-less DCE/RPC - invalid major version
- **133:41** (dce_udp) connection-less DCE/RPC - invalid PDU type
- **133:42** (dce_udp) connection-less DCE/RPC - data length less than header size
- **133:43** (dce_udp) connection-less DCE/RPC - bad sequence number

Peg counts:

- **dce_udp.events**: total events
- **dce_udp.udp sessions**: total udp sessions
- **dce_udp.udp packets**: total udp packets
- **dce_udp.Requests**: total connection-less requests
- **dce_udp.Acks**: total connection-less acks
- **dce_udp.Cancels**: total connection-less cancels
- **dce_udp.Client facks**: total connection-less client facks
- **dce_udp.Ping**: total connection-less ping
- **dce_udp.Responses**: total connection-less responses
- **dce_udp.Rejects**: total connection-less rejects
- **dce_udp.Cancel acks**: total connection-less cancel acks
- **dce_udp.Server facks**: total connection-less server facks
- **dce_udp.Faults**: total connection-less faults
- **dce_udp.No calls**: total connection-less no calls
- **dce_udp.Working**: total connection-less working
- **dce_udp.Other requests**: total connection-less other requests
- **dce_udp.Other responses**: total connection-less other responses
- **dce_udp.Fragments**: total connection-less fragments
- **dce_udp.Max fragment size**: connection-less maximum fragment size
- **dce_udp.Frags reassembled**: total connection-less fragments reassembled
- **dce_udp.Max seqnum**: max connection-less seqnum

8.8 dnp3

What: dnp3 inspection

Type: inspector

Configuration:

- bool **dnp3.check_crc** = false: validate checksums in DNP3 link layer frames

Rules:

- **145:1** (dnp3) DNP3 link-layer frame contains bad CRC
- **145:2** (dnp3) DNP3 link-layer frame was dropped
- **145:3** (dnp3) DNP3 transport-layer segment was dropped during reassembly
- **145:4** (dnp3) DNP3 reassembly buffer was cleared without reassembling a complete message
- **145:5** (dnp3) DNP3 link-layer frame uses a reserved address

- **145:6** (dnp3) DNP3 application-layer fragment uses a reserved function code

Peg counts:

- **dnp3.total packets**: total packets
- **dnp3.udp packets**: total udp packets
- **dnp3.tcp pdus**: total tcp pdus
- **dnp3.dnp3 link layer frames**: total dnp3 link layer frames
- **dnp3.dnp3 application pdus**: total dnp3 application pdus

8.9 dns

What: dns inspection

Type: inspector

Rules:

- **131:1** (dns) obsolete DNS RR types
- **131:2** (dns) experimental DNS RR types
- **131:3** (dns) DNS client rdata txt overflow

Peg counts:

- **dns.packets**: total packets processed
- **dns.requests**: total dns requests
- **dns.responses**: total dns responses

8.10 file_log

What: log file event to file.log

Type: inspector

Configuration:

- bool **file_log.log_pkt_time** = true: log the packet time when event generated
- bool **file_log.log_sys_time** = false: log the system time when event generated

Peg counts:

- **file_log.total events**: total file events

8.11 ftp_client

What: FTP client configuration module for use with ftp_server

Type: inspector

Configuration:

- bool **ftp_client.bounce** = false: check for bounces
- addr **ftp_client.bounce_to[].address** = 1.0.0.0/32: allowed ip address in CIDR format
- port **ftp_client.bounce_to[].port** = 20: allowed port { 1: }
- port **ftp_client.bounce_to[].last_port**: optional allowed range from port to last_port inclusive { 0: }
- bool **ftp_client.ignore_telnet_erase_cmds** = false: ignore erase character and erase line commands when normalizing
- int **ftp_client.max_resp_len** = -1: maximum ftp response accepted by client { -1: }
- bool **ftp_client.telnet_cmds** = false: detect telnet escape sequences on ftp control channel

8.12 ftp_data

What: FTP data channel handler

Type: inspector

Peg counts:

- **ftp_data.packets**: total packets

8.13 ftp_server

What: main FTP module; ftp_client should also be configured

Type: inspector

Configuration:

- string **ftp_server.chk_str_fmt**: check the formatting of the given commands
- string **ftp_server.data_chan_cmds**: check the formatting of the given commands
- string **ftp_server.data_rest_cmds**: check the formatting of the given commands
- string **ftp_server.data_xfer_cmds**: check the formatting of the given commands
- string **ftp_server.directory_cmds[].dir_cmd**: directory command
- int **ftp_server.directory_cmds[].rsp_code** = 200: expected successful response code for command { 200: }
- string **ftp_server.file_put_cmds**: check the formatting of the given commands
- string **ftp_server.file_get_cmds**: check the formatting of the given commands
- string **ftp_server.encr_cmds**: check the formatting of the given commands
- string **ftp_server.login_cmds**: check the formatting of the given commands
- bool **ftp_server.check_encrypted** = false: check for end of encryption
- string **ftp_server.cmd_validity[].command**: command string

- string **ftp_server.cmd_validity[].format**: format specification
- int **ftp_server.cmd_validity[].length** = 0: specify non-default maximum for command { 0: }
- int **ftp_server.def_max_param_len** = 100: default maximum length of commands handled by server; 0 is unlimited { 1: }
- bool **ftp_server.encrypted_traffic** = false: check for encrypted telnet and ftp
- string **ftp_server.ftp_cmds**: specify additional commands supported by server beyond RFC 959
- bool **ftp_server.ignore_data_chan** = false: do not inspect ftp data channels
- bool **ftp_server.ignore_telnet_erase_cmds** = false: ignore erase character and erase line commands when normalizing
- bool **ftp_server.print_cmds** = false: print command configurations on start up
- bool **ftp_server.telnet_cmds** = false: detect telnet escape sequences of ftp control channel

Rules:

- **125:1** (ftp_server) TELNET cmd on FTP command channel
- **125:2** (ftp_server) invalid FTP command
- **125:3** (ftp_server) FTP command parameters were too long
- **125:4** (ftp_server) FTP command parameters were malformed
- **125:5** (ftp_server) FTP command parameters contained potential string format
- **125:6** (ftp_server) FTP response message was too long
- **125:7** (ftp_server) FTP traffic encrypted
- **125:8** (ftp_server) FTP bounce attempt
- **125:9** (ftp_server) evasive (incomplete) TELNET cmd on FTP command channel

Peg counts:

- **ftp_server.packets**: total packets

8.14 gtp_inspect

What: gtp control channel inspection

Type: inspector

Configuration:

- int **gtp_inspect[].version** = 2: gtp version { 0:2 }
- int **gtp_inspect[].messages[].type** = 0: message type code { 0:255 }
- string **gtp_inspect[].messages[].name**: message name
- int **gtp_inspect[].infos[].type** = 0: information element type code { 0:255 }
- string **gtp_inspect[].infos[].name**: information element name
- int **gtp_inspect[].infos[].length** = 0: information element type code { 0:255 }

Rules:

- **143:1** (gtp_inspect) message length is invalid
- **143:2** (gtp_inspect) information element length is invalid
- **143:3** (gtp_inspect) information elements are out of order

Peg counts:

- **gtp_inspect.sessions**: total sessions processed
- **gtp_inspect.events**: requests
- **gtp_inspect.unknown types**: unknown message types
- **gtp_inspect.unknown infos**: unknown information elements

8.15 http_inspect

What: HTTP inspector

Type: inspector

Configuration:

- int **http_inspect.request_depth** = -1: maximum request message body bytes to examine (-1 no limit) { -1: }
- int **http_inspect.response_depth** = -1: maximum response message body bytes to examine (-1 no limit) { -1: }
- bool **http_inspect.unzip** = true: decompress gzip and deflate message bodies
- bool **http_inspect.normalize_utf** = true: normalize charset utf encodings in response bodies
- bool **http_inspect.normalize_javascript** = false: normalize javascript in response bodies
- int **http_inspect.max_javascript_whitespaces** = 200: maximum consecutive whitespaces allowed within the Javascript obfuscated data { 1:65535 }
- bit_list **http_inspect.bad_characters**: alert when any of specified bytes are present in URI after percent decoding { 255 }
- string **http_inspect.ignore_unreserved**: do not alert when the specified unreserved characters are percent-encoded in a URI.Unreserved characters are 0-9, a-z, A-Z, period, underscore, tilde, and minus. { (optional) }
- bool **http_inspect.percent_u** = false: normalize %uNNNN and %UNNNN encodings
- bool **http_inspect.utf8** = true: normalize 2-byte and 3-byte UTF-8 characters to a single byte
- bool **http_inspect.utf8_bare_byte** = false: when doing UTF-8 character normalization include bytes that were not percent encoded
- bool **http_inspect.iis_unicode** = false: use IIS unicode code point mapping to normalize characters
- string **http_inspect.iis_unicode_map_file**: file containing code points for IIS unicode. { (optional) }
- int **http_inspect.iis_unicode_code_page** = 1252: code page to use from the IIS unicode map file { 0:65535 }
- bool **http_inspect.iis_double_decode** = false: perform double decoding of percent encodings to normalize characters
- int **http_inspect.oversize_dir_length** = 300: maximum length for URL directory { 1:65535 }
- bool **http_inspect.backslash_to_slash** = false: replace \ with / when normalizing URIs
- bool **http_inspect.plus_to_space** = true: replace + with <sp> when normalizing URIs
- bool **http_inspect.simplify_path** = true: reduce URI directory path to simplest form

- bool **http_inspect.test_input** = false: read HTTP messages from text file
- bool **http_inspect.test_output** = false: print out HTTP section data
- int **http_inspect.print_amount** = 1200: number of characters to print from a Field { 1:1000000 }
- bool **http_inspect.print_hex** = false: nonprinting characters printed in [HH] format instead of using an asterisk
- bool **http_inspect.show_pegs** = true: display peg counts with test output

Rules:

- **119:1** (http_inspect) ascii encoding
- **119:2** (http_inspect) double decoding attack
- **119:3** (http_inspect) u encoding
- **119:4** (http_inspect) bare byte unicode encoding
- **119:5** (http_inspect) obsolete event—should not appear
- **119:6** (http_inspect) UTF-8 encoding
- **119:7** (http_inspect) IIS unicode codepoint encoding
- **119:8** (http_inspect) multi_slash encoding
- **119:9** (http_inspect) IIS backslash evasion
- **119:10** (http_inspect) self directory traversal
- **119:11** (http_inspect) directory traversal
- **119:12** (http_inspect) apache whitespace (tab)
- **119:13** (http_inspect) non-RFC http delimiter
- **119:14** (http_inspect) non-RFC defined char
- **119:15** (http_inspect) oversize request-uri directory
- **119:16** (http_inspect) oversize chunk encoding
- **119:17** (http_inspect) unauthorized proxy use detected
- **119:18** (http_inspect) webroot directory traversal
- **119:19** (http_inspect) long header
- **119:20** (http_inspect) max header fields
- **119:21** (http_inspect) multiple content length
- **119:22** (http_inspect) chunk size mismatch detected
- **119:23** (http_inspect) invalid IP in true-client-IP/XFF header
- **119:24** (http_inspect) multiple host hdrs detected
- **119:25** (http_inspect) hostname exceeds 255 characters
- **119:26** (http_inspect) header parsing space saturation
- **119:27** (http_inspect) client consecutive small chunk sizes
- **119:28** (http_inspect) post w/o content-length or chunks

- **119:29** (http_inspect) multiple true ips in a session
- **119:30** (http_inspect) both true-client-IP and XFF hdrs present
- **119:31** (http_inspect) unknown method
- **119:32** (http_inspect) simple request
- **119:33** (http_inspect) unescaped space in HTTP URI
- **119:34** (http_inspect) too many pipelined requests
- **119:35** (http_inspect) anomalous http server on undefined HTTP port
- **119:36** (http_inspect) invalid status code in HTTP response
- **119:37** (http_inspect) no content-length or transfer-encoding in HTTP response
- **119:38** (http_inspect) HTTP response has UTF charset which failed to normalize
- **119:39** (http_inspect) HTTP response has UTF-7 charset
- **119:40** (http_inspect) HTTP response gzip decompression failed
- **119:41** (http_inspect) server consecutive small chunk sizes
- **119:42** (http_inspect) invalid content-length or chunk size
- **119:43** (http_inspect) javascript obfuscation levels exceeds 1
- **119:44** (http_inspect) javascript whitespaces exceeds max allowed
- **119:45** (http_inspect) multiple encodings within javascript obfuscated data
- **119:46** (http_inspect) SWF file zlib decompression failure
- **119:47** (http_inspect) SWF file LZMA decompression failure
- **119:48** (http_inspect) PDF file deflate decompression failure
- **119:49** (http_inspect) PDF file unsupported compression type
- **119:50** (http_inspect) PDF file cascaded compression
- **119:51** (http_inspect) PDF file parse failure
- **119:52** (http_inspect) not HTTP traffic
- **119:53** (http_inspect) chunk length has excessive leading zeros
- **119:54** (http_inspect) white space before or between messages
- **119:55** (http_inspect) request message without URI
- **119:56** (http_inspect) control character in reason phrase
- **119:57** (http_inspect) illegal extra whitespace in start line
- **119:58** (http_inspect) corrupted HTTP version
- **119:59** (http_inspect) unknown HTTP version
- **119:60** (http_inspect) format error in HTTP header
- **119:61** (http_inspect) chunk header options present
- **119:62** (http_inspect) URI badly formatted
- **119:63** (http_inspect) unrecognized type of percent encoding in URI

- **119:64** (http_inspect) HTTP chunk misformatted
- **119:65** (http_inspect) white space following chunk length
- **119:66** (http_inspect) white space within header name
- **119:67** (http_inspect) excessive gzip compression
- **119:68** (http_inspect) gzip decompression failed
- **119:69** (http_inspect) HTTP 0.9 requested followed by another request
- **119:70** (http_inspect) HTTP 0.9 request following a normal request
- **119:71** (http_inspect) message has both Content-Length and Transfer-Encoding
- **119:72** (http_inspect) status code implying no body combined with Transfer-Encoding or nonzero Content-Length
- **119:73** (http_inspect) Transfer-Encoding did not end with chunked
- **119:74** (http_inspect) Transfer-Encoding with chunked not at end
- **119:75** (http_inspect) misformatted HTTP traffic
- **119:76** (http_inspect) unsupported Transfer-Encoding or Content-Encoding used
- **119:77** (http_inspect) unknown Transfer-Encoding or Content-Encoding used
- **119:78** (http_inspect) multiple layers of compression encodings applied

Peg counts:

- **http_inspect.flows**: HTTP connections inspected
- **http_inspect.scans**: TCP segments scanned looking for HTTP messages
- **http_inspect.reassembles**: TCP segments combined into HTTP messages
- **http_inspect.inspections**: total message sections inspected
- **http_inspect.requests**: HTTP request messages inspected
- **http_inspect.responses**: HTTP response messages inspected
- **http_inspect.GET requests**: GET requests inspected
- **http_inspect.HEAD requests**: HEAD requests inspected
- **http_inspect.POST requests**: POST requests inspected
- **http_inspect.PUT requests**: PUT requests inspected
- **http_inspect.DELETE requests**: DELETE requests inspected
- **http_inspect.CONNECT requests**: CONNECT requests inspected
- **http_inspect.OPTIONS requests**: OPTIONS requests inspected
- **http_inspect.TRACE requests**: TRACE requests inspected
- **http_inspect.other requests**: other request methods inspected
- **http_inspect.request bodies**: POST, PUT, and other requests with message bodies
- **http_inspect.chunked**: chunked message bodies
- **http_inspect.URI normalizations**: URIs needing to be normalization
- **http_inspect.URI path**: URIs with path problems
- **http_inspect.URI coding**: URIs with character coding problems

8.16 imap

What: imap inspection

Type: inspector

Configuration:

- int **imap.b64_decode_depth** = 1460: base64 decoding depth { -1:65535 }
- int **imap.bitenc_decode_depth** = 1460: non-Encoded MIME attachment extraction depth { -1:65535 }
- int **imap.qp_decode_depth** = 1460: quoted Printable decoding depth { -1:65535 }
- int **imap.uu_decode_depth** = 1460: Unix-to-Unix decoding depth { -1:65535 }

Rules:

- **141:1** (imap) unknown IMAP3 command
- **141:2** (imap) unknown IMAP3 response
- **141:4** (imap) base64 decoding failed
- **141:5** (imap) quoted-printable decoding failed
- **141:7** (imap) Unix-to-Unix decoding failed

Peg counts:

- **imap.packets**: total packets processed
- **imap.sessions**: total imap sessions
- **imap.b64 attachments**: total base64 attachments decoded
- **imap.b64 decoded bytes**: total base64 decoded bytes
- **imap.qp attachments**: total quoted-printable attachments decoded
- **imap.qp decoded bytes**: total quoted-printable decoded bytes
- **imap.uu attachments**: total uu attachments decoded
- **imap.uu decoded bytes**: total uu decoded bytes
- **imap.non-encoded attachments**: total non-encoded attachments extracted
- **imap.non-encoded bytes**: total non-encoded extracted bytes

8.17 modbus

What: modbus inspection

Type: inspector

Rules:

- **144:1** (modbus) length in Modbus MBAP header does not match the length needed for the given function
- **144:2** (modbus) Modbus protocol ID is non-zero
- **144:3** (modbus) reserved Modbus function code in use

Peg counts:

- **modbus.sessions**: total sessions processed
- **modbus.frames**: total Modbus messages

8.18 normalizer

What: packet scrubbing for inline mode

Type: inspector

Configuration:

- bool **normalizer.ip4.base** = true: clear options
- bool **normalizer.ip4.df** = false: clear don’t frag flag
- bool **normalizer.ip4.rf** = false: clear reserved flag
- bool **normalizer.ip4.tos** = false: clear tos / differentiated services byte
- bool **normalizer.ip4.trim** = false: truncate excess payload beyond datagram length
- bool **normalizer.tcp.base** = true: clear reserved bits and option padding and fix urgent pointer / flags issues
- bool **normalizer.tcp.block** = true: allow packet drops during TCP normalization
- bool **normalizer.tcp.urp** = true: adjust urgent pointer if beyond segment length
- bool **normalizer.tcp.ips** = false: ensure consistency in retransmitted data
- select **normalizer.tcp.ecn** = off: clear ecn for all packets | sessions w/o ecn setup { off | packet | stream }
- bool **normalizer.tcp.pad** = true: clear any option padding bytes
- bool **normalizer.tcp.trim_syn** = false: remove data on SYN
- bool **normalizer.tcp.trim_rst** = false: remove any data from RST packet
- bool **normalizer.tcp.trim_win** = false: trim data to window
- bool **normalizer.tcp.trim_mss** = false: trim data to MSS
- bool **normalizer.tcp.trim** = false: enable all of the TCP trim options
- bool **normalizer.tcp.opts** = true: clear all options except mss, wscale, timestamp, and any explicitly allowed
- bool **normalizer.tcp.req_urg** = true: clear the urgent pointer if the urgent flag is not set
- bool **normalizer.tcp.req_pay** = true: clear the urgent pointer and the urgent flag if there is no payload
- bool **normalizer.tcp.rsv** = true: clear the reserved bits in the TCP header
- bool **normalizer.tcp.req_urp** = true: clear the urgent flag if the urgent pointer is not set
- multi **normalizer.tcp.allow_names**: don’t clear given option names { sack | echo | partial_order | conn_count | alt_checksum | md5 }
- string **normalizer.tcp.allow_codes**: don’t clear given option codes
- bool **normalizer.ip6** = false: clear reserved flag
- bool **normalizer.icmp4** = false: clear reserved flag
- bool **normalizer.icmp6** = false: clear reserved flag

Peg counts:

- **normalizer.ip4 trim**: eth packets trimmed to datagram size
- **normalizer.test ip4 trim**: test eth packets trimmed to datagram size

- **normalizer.ip4 tos**: type of service normalizations
- **normalizer.test ip4 tos**: test type of service normalizations
- **normalizer.ip4 df**: don't frag bit normalizations
- **normalizer.test ip4 df**: test don't frag bit normalizations
- **normalizer.ip4 rf**: reserved flag bit clears
- **normalizer.test ip4 rf**: test reserved flag bit clears
- **normalizer.ip4 ttl**: time-to-live normalizations
- **normalizer.test ip4 ttl**: test time-to-live normalizations
- **normalizer.ip4 opts**: ip4 options cleared
- **normalizer.test ip4 opts**: test ip4 options cleared
- **normalizer.icmp4 echo**: icmp4 ping normalizations
- **normalizer.test icmp4 echo**: test icmp4 ping normalizations
- **normalizer.ip6 hops**: ip6 hop limit normalizations
- **normalizer.test ip6 hops**: test ip6 hop limit normalizations
- **normalizer.ip6 options**: ip6 options cleared
- **normalizer.test ip6 options**: test ip6 options cleared
- **normalizer.icmp6 echo**: icmp6 echo normalizations
- **normalizer.test icmp6 echo**: test icmp6 echo normalizations
- **normalizer.tcp syn options**: SYN only options cleared from non-SYN packets
- **normalizer.test tcp syn options**: test SYN only options cleared from non-SYN packets
- **normalizer.tcp options**: packets with options cleared
- **normalizer.test tcp options**: test packets with options cleared
- **normalizer.tcp paddding**: packets with padding cleared
- **normalizer.test tcp paddding**: test packets with padding cleared
- **normalizer.tcp reserved**: packets with reserved bits cleared
- **normalizer.test tcp reserved**: test packets with reserved bits cleared
- **normalizer.tcp nonce**: packets with nonce bit cleared
- **normalizer.test tcp nonce**: test packets with nonce bit cleared
- **normalizer.tcp urgent ptr**: packets without data with urgent pointer cleared
- **normalizer.test tcp urgent ptr**: test packets without data with urgent pointer cleared
- **normalizer.tcp ecn pkt**: packets with ECN bits cleared
- **normalizer.test tcp ecn pkt**: test packets with ECN bits cleared
- **normalizer.tcp ts ecr**: timestamp cleared on non-ACKs
- **normalizer.test tcp ts ecr**: test timestamp cleared on non-ACKs
- **normalizer.tcp req urg**: cleared urgent pointer when urgent flag is not set

- **normalizer.test tcp req urg**: test cleared urgent pointer when urgent flag is not set
- **normalizer.tcp req pay**: cleared urgent pointer and urgent flag when there is no payload
- **normalizer.test tcp req pay**: test cleared urgent pointer and urgent flag when there is no payload
- **normalizer.tcp req urp**: cleared the urgent flag if the urgent pointer is not set
- **normalizer.test tcp req urp**: test cleared the urgent flag if the urgent pointer is not set
- **normalizer.tcp trim syn**: tcp segments trimmed on SYN
- **normalizer.test tcp trim syn**: test tcp segments trimmed on SYN
- **normalizer.tcp trim rst**: RST packets with data trimmed
- **normalizer.test tcp trim rst**: test RST packets with data trimmed
- **normalizer.tcp trim win**: data trimed to window
- **normalizer.test tcp trim win**: test data trimed to window
- **normalizer.tcp trim mss**: data trimmed to MSS
- **normalizer.test tcp trim mss**: test data trimmed to MSS
- **normalizer.tcp ecn session**: ECN bits cleared
- **normalizer.test tcp ecn session**: test ECN bits cleared
- **normalizer.tcp ts nop**: timestamp options cleared
- **normalizer.test tcp ts nop**: test timestamp options cleared
- **normalizer.tcp ips data**: normalized segments
- **normalizer.test tcp ips data**: test normalized segments
- **normalizer.tcp block**: blocked segments
- **normalizer.test tcp block**: test blocked segments

8.19 packet_capture

What: raw packet dumping facility

Type: inspector

Configuration:

- bool **packet_capture.enable** = false: initially enable packet dumping
- string **packet_capture.filter**: bpf filter to use for packet dump

Commands:

- **packet_capture.enable**(filter): dump raw packets
- **packet_capture.disable**(): stop packet dump

Peg counts:

- **packet_capture.processed**: packets processed against filter
- **packet_capture.captured**: packets matching dumped after matching filter

8.20 perf_monitor

What: performance monitoring and flow statistics collection

Type: inspector

Configuration:

- bool **perf_monitor.base** = true: enable base statistics { nullptr }
- bool **perf_monitor.cpu** = false: enable cpu statistics { nullptr }
- bool **perf_monitor.flow** = false: enable traffic statistics
- bool **perf_monitor.flow_ip** = false: enable statistics on host pairs
- int **perf_monitor.packets** = 10000: minimum packets to report { 0: }
- int **perf_monitor.seconds** = 60: report interval { 1: }
- int **perf_monitor.flow_ip_memcap** = 52428800: maximum memory in bytes for flow tracking { 8200: }
- int **perf_monitor.max_file_size** = 1073741824: files will be rolled over if they exceed this size { 4096: }
- int **perf_monitor.flow_ports** = 1023: maximum ports to track { 0:65535 }
- enum **perf_monitor.output** = file: output location for stats { file | console }
- string **perf_monitor.modules[].name**: name of the module
- string **perf_monitor.modules[].pegs**: list of statistics to track or empty for all counters
- enum **perf_monitor.format** = csv: output format for stats { csv | text }
- bool **perf_monitor.summary** = false: output summary at shutdown

Peg counts:

- **perf_monitor.packets**: total packets

8.21 pop

What: pop inspection

Type: inspector

Configuration:

- int **pop.b64_decode_depth** = 1460: base64 decoding depth { -1:65535 }
- int **pop.bitenc_decode_depth** = 1460: Non-Encoded MIME attachment extraction depth { -1:65535 }
- int **pop.qp_decode_depth** = 1460: Quoted Printable decoding depth { -1:65535 }
- int **pop.uu_decode_depth** = 1460: Unix-to-Unix decoding depth { -1:65535 }

Rules:

- **142:1** (pop) unknown POP3 command
- **142:2** (pop) unknown POP3 response
- **142:4** (pop) base64 decoding failed

- **142:5** (pop) quoted-printable decoding failed
- **142:7** (pop) Unix-to-Unix decoding failed

Peg counts:

- **pop.packets**: total packets processed
- **pop.sessions**: total pop sessions
- **pop.b64 attachments**: total base64 attachments decoded
- **pop.b64 decoded bytes**: total base64 decoded bytes
- **pop.qp attachments**: total quoted-printable attachments decoded
- **pop.qp decoded bytes**: total quoted-printable decoded bytes
- **pop.uu attachments**: total uu attachments decoded
- **pop.uu decoded bytes**: total uu decoded bytes
- **pop.non-encoded attachments**: total non-encoded attachments extracted
- **pop.non-encoded bytes**: total non-encoded extracted bytes

8.22 port_scan

What: port scan inspector; also configure port_scan_global

Type: inspector

Configuration:

- multi **port_scan.protos** = all: choose the protocols to monitor { tcp | udp | icmp | ip | all }
- multi **port_scan.scan_types** = all: choose type of scans to look for { portscan | portsweep | decoy_portscan | distributed_portscan | all }
- enum **port_scan.sense_level** = medium: choose the level of detection { low | medium | high }
- string **port_scan.watch_ip**: list of CIDRs with optional ports to watch
- string **port_scan.ignore_scanners**: list of CIDRs with optional ports to ignore if the source of scan alerts
- string **port_scan.ignore_scanned**: list of CIDRs with optional ports to ignore if the destination of scan alerts
- bool **port_scan.include_midstream** = false: list of CIDRs with optional ports
- bool **port_scan.logfile** = false: write scan events to file

Rules:

- **122:1** (port_scan) TCP portscan
- **122:2** (port_scan) TCP decoy portscan
- **122:3** (port_scan) TCP portsweep
- **122:4** (port_scan) TCP distributed portscan
- **122:5** (port_scan) TCP filtered portscan
- **122:6** (port_scan) TCP filtered decoy portscan

- **122:7** (port_scan) TCP filtered portsweep
- **122:8** (port_scan) TCP filtered distributed portscan
- **122:9** (port_scan) IP protocol scan
- **122:10** (port_scan) IP decoy protocol scan
- **122:11** (port_scan) IP protocol sweep
- **122:12** (port_scan) IP distributed protocol scan
- **122:13** (port_scan) IP filtered protocol scan
- **122:14** (port_scan) IP filtered decoy protocol scan
- **122:15** (port_scan) IP filtered protocol sweep
- **122:16** (port_scan) IP filtered distributed protocol scan
- **122:17** (port_scan) UDP portscan
- **122:18** (port_scan) UDP decoy portscan
- **122:19** (port_scan) UDP portsweep
- **122:20** (port_scan) UDP distributed portscan
- **122:21** (port_scan) UDP filtered portscan
- **122:22** (port_scan) UDP filtered decoy portscan
- **122:23** (port_scan) UDP filtered portsweep
- **122:24** (port_scan) UDP filtered distributed portscan
- **122:25** (port_scan) ICMP sweep
- **122:26** (port_scan) ICMP filtered sweep
- **122:27** (port_scan) open port

8.23 port_scan_global

What: shared settings for port_scan inspectors for use with port_scan

Type: inspector

Configuration:

- int **port_scan_global.memcap** = 1048576: maximum tracker memory in bytes { 1: }

Peg counts:

- **port_scan_global.packets**: total packets

8.24 reputation

What: reputation inspection

Type: inspector

Configuration:

- string **reputation.blacklist**: blacklist file name with ip lists
- int **reputation.memcap** = 500: maximum total MB of memory allocated { 1:4095 }
- enum **reputation.nested_ip** = inner: ip to use when there is IP encapsulation { inner|outer|all }
- enum **reputation.priority** = whitelist: defines priority when there is a decision conflict during run-time { blacklist|whitelist }
- bool **reputation.scan_local** = false: inspect local address defined in RFC 1918
- enum **reputation.white** = unblack: specify the meaning of whitelist { unblack|trust }
- string **reputation.whitelist**: whitelist file name with ip lists

Rules:

- **136:1** (reputation) packets blacklisted
- **136:2** (reputation) packets whitelisted
- **136:3** (reputation) packets monitored

Peg counts:

- **reputation.packets**: total packets processed
- **reputation.blacklisted**: number of packets blacklisted
- **reputation.whitelisted**: number of packets whitelisted
- **reputation.monitored**: number of packets monitored
- **reputation.memory allocated**: total memory allocated

8.25 rpc_decode

What: RPC inspector

Type: inspector

Rules:

- **106:1** (rpc_decode) fragmented RPC records
- **106:2** (rpc_decode) multiple RPC records
- **106:3** (rpc_decode) large RPC record fragment
- **106:4** (rpc_decode) incomplete RPC segment
- **106:5** (rpc_decode) zero-length RPC fragment

Peg counts:

- **rpc_decode.packets**: total packets

8.26 sip

What: sip inspection

Type: inspector

Configuration:

- bool **sip.ignore_call_channel** = false: enables the support for ignoring audio/video data channel
- int **sip.max_call_id_len** = 256: maximum call id field size { 0:65535 }
- int **sip.max_contact_len** = 256: maximum contact field size { 0:65535 }
- int **sip.max_content_len** = 1024: maximum content length of the message body { 0:65535 }
- int **sip.max_dialogs** = 4: maximum number of dialogs within one stream session { 1:4194303 }
- int **sip.max_from_len** = 256: maximum from field size { 0:65535 }
- int **sip.max_requestName_len** = 20: maximum request name field size { 0:65535 }
- int **sip.max_sessions** = 10000: maximum number of sessions that can be allocated { 1024:4194303 }
- int **sip.max_to_len** = 256: maximum to field size { 0:65535 }
- int **sip.max_uri_len** = 256: maximum request uri field size { 0:65535 }
- int **sip.max_via_len** = 1024: maximum via field size { 0:65535 }
- string **sip.methods** = invite cancel ack bye register options: list of methods to check in sip messages

Rules:

- **140:1** (sip) maximum sessions reached
- **140:2** (sip) empty request URI
- **140:3** (sip) URI is too long
- **140:4** (sip) empty call-Id
- **140:5** (sip) Call-Id is too long
- **140:6** (sip) CSeq number is too large or negative
- **140:7** (sip) request name in CSeq is too long
- **140:8** (sip) empty From header
- **140:9** (sip) From header is too long
- **140:10** (sip) empty To header
- **140:11** (sip) To header is too long
- **140:12** (sip) empty Via header
- **140:13** (sip) Via header is too long
- **140:14** (sip) empty Contact
- **140:15** (sip) contact is too long
- **140:16** (sip) content length is too large or negative
- **140:17** (sip) multiple SIP messages in a packet

- **140:18** (sip) content length mismatch
- **140:19** (sip) request name is invalid
- **140:20** (sip) Invite replay attack
- **140:21** (sip) illegal session information modification
- **140:22** (sip) response status code is not a 3 digit number
- **140:23** (sip) empty Content-type header
- **140:24** (sip) SIP version is invalid
- **140:25** (sip) mismatch in METHOD of request and the CSEQ header
- **140:26** (sip) method is unknown
- **140:27** (sip) maximum dialogs within a session reached

Peg counts:

- **sip.packets**: total packets
- **sip.sessions**: total sessions
- **sip.events**: events generated
- **sip.dialogs**: total dialogs
- **sip.ignored channels**: total channels ignored
- **sip.ignored sessions**: total sessions ignored
- **sip.total requests**: total requests
- **sip.invite**: invite
- **sip.cancel**: cancel
- **sip.ack**: ack
- **sip.bye**: bye
- **sip.register**: register
- **sip.options**: options
- **sip.refer**: refer
- **sip.subscribe**: subscribe
- **sip.update**: update
- **sip.join**: join
- **sip.info**: info
- **sip.message**: message
- **sip.notify**: notify
- **sip.prack**: prack
- **sip.total responses**: total responses
- **sip.1xx**: 1xx

- **sip.2xx**: 2xx
- **sip.3xx**: 3xx
- **sip.4xx**: 4xx
- **sip.5xx**: 5xx
- **sip.6xx**: 6xx
- **sip.7xx**: 7xx
- **sip.8xx**: 8xx
- **sip.9xx**: 9xx

8.27 smtp

What: smtp inspection

Type: inspector

Configuration:

- string **smtp.alt_max_command_line_len[].command**: command string
- int **smtp.alt_max_command_line_len[].length** = 0: specify non-default maximum for command { 0: }
- string **smtp.auth_cmds**: commands that initiate an authentication exchange
- string **smtp.binary_data_cmds**: commands that initiate sending of data and use a length value after the command
- int **smtp.bitenc_decode_depth** = 25: depth used to extract the non-encoded MIME attachments { -1:65535 }
- int **smtp.b64_decode_depth** = 25: depth used to decode the base64 encoded MIME attachments { -1:65535 }
- string **smtp.data_cmds**: commands that initiate sending of data with an end of data delimiter
- int **smtp.email_hdrs_log_depth** = 1464: depth for logging email headers { 0:20480 }
- bool **smtp.ignore_data** = false: ignore data section of mail
- bool **smtp.ignore_tls_data** = false: ignore TLS-encrypted data when processing rules
- string **smtp.invalid_cmds**: alert if this command is sent from client side
- bool **smtp.log_email_hdrs** = false: log the SMTP email headers extracted from SMTP data
- bool **smtp.log_filename** = false: log the MIME attachment filenames extracted from the Content-Disposition header within the MIME body
- bool **smtp.log_mailfrom** = false: log the sender's email address extracted from the MAIL FROM command
- bool **smtp.log_rcptto** = false: log the recipient's email address extracted from the RCPT TO command
- int **smtp.max_auth_command_line_len** = 1000: max auth command Line Length { 0:65535 }
- int **smtp.max_command_line_len** = 0: max Command Line Length { 0:65535 }
- int **smtp.max_header_line_len** = 0: max SMTP DATA header line { 0:65535 }
- int **smtp.max_response_line_len** = 0: max SMTP response line { 0:65535 }
- enum **smtp.normalize** = none: turns on/off normalization { none | cmds | all }
- string **smtp.normalize_cmds**: list of commands to normalize

- int **smtp.qp_decode_depth** = 25: quoted-Printable decoding depth { -1:65535 }
- int **smtp.uu_decode_depth** = 25: unix-to-Unix decoding depth { -1:65535 }
- string **smtp.valid_cmds**: list of valid commands
- enum **smtp.xlink2state** = alert: enable/disable xlink2state alert { disable | alert | drop }

Rules:

- **124:1** (smtp) attempted command buffer overflow
- **124:2** (smtp) attempted data header buffer overflow
- **124:3** (smtp) attempted response buffer overflow
- **124:4** (smtp) attempted specific command buffer overflow
- **124:5** (smtp) unknown command
- **124:6** (smtp) illegal command
- **124:7** (smtp) attempted header name buffer overflow
- **124:8** (smtp) attempted X-Link2State command buffer overflow
- **124:10** (smtp) base64 decoding failed
- **124:11** (smtp) quoted-printable decoding failed
- **124:13** (smtp) Unix-to-Unix decoding failed
- **124:14** (smtp) Cyrus SASL authentication attack
- **124:15** (smtp) attempted authentication command buffer overflow

Peg counts:

- **smtp.packets**: total packets processed
- **smtp.sessions**: total smtp sessions
- **smtp.concurrent sessions**: total concurrent smtp sessions
- **smtp.max concurrent sessions**: maximum concurrent smtp sessions
- **smtp.b64 attachments**: total base64 attachments decoded
- **smtp.b64 decoded bytes**: total base64 decoded bytes
- **smtp.qp attachments**: total quoted-printable attachments decoded
- **smtp.qp decoded bytes**: total quoted-printable decoded bytes
- **smtp.uu attachments**: total uu attachments decoded
- **smtp.uu decoded bytes**: total uu decoded bytes
- **smtp.non-encoded attachments**: total non-encoded attachments extracted
- **smtp.non-encoded bytes**: total non-encoded extracted bytes

8.28 ssh

What: ssh inspection

Type: inspector

Configuration:

- int **ssh.max_encrypted_packets** = 25: ignore session after this many encrypted packets { 0:65535 }
- int **ssh.max_client_bytes** = 19600: number of unanswered bytes before alerting on challenge-response overflow or CRC32 { 0:65535 }
- int **ssh.max_server_version_len** = 80: limit before alerting on secure CRT server version string overflow { 0:255 }

Rules:

- **128:1** (ssh) challenge-response overflow exploit
- **128:2** (ssh) SSH1 CRC32 exploit
- **128:3** (ssh) server version string overflow
- **128:5** (ssh) bad message direction
- **128:6** (ssh) payload size incorrect for the given payload
- **128:7** (ssh) failed to detect SSH version string

Peg counts:

- **ssh.packets**: total packets

8.29 ssl

What: ssl inspection

Type: inspector

Configuration:

- bool **ssl.trust_servers** = false: disables requirement that application (encrypted) data must be observed on both sides
- int **ssl.max_heartbeat_length** = 0: maximum length of heartbeat record allowed { 0:65535 }

Rules:

- **137:1** (ssl) invalid client HELLO after server HELLO detected
- **137:2** (ssl) invalid server HELLO without client HELLO detected
- **137:3** (ssl) heartbeat read overrun attempt detected
- **137:4** (ssl) large heartbeat response detected

Peg counts:

- **ssl.packets**: total packets processed
- **ssl.decoded**: ssl packets decoded
- **ssl.client hello**: total client hellos

- **ssl.server hello**: total server hellos
- **ssl.certificate**: total ssl certificates
- **ssl.server done**: total server done
- **ssl.client key exchange**: total client key exchanges
- **ssl.server key exchange**: total server key exchanges
- **ssl.change cipher**: total change cipher records
- **ssl.finished**: total handshakes finished
- **ssl.client application**: total client application records
- **ssl.server application**: total server application records
- **ssl.alert**: total ssl alert records
- **ssl.unrecognized records**: total unrecognized records
- **ssl.handshakes completed**: total completed ssl handshakes
- **ssl.bad handshakes**: total bad handshakes
- **ssl.sessions ignored**: total sessions ignore
- **ssl.detection disabled**: total detection disabled

8.30 stream

What: common flow tracking

Type: inspector

Configuration:

- bool **stream.ip_frags_only** = false: don't process non-frag flows
- int **stream.ip_cache.max_sessions** = 16384: maximum simultaneous sessions tracked before pruning { 2: }
- int **stream.ip_cache.pruning_timeout** = 30: minimum inactive time before being eligible for pruning { 1: }
- int **stream.ip_cache.idle_timeout** = 180: maximum inactive time before retiring session tracker { 1: }
- int **stream.icmp_cache.max_sessions** = 65536: maximum simultaneous sessions tracked before pruning { 2: }
- int **stream.icmp_cache.pruning_timeout** = 30: minimum inactive time before being eligible for pruning { 1: }
- int **stream.icmp_cache.idle_timeout** = 180: maximum inactive time before retiring session tracker { 1: }
- int **stream.tcp_cache.max_sessions** = 262144: maximum simultaneous sessions tracked before pruning { 2: }
- int **stream.tcp_cache.pruning_timeout** = 30: minimum inactive time before being eligible for pruning { 1: }
- int **stream.tcp_cache.idle_timeout** = 180: maximum inactive time before retiring session tracker { 1: }
- int **stream.udp_cache.max_sessions** = 131072: maximum simultaneous sessions tracked before pruning { 2: }
- int **stream.udp_cache.pruning_timeout** = 30: minimum inactive time before being eligible for pruning { 1: }
- int **stream.udp_cache.idle_timeout** = 180: maximum inactive time before retiring session tracker { 1: }
- int **stream.user_cache.max_sessions** = 1024: maximum simultaneous sessions tracked before pruning { 2: }
- int **stream.user_cache.pruning_timeout** = 30: minimum inactive time before being eligible for pruning { 1: }

- int **stream.user_cache.idle_timeout** = 180: maximum inactive time before retiring session tracker { 1: }
- int **stream.file_cache.max_sessions** = 128: maximum simultaneous sessions tracked before pruning { 2: }
- int **stream.file_cache.pruning_timeout** = 30: minimum inactive time before being eligible for pruning { 1: }
- int **stream.file_cache.idle_timeout** = 180: maximum inactive time before retiring session tracker { 1: }

Peg counts:

- **stream.ip flows**: total ip sessions
- **stream.ip total prunes**: total ip sessions pruned
- **stream.ip idle prunes**: ip sessions pruned due to timeout
- **stream.ip excess prunes**: ip sessions pruned due to excess
- **stream.ip uni prunes**: ip uni sessions pruned
- **stream.ip preemptive prunes**: ip sessions pruned during preemptive pruning
- **stream.ip memcap prunes**: ip sessions pruned due to memcap
- **stream.ip ha prunes**: ip sessions pruned by high availability sync
- **stream.icmp flows**: total icmp sessions
- **stream.icmp total prunes**: total icmp sessions pruned
- **stream.icmp idle prunes**: icmp sessions pruned due to timeout
- **stream.icmp excess prunes**: icmp sessions pruned due to excess
- **stream.icmp uni prunes**: icmp uni sessions pruned
- **stream.icmp preemptive prunes**: icmp sessions pruned during preemptive pruning
- **stream.icmp memcap prunes**: icmp sessions pruned due to memcap
- **stream.icmp ha prunes**: icmp sessions pruned by high availability sync
- **stream.tcp flows**: total tcp sessions
- **stream.tcp total prunes**: total tcp sessions pruned
- **stream.tcp idle prunes**: tcp sessions pruned due to timeout
- **stream.tcp excess prunes**: tcp sessions pruned due to excess
- **stream.tcp uni prunes**: tcp uni sessions pruned
- **stream.tcp preemptive prunes**: tcp sessions pruned during preemptive pruning
- **stream.tcp memcap prunes**: tcp sessions pruned due to memcap
- **stream.tcp ha prunes**: tcp sessions pruned by high availability sync
- **stream.udp flows**: total udp sessions
- **stream.udp total prunes**: total udp sessions pruned
- **stream.udp idle prunes**: udp sessions pruned due to timeout
- **stream.udp excess prunes**: udp sessions pruned due to excess
- **stream.udp uni prunes**: udp uni sessions pruned

- **stream.udp preemptive prunes**: udp sessions pruned during preemptive pruning
- **stream.udp memcap prunes**: udp sessions pruned due to memcap
- **stream.udp ha prunes**: udp sessions pruned by high availability sync
- **stream.user flows**: total user sessions
- **stream.user total prunes**: total user sessions pruned
- **stream.user idle prunes**: user sessions pruned due to timeout
- **stream.user excess prunes**: user sessions pruned due to excess
- **stream.user uni prunes**: user uni sessions pruned
- **stream.user preemptive prunes**: user sessions pruned during preemptive pruning
- **stream.user memcap prunes**: user sessions pruned due to memcap
- **stream.user ha prunes**: user sessions pruned by high availability sync
- **stream.file flows**: total file sessions
- **stream.file total prunes**: total file sessions pruned
- **stream.file idle prunes**: file sessions pruned due to timeout
- **stream.file excess prunes**: file sessions pruned due to excess
- **stream.file uni prunes**: file uni sessions pruned
- **stream.file preemptive prunes**: file sessions pruned during preemptive pruning
- **stream.file memcap prunes**: file sessions pruned due to memcap
- **stream.file ha prunes**: file sessions pruned by high availability sync

8.31 stream_file

What: stream inspector for file flow tracking and processing

Type: inspector

Configuration:

- bool **stream_file.upload** = false: indicate file transfer direction

8.32 stream_icmp

What: stream inspector for ICMP flow tracking

Type: inspector

Configuration:

- int **stream_icmp.session_timeout** = 30: session tracking timeout { 1:86400 }

Peg counts:

- **stream_icmp.sessions**: total icmp sessions
- **stream_icmp.max**: max icmp sessions
- **stream_icmp.created**: icmp session trackers created
- **stream_icmp.released**: icmp session trackers released
- **stream_icmp.timeouts**: icmp session timeouts
- **stream_icmp.prunes**: icmp session prunes

8.33 stream_ip

What: stream inspector for IP flow tracking and defragmentation

Type: inspector

Configuration:

- int **stream_ip.max_frags** = 8192: maximum number of simultaneous fragments being tracked { 1: }
- int **stream_ip.max_overlaps** = 0: maximum allowed overlaps per datagram; 0 is unlimited { 0: }
- int **stream_ip.min_frag_length** = 0: alert if fragment length is below this limit before or after trimming { 0: }
- int **stream_ip.min_ttl** = 1: discard fragments with ttl below the minimum { 1:255 }
- enum **stream_ip.policy** = linux: fragment reassembly policy { first | linux | bsd | bsd_right | last | windows | solaris }
- int **stream_ip.session_timeout** = 30: session tracking timeout { 1:86400 }
- int **stream_ip.trace**: mask for enabling debug traces in module

Rules:

- **123:1** (stream_ip) inconsistent IP options on fragmented packets
- **123:2** (stream_ip) teardrop attack
- **123:3** (stream_ip) short fragment, possible DOS attempt
- **123:4** (stream_ip) fragment packet ends after defragmented packet
- **123:5** (stream_ip) zero-byte fragment packet
- **123:6** (stream_ip) bad fragment size, packet size is negative
- **123:7** (stream_ip) bad fragment size, packet size is greater than 65536
- **123:8** (stream_ip) fragmentation overlap
- **123:11** (stream_ip) TTL value less than configured minimum, not using for reassembly
- **123:12** (stream_ip) excessive fragment overlap
- **123:13** (stream_ip) tiny fragment

Peg counts:

- **stream_ip.sessions**: total ip sessions
- **stream_ip.max**: max ip sessions
- **stream_ip.created**: ip session trackers created
- **stream_ip.released**: ip session trackers released
- **stream_ip.timeouts**: ip session timeouts
- **stream_ip.prunes**: ip session prunes
- **stream_ip.total frags**: total fragments
- **stream_ip.current frags**: current fragments
- **stream_ip.max frags**: max fragments

- **stream_ip.reassembled**: reassembled datagrams
- **stream_ip.discards**: fragments discarded
- **stream_ip.frag timeouts**: datagrams abandoned
- **stream_ip.overlaps**: overlapping fragments
- **stream_ip.anomalies**: anomalies detected
- **stream_ip.alerts**: alerts generated
- **stream_ip.drops**: fragments dropped
- **stream_ip.trackers added**: datagram trackers created
- **stream_ip.trackers freed**: datagram trackers released
- **stream_ip.trackers cleared**: datagram trackers cleared
- **stream_ip.trackers completed**: datagram trackers completed
- **stream_ip.nodes inserted**: fragments added to tracker
- **stream_ip.nodes deleted**: fragments deleted from tracker
- **stream_ip.memory used**: current memory usage in bytes
- **stream_ip.reassembled bytes**: total reassembled bytes
- **stream_ip.fragmented bytes**: total fragmented bytes

8.34 stream_tcp

What: stream inspector for TCP flow tracking and stream normalization and reassembly

Type: inspector

Configuration:

- int **stream_tcp.flush_factor** = 0: flush upon seeing a drop in segment size after given number of non-decreasing segments { 0: }
- bool **stream_tcp.ignore_any_rules** = false: process tcp content rules w/o ports only if rules with ports are present
- int **stream_tcp.max_window** = 0: maximum allowed tcp window { 0:1073725440 }
- int **stream_tcp.overlap_limit** = 0: maximum number of allowed overlapping segments per session { 0:255 }
- int **stream_tcp.max_pdu** = 16384: maximum reassembled PDU size { 1460:65535 }
- enum **stream_tcp.policy** = bsd: determines operating system characteristics like reassembly { first | last | linux | old_linux | bsd | macos | solaris | irix | hpux11 | hpux10 | windows | win_2003 | vista | proxy }
- bool **stream_tcp.reassemble_async** = true: queue data for reassembly before traffic is seen in both directions
- int **stream_tcp.require_3whs** = -1: don’t track midstream sessions after given seconds from start up; -1 tracks all { -1:86400 }
- bool **stream_tcp.show_rebuilt_packets** = false: enable cmg like output of reassembled packets
- int **stream_tcp.queue_limit.max_bytes** = 1048576: don’t queue more than given bytes per session and direction { 0: }
- int **stream_tcp.queue_limit.max_segments** = 2621: don’t queue more than given segments per session and direction { 0: }
- int **stream_tcp.small_segments.count** = 0: limit number of small segments queued { 0:2048 }

- int **stream_tcp.small_segments.maximum_size** = 0: limit number of small segments queued { 0:2048 }
- int **stream_tcp.session_timeout** = 30: session tracking timeout { 1:86400 }
- int **stream_tcp.footprint** = 0: use zero for production, non-zero for testing at given size { 0: }

Rules:

- **129:1** (stream_tcp) SYN on established session
- **129:2** (stream_tcp) data on SYN packet
- **129:3** (stream_tcp) data sent on stream not accepting data
- **129:4** (stream_tcp) TCP timestamp is outside of PAWS window
- **129:5** (stream_tcp) bad segment, adjusted size ⇐ 0 (deprecated)
- **129:6** (stream_tcp) window size (after scaling) larger than policy allows
- **129:7** (stream_tcp) limit on number of overlapping TCP packets reached
- **129:8** (stream_tcp) data sent on stream after TCP reset sent
- **129:9** (stream_tcp) TCP client possibly hijacked, different ethernet address
- **129:10** (stream_tcp) TCP server possibly hijacked, different ethernet address
- **129:11** (stream_tcp) TCP data with no TCP flags set
- **129:12** (stream_tcp) consecutive TCP small segments exceeding threshold
- **129:13** (stream_tcp) 4-way handshake detected
- **129:14** (stream_tcp) TCP timestamp is missing
- **129:15** (stream_tcp) reset outside window
- **129:16** (stream_tcp) FIN number is greater than prior FIN
- **129:17** (stream_tcp) ACK number is greater than prior FIN
- **129:18** (stream_tcp) data sent on stream after TCP reset received
- **129:19** (stream_tcp) TCP window closed before receiving data
- **129:20** (stream_tcp) TCP session without 3-way handshake

Peg counts:

- **stream_tcp.sessions**: total tcp sessions
- **stream_tcp.max**: max tcp sessions
- **stream_tcp.created**: tcp session trackers created
- **stream_tcp.released**: tcp session trackers released
- **stream_tcp.timeouts**: tcp session timeouts
- **stream_tcp.prunes**: tcp session prunes
- **stream_tcp.resyns**: SYN received on established session
- **stream_tcp.discards**: tcp packets discarded
- **stream_tcp.events**: events generated

- **stream_tcp.ignored**: tcp packets ignored
- **stream_tcp.untracked**: tcp packets not tracked
- **stream_tcp.syn trackers**: tcp session tracking started on syn
- **stream_tcp.syn-ack trackers**: tcp session tracking started on syn-ack
- **stream_tcp.3way trackers**: tcp session tracking started on ack
- **stream_tcp.data trackers**: tcp session tracking started on data
- **stream_tcp.segs queued**: total segments queued
- **stream_tcp.segs released**: total segments released
- **stream_tcp.segs split**: tcp segments split when reassembling PDUs
- **stream_tcp.segs used**: queued tcp segments applied to reassembled PDUs
- **stream_tcp.rebuilt packets**: total reassembled PDUs
- **stream_tcp.rebuilt buffers**: rebuilt PDU sections
- **stream_tcp.rebuilt bytes**: total rebuilt bytes
- **stream_tcp.overlaps**: overlapping segments queued
- **stream_tcp.gaps**: missing data between PDUs
- **stream_tcp.max segs**: number of times the maximum queued segment limit was reached
- **stream_tcp.max bytes**: number of times the maximum queued byte limit was reached
- **stream_tcp.internal events**: 135:X events generated
- **stream_tcp.client cleanups**: number of times data from server was flushed when session released
- **stream_tcp.server cleanups**: number of times data from client was flushed when session released
- **stream_tcp.memory**: current memory in use
- **stream_tcp.initializing**: number of sessions currently initializing
- **stream_tcp.established**: number of sessions currently established
- **stream_tcp.closing**: number of sessions currently closing

8.35 stream_udp

What: stream inspector for UDP flow tracking

Type: inspector

Configuration:

- int **stream_udp.session_timeout** = 30: session tracking timeout { 1:86400 }
- bool **stream_udp.ignore_any_rules** = false: process udp content rules w/o ports only if rules with ports are present

Peg counts:

- **stream_udp.sessions**: total udp sessions
- **stream_udp.max**: max udp sessions
- **stream_udp.created**: udp session trackers created
- **stream_udp.released**: udp session trackers released
- **stream_udp.timeouts**: udp session timeouts
- **stream_udp.prunes**: udp session prunes

8.36 stream_user

What: stream inspector for user flow tracking and reassembly

Type: inspector

Configuration:

- int **stream_user.session_timeout** = 30: session tracking timeout { 1:86400 }

8.37 telnet

What: telnet inspection and normalization

Type: inspector

Configuration:

- int **telnet.ayt_attack_thresh** = -1: alert on this number of consecutive telnet AYT commands { -1: }
- bool **telnet.check_encrypted** = false: check for end of encryption
- bool **telnet.encrypted_traffic** = false: check for encrypted telnet and ftp
- bool **telnet.normalize** = false: eliminate escape sequences

Rules:

- **126:1** (telnet) consecutive telnet AYT commands beyond threshold
- **126:2** (telnet) telnet traffic encrypted
- **126:3** (telnet) telnet subnegotiation begin command without subnegotiation end

Peg counts:

- **telnet.packets**: total packets

8.38 wizard

What: inspector that implements port-independent protocol identification

Type: inspector

Configuration:

- string **wizard.hexes[].service**: name of service
- select **wizard.hexes[].proto** = tcp: protocol to scan { tcp | udp }
- bool **wizard.hexes[].client_first** = true: which end initiates data transfer
- string **wizard.hexes[].to_server[].hex**: sequence of data with wild chars (?)
- string **wizard.hexes[].to_client[].hex**: sequence of data with wild chars (?)
- string **wizard.spells[].service**: name of service
- select **wizard.spells[].proto** = tcp: protocol to scan { tcp | udp }
- bool **wizard.spells[].client_first** = true: which end initiates data transfer

- string **wizard.spells[].to_server[].spell**: sequence of data with wild cards (*)
- string **wizard.spells[].to_client[].spell**: sequence of data with wild cards (*)
- multi **wizard.curses**: enable service identification based on internal algorithm { dce_smb | dce_udp | dce_tcp }

Peg counts:

- **wizard.tcp scans**: tcp payload scans
- **wizard.tcp hits**: tcp identifications
- **wizard.udp scans**: udp payload scans
- **wizard.udp hits**: udp identifications
- **wizard.user scans**: user payload scans
- **wizard.user hits**: user identifications

9 IPS Action Modules

IPS actions allow you to perform custom actions when events are generated. Unlike loggers, these are invoked before thresholding and can be used to control external agents.

Externally defined actions must be configured to become available to the parser. For the reject rule, you can set reject = { } to get the rule to parse.

9.1 react

What: send response to client and terminate session

Type: ips_action

Configuration:

- bool **react.msg** = false: use rule msg in response page instead of default message
- string **react.page**: file containing HTTP response (headers and body)

9.2 reject

What: terminate session with TCP reset or ICMP unreachable

Type: ips_action

Configuration:

- enum **reject.reset**: send tcp reset to one or both ends { source|dest|both }
- enum **reject.control**: send icmp unreachable(s) { network|host|port|all }

9.3 rewrite

What: overwrite packet contents

Type: ips_action

10 IPS Option Modules

IPS options are the building blocks of IPS rules.

10.1 ack

What: rule option to match on TCP ack numbers

Type: ips_option

Configuration:

- string **ack.~range**: check if tcp ack value is *value* | *min<>max* | *<max* | *>min*

10.2 appids

What: detection option for application ids

Type: ips_option

Configuration:

- string **appids.~**: appid option

10.3 asn1

What: rule option for asn1 detection

Type: ips_option

Configuration:

- implied **asn1.bitstring_overflow**: detects invalid bitstring encodings that are known to be remotely exploitable
- implied **asn1.double_overflow**: detects a double ASCII encoding that is larger than a standard buffer
- implied **asn1.print**: dump decode data to console; always true
- int **asn1.oversize_length**: compares ASN.1 type lengths with the supplied argument { 0: }
- int **asn1.absolute_offset**: absolute offset from the beginning of the packet { 0: }
- int **asn1.relative_offset**: relative offset from the cursor

10.4 base64_decode

What: rule option to decode base64 data - must be used with base64_data option

Type: ips_option

Configuration:

- int **base64_decode.bytes**: number of base64 encoded bytes to decode { 1: }
- int **base64_decode.offset** = 0: bytes past start of buffer to start decoding { 0: }
- implied **base64_decode.relative**: apply offset to cursor instead of start of buffer

10.5 bufferlen

What: rule option to check length of current buffer

Type: ips_option

Configuration:

- string **bufferlen.~range**: len | min<>max | <max | >min

10.6 byte_extract

What: rule option to convert data to an integer variable

Type: ips_option

Configuration:

- int **byte_extract.~count**: number of bytes to pick up from the buffer { 1:10 }
- int **byte_extract.~offset**: number of bytes into the buffer to start processing { -65535:65535 }
- string **byte_extract.~name**: name of the variable that will be used in other rule options
- implied **byte_extract.relative**: offset from cursor instead of start of buffer
- int **byte_extract.multiplier** = 1: scale extracted value by given amount { 1:65535 }
- int **byte_extract.align** = 0: round the number of converted bytes up to the next 2- or 4-byte boundary { 0:4 }
- implied **byte_extract.big**: big endian
- implied **byte_extract.little**: little endian
- implied **byte_extract.dce**: dcerpc2 determines endianness
- implied **byte_extract.string**: convert from string
- implied **byte_extract.hex**: convert from hex string
- implied **byte_extract.oct**: convert from octal string
- implied **byte_extract.dec**: convert from decimal string

10.7 byte_jump

What: rule option to move the detection cursor

Type: ips_option

Configuration:

- int **byte_jump.~count**: number of bytes to pick up from the buffer { 1:10 }
- string **byte_jump.~offset**: variable name or number of bytes into the buffer to start processing
- implied **byte_jump.relative**: offset from cursor instead of start of buffer
- implied **byte_jump.from_beginning**: jump from start of buffer instead of cursor
- int **byte_jump.multiplier** = 1: scale extracted value by given amount { 1:65535 }
- int **byte_jump.align** = 0: round the number of converted bytes up to the next 2- or 4-byte boundary { 0:4 }

- int **byte_jump.post_offset** = 0: also skip forward or backwards (positive of negative value) this number of bytes { -65535:65535 }
- implied **byte_jump.big**: big endian
- implied **byte_jump.little**: little endian
- implied **byte_jump.dce**: dcerpc2 determines endianness
- implied **byte_jump.string**: convert from string
- implied **byte_jump.hex**: convert from hex string
- implied **byte_jump.oct**: convert from octal string
- implied **byte_jump.dec**: convert from decimal string

10.8 byte_test

What: rule option to convert data to integer and compare

Type: ips_option

Configuration:

- int **byte_test.~count**: number of bytes to pick up from the buffer { 1:10 }
- string **byte_test.~operator**: variable name or number of bytes into the buffer to start processing
- string **byte_test.~compare**: variable name or value to test the converted result against
- string **byte_test.~offset**: variable name or number of bytes into the payload to start processing
- implied **byte_test.relative**: offset from cursor instead of start of buffer
- implied **byte_test.big**: big endian
- implied **byte_test.little**: little endian
- implied **byte_test.dce**: dcerpc2 determines endianness
- implied **byte_test.string**: convert from string
- implied **byte_test.hex**: convert from hex string
- implied **byte_test.oct**: convert from octal string
- implied **byte_test.dec**: convert from decimal string

10.9 classtype

What: general rule option for rule classification

Type: ips_option

Configuration:

- string **classtype.~**: classification for this rule

10.10 content

What: payload rule option for basic pattern matching

Type: ips_option

Configuration:

- string **content.~data**: data to match
- implied **content.nocase**: case insensitive match
- implied **content.fast_pattern**: use this content in the fast pattern matcher instead of the content selected by default
- int **content.fast_pattern_offset** = 0: number of leading characters of this content the fast pattern matcher should exclude { 0: }
- int **content.fast_pattern_length**: maximum number of characters from this content the fast pattern matcher should use { 1: }
- string **content.offset**: var or number of bytes from start of buffer to start search
- string **content.depth**: var or maximum number of bytes to search from beginning of buffer
- string **content.distance**: var or number of bytes from cursor to start search
- string **content.within**: var or maximum number of bytes to search from cursor

10.11 cvs

What: payload rule option for detecting specific attacks

Type: ips_option

Configuration:

- implied **cvs.invalid-entry**: looks for an invalid Entry string

10.12 dce_iface

What: detection option to check dcerpc interface

Type: ips_option

Configuration:

- string **dce_iface.uuid**: match given dcerpc uuid
- string **dce_iface.version**: interface version
- implied **dce_iface.any_frag**: match on any fragment

10.13 dce_opnum

What: detection option to check dcerpc operation number

Type: ips_option

Configuration:

- string **dce_opnum.~**: match given dcerpc operation number, range or list

10.14 dce_stub_data

What: sets the cursor to dcerpc stub data

Type: ips_option

10.15 detection_filter

What: rule option to require multiple hits before a rule generates an event

Type: ips_option

Configuration:

- enum **detection_filter.track**: track hits by source or destination IP address { by_src | by_dst }
- int **detection_filter.count**: hits in interval before allowing the rule to fire { 1: }
- int **detection_filter.seconds**: length of interval to count hits { 1: }

10.16 dnp3_data

What: sets the cursor to dnp3 data

Type: ips_option

10.17 dnp3_func

What: detection option to check dnp3 function code

Type: ips_option

Configuration:

- string **dnp3_func.~**: match dnp3 function code or name

10.18 dnp3_ind

What: detection option to check dnp3 indicator flags

Type: ips_option

Configuration:

- string **dnp3_ind.~**: match given dnp3 indicator flags

10.19 dnp3_obj

What: detection option to check dnp3 object headers

Type: ips_option

Configuration:

- int **dnp3_obj.group** = 0: match given dnp3 object header group { 0:255 }
- int **dnp3_obj.var** = 0: match given dnp3 object header var { 0:255 }

10.20 dsize

What: rule option to test payload size

Type: ips_option

Configuration:

- string **dsize.~range**: check if packet payload size is *size* | *min*<>*max* | <*max* | >*min*

10.21 file_data

What: rule option to set detection cursor to file data

Type: ips_option

10.22 file_type

What: rule option to check file type

Type: ips_option

Configuration:

- string **file_type.~**: list of file type IDs to match

10.23 flags

What: rule option to test TCP control flags

Type: ips_option

Configuration:

- string **flags.~test_flags**: these flags are tested
- string **flags.~mask_flags**: these flags are don’t cares

10.24 flow

What: rule option to check session properties

Type: ips_option

Configuration:

- implied **flow.to_client**: match on server responses
- implied **flow.to_server**: match on client requests
- implied **flow.from_client**: same as to_server
- implied **flow.from_server**: same as to_client
- implied **flow.established**: match only during data transfer phase
- implied **flow.not_established**: match only outside data transfer phase
- implied **flow.stateless**: match regardless of stream state
- implied **flow.no_stream**: match on raw packets only
- implied **flow.only_stream**: match on reassembled packets only
- implied **flow.no_frag**: match on raw packets only
- implied **flow.only_frag**: match on defragmented packets only

10.25 flowbits

What: rule option to set and test arbitrary boolean flags

Type: ips_option

Configuration:

- string **flowbits.~command**: set|reset|isset|etc.
- string **flowbits.~arg1**: bits or group
- string **flowbits.~arg2**: group if arg1 is bits

10.26 fragbits

What: rule option to test IP frag flags

Type: ips_option

Configuration:

- string **fragbits.~flags**: these flags are tested

10.27 fragoffset

What: rule option to test IP frag offset

Type: ips_option

Configuration:

- string **fragoffset.~range**: check if ip fragment offset value is *value* | *min*<>*max* | <*max* | >*min*

10.28 gid

What: rule option specifying rule generator

Type: ips_option

Configuration:

- int **gid.~**: generator id { 1: }

10.29 gtp_info

What: rule option to check gtp info element

Type: ips_option

Configuration:

- string **gtp_info.~**: info element to match

10.30 gtp_type

What: rule option to check gtp types

Type: ips_option

Configuration:

- string **gtp_type.~**: list of types to match

10.31 gtp_version

What: rule option to check gtp version

Type: ips_option

Configuration:

- int **gtp_version.~**: version to match { 0:2 }

10.32 http_client_body

What: rule option to set the detection cursor to the request body

Type: ips_option

10.33 http_cookie

What: rule option to set the detection cursor to the HTTP cookie

Type: ips_option

Configuration:

- implied **http_cookie.request**: match against the cookie from the request message even when examining the response
- implied **http_cookie.with_body**: parts of this rule examine HTTP message body
- implied **http_cookie.with_trailer**: parts of this rule examine HTTP message trailers

10.34 http_header

What: rule option to set the detection cursor to the normalized headers

Type: ips_option

Configuration:

- string **http_header.field**: restrict to given header. Header name is case insensitive.
- implied **http_header.request**: match against the headers from the request message even when examining the response
- implied **http_header.with_body**: parts of this rule examine HTTP message body
- implied **http_header.with_trailer**: parts of this rule examine HTTP message trailers

10.35 http_method

What: rule option to set the detection cursor to the HTTP request method

Type: ips_option

Configuration:

- implied **http_method.with_body**: parts of this rule examine HTTP message body
- implied **http_method.with_trailer**: parts of this rule examine HTTP message trailers

10.36 http_raw_cookie

What: rule option to set the detection cursor to the unnormalized cookie

Type: ips_option

Configuration:

- implied **http_raw_cookie.request**: match against the cookie from the request message even when examining the response
- implied **http_raw_cookie.with_body**: parts of this rule examine HTTP message body
- implied **http_raw_cookie.with_trailer**: parts of this rule examine HTTP message trailers

10.37 http_raw_header

What: rule option to set the detection cursor to the unnormalized headers

Type: ips_option

Configuration:

- implied **http_raw_header.request**: match against the headers from the request message even when examining the response
- implied **http_raw_header.with_body**: parts of this rule examine HTTP message body
- implied **http_raw_header.with_trailer**: parts of this rule examine HTTP message trailers

10.38 http_raw_request

What: rule option to set the detection cursor to the unnormalized request line

Type: ips_option

Configuration:

- implied **http_raw_request.with_body**: parts of this rule examine HTTP message body
- implied **http_raw_request.with_trailer**: parts of this rule examine HTTP message trailers

10.39 http_raw_status

What: rule option to set the detection cursor to the unnormalized status line

Type: ips_option

Configuration:

- implied **http_raw_status.with_body**: parts of this rule examine HTTP message body
- implied **http_raw_status.with_trailer**: parts of this rule examine HTTP message trailers

10.40 http_raw_trailer

What: rule option to set the detection cursor to the unnormalized trailers

Type: ips_option

Configuration:

- implied **http_raw_trailer.request**: match against the trailers from the request message even when examining the response
- implied **http_raw_trailer.with_header**: parts of this rule examine HTTP response message headers (must be combined with request)
- implied **http_raw_trailer.with_body**: parts of this rule examine HTTP response message body (must be combined with request)

10.41 http_raw_uri

What: rule option to set the detection cursor to the unnormalized URI

Type: ips_option

Configuration:

- implied **http_raw_uri.with_body**: parts of this rule examine HTTP message body
- implied **http_raw_uri.with_trailer**: parts of this rule examine HTTP message trailers
- implied **http_raw_uri.scheme**: match against scheme section of URI only
- implied **http_raw_uri.host**: match against host section of URI only
- implied **http_raw_uri.port**: match against port section of URI only
- implied **http_raw_uri.path**: match against path section of URI only
- implied **http_raw_uri.query**: match against query section of URI only
- implied **http_raw_uri.fragment**: match against fragment section of URI only

10.42 http_stat_code

What: rule option to set the detection cursor to the HTTP status code

Type: ips_option

Configuration:

- implied **http_stat_code.with_body**: parts of this rule examine HTTP message body
- implied **http_stat_code.with_trailer**: parts of this rule examine HTTP message trailers

10.43 http_stat_msg

What: rule option to set the detection cursor to the HTTP status message

Type: ips_option

Configuration:

- implied **http_stat_msg.with_body**: parts of this rule examine HTTP message body
- implied **http_stat_msg.with_trailer**: parts of this rule examine HTTP message trailers

10.44 http_trailer

What: rule option to set the detection cursor to the normalized trailers

Type: ips_option

Configuration:

- string **http_trailer.field**: restrict to given trailer
- implied **http_trailer.request**: match against the trailers from the request message even when examining the response
- implied **http_trailer.with_header**: parts of this rule examine HTTP response message headers (must be combined with request)
- implied **http_trailer.with_body**: parts of this rule examine HTTP message body (must be combined with request)

10.45 http_uri

What: rule option to set the detection cursor to the normalized URI buffer

Type: ips_option

Configuration:

- implied **http_uri.with_body**: parts of this rule examine HTTP message body
- implied **http_uri.with_trailer**: parts of this rule examine HTTP message trailers
- implied **http_uri.scheme**: match against scheme section of URI only
- implied **http_uri.host**: match against host section of URI only
- implied **http_uri.port**: match against port section of URI only
- implied **http_uri.path**: match against path section of URI only
- implied **http_uri.query**: match against query section of URI only
- implied **http_uri.fragment**: match against fragment section of URI only

10.46 http_version

What: rule option to set the detection cursor to the version buffer

Type: ips_option

Configuration:

- implied **http_version.request**: match against the version from the request message even when examining the response
- implied **http_version.with_body**: parts of this rule examine HTTP message body
- implied **http_version.with_trailer**: parts of this rule examine HTTP message trailers

10.47 icmp_id

What: rule option to check ICMP ID

Type: ips_option

Configuration:

- string **icmp_id.~range**: check if icmp id is *id* | *min*<>*max* | <*max* | >*min*

10.48 icmp_seq

What: rule option to check ICMP sequence number

Type: ips_option

Configuration:

- string **icmp_seq.~range**: check if icmp sequence number is *seq* | *min*<>*max* | <*max* | >*min*

10.49 icode

What: rule option to check ICMP code

Type: ips_option

Configuration:

- string **icode.~range**: check if ICMP code is *code* | *min<>max* | *<max* | *>min*

10.50 id

What: rule option to check the IP ID field

Type: ips_option

Configuration:

- string **id.~range**: check if the IP ID is *id* | *min<>max* | *<max* | *>min*

10.51 ip_proto

What: rule option to check the IP protocol number

Type: ips_option

Configuration:

- string **ip_proto.~proto**: [!|>|<] name or number

10.52 ipopts

What: rule option to check for IP options

Type: ips_option

Configuration:

- select **ipopts.~opt**: output format { rr|eol|nop|ts|sec|esec|lsrr|lsrre|ssrr|satid|any }

10.53 isdataat

What: rule option to check for the presence of payload data

Type: ips_option

Configuration:

- string **isdataat.~length**: num | !num
- implied **isdataat.relative**: offset from cursor instead of start of buffer

10.54 itype

What: rule option to check ICMP type

Type: ips_option

Configuration:

- string **itype.~range**: check if icmp type is *type* | *min<>max* | *<max* | *>min*

10.55 md5

What: payload rule option for hash matching

Type: ips_option

Configuration:

- string **md5.~hash**: data to match
- int **md5.length**: number of octets in plain text { 1:65535 }
- string **md5.offset**: var or number of bytes from start of buffer to start search
- implied **md5.relative** = false: offset from cursor instead of start of buffer

10.56 metadata

What: rule option for conveying arbitrary name, value data within the rule text

Type: ips_option

Configuration:

- string **metadata.service**: service name
- string **metadata.***: additional parameters not used by snort

10.57 modbus_data

What: rule option to set cursor to modbus data

Type: ips_option

10.58 modbus_func

What: rule option to check modbus function code

Type: ips_option

Configuration:

- string **modbus_func.~**: function code to match

10.59 modbus_unit

What: rule option to check modbus unit ID

Type: ips_option

Configuration:

- int **modbus_unit.~**: modbus unit ID { 0:255 }

10.60 msg

What: rule option summarizing rule purpose output with events

Type: ips_option

Configuration:

- string **msg.~**: message describing rule

10.61 pcre

What: rule option for matching payload data with pcre

Type: ips_option

Configuration:

- string **pcre.~re**: Snort regular expression

10.62 pkt_data

What: rule option to set the detection cursor to the normalized packet data

Type: ips_option

10.63 priority

What: rule option for prioritizing events

Type: ips_option

Configuration:

- int **priority.~**: relative severity level; 1 is highest priority { 1: }

10.64 raw_data

What: rule option to set the detection cursor to the raw packet data

Type: ips_option

10.65 reference

What: rule option to indicate relevant attack identification system

Type: ips_option

Configuration:

- string **reference.~scheme**: reference scheme
- string **reference.~id**: reference id

10.66 regex

What: rule option for matching payload data with hyperscan regex

Type: ips_option

Configuration:

- string **regex.~re**: hyperscan regular expression
- implied **regex.nocase**: case insensitive match
- implied **regex.dotall**: matching a . will not exclude newlines
- implied **regex.multiline**: ˆ and $ anchors match any newlines in data
- implied **regex.relative**: start search from end of last match instead of start of buffer

10.67 rem

What: rule option to convey an arbitrary comment in the rule body

Type: ips_option

Configuration:

- string **rem.~**: comment

10.68 replace

What: rule option to overwrite payload data; use with rewrite action

Type: ips_option

Configuration:

- string **replace.~**: byte code to replace with

10.69 rev

What: rule option to indicate current revision of signature

Type: ips_option

Configuration:

- int **rev.~**: revision { 1: }

10.70 rpc

What: rule option to check SUNRPC CALL parameters

Type: ips_option

Configuration:

- int **rpc.~app**: application number
- int **rpc.ver**: version number or * for any
- int **rpc.proc**: procedure number or * for any

10.71 sd_pattern

What: rule option for detecting sensitive data

Type: ips_option

Configuration:

- string **sd_pattern.~pattern**: The pattern to search for
- int **sd_pattern.threshold**: number of matches before alerting { 1 }

Peg counts:

- **sd_pattern.below threshold**: sd_pattern matched but missed threshold
- **sd_pattern.pattern not found**: sd_pattern did not not match
- **sd_pattern.terminated**: hyperscan terminated

10.72 seq

What: rule option to check TCP sequence number

Type: ips_option

Configuration:

- string **seq.~range**: check if tcp sequence number value is *value* | *min*<>*max* | <*max* | >*min*

10.73 session

What: rule option to check user data from TCP sessions

Type: ips_option

Configuration:

- enum **session.~mode**: output format { printable|binary|all }

10.74 sha256

What: payload rule option for hash matching

Type: ips_option

Configuration:

- string **sha256.~hash**: data to match
- int **sha256.length**: number of octets in plain text { 1:65535 }
- string **sha256.offset**: var or number of bytes from start of buffer to start search
- implied **sha256.relative** = false: offset from cursor instead of start of buffer

10.75 sha512

What: payload rule option for hash matching

Type: ips_option

Configuration:

- string **sha512.~hash**: data to match
- int **sha512.length**: number of octets in plain text { 1:65535 }
- string **sha512.offset**: var or number of bytes from start of buffer to start search
- implied **sha512.relative** = false: offset from cursor instead of start of buffer

10.76 sid

What: rule option to indicate signature number

Type: ips_option

Configuration:

- int **sid.~**: signature id { 1: }

10.77 sip_body

What: rule option to set the detection cursor to the request body

Type: ips_option

10.78 sip_header

What: rule option to set the detection cursor to the SIP header buffer

Type: ips_option

10.79 sip_method

What: detection option for sip stat code

Type: ips_option

Configuration:

- string **sip_method.*method**: sip method

10.80 sip_stat_code

What: detection option for sip stat code

Type: ips_option

Configuration:

- int **sip_stat_code.*code**: stat code { 1:999 }

10.81 so

What: rule option to call custom eval function

Type: ips_option

Configuration:

- string **so.~func**: name of eval function

10.82 soid

What: rule option to specify a shared object rule ID

Type: ips_option

Configuration:

- string **soid.~**: SO rule ID has <gid>|<sid> format, like 3|12345

10.83 ssl_state

What: detection option for ssl state

Type: ips_option

Configuration:

- implied **ssl_state.client_hello**: check for client hello
- implied **ssl_state.server_hello**: check for server hello
- implied **ssl_state.client_keyx**: check for client keyx
- implied **ssl_state.server_keyx**: check for server keyx
- implied **ssl_state.unknown**: check for unknown record
- implied **ssl_state.!client_hello**: check for records that are not client hello
- implied **ssl_state.!server_hello**: check for records that are not server hello
- implied **ssl_state.!client_keyx**: check for records that are not client keyx
- implied **ssl_state.!server_keyx**: check for records that are not server keyx
- implied **ssl_state.!unknown**: check for records that are not unknown

10.84 ssl_version

What: detection option for ssl version

Type: ips_option

Configuration:

- implied **ssl_version.sslv2**: check for sslv2
- implied **ssl_version.sslv3**: check for sslv3
- implied **ssl_version.tls1.0**: check for tls1.0
- implied **ssl_version.tls1.1**: check for tls1.1
- implied **ssl_version.tls1.2**: check for tls1.2
- implied **ssl_version.!sslv2**: check for records that are not sslv2
- implied **ssl_version.!sslv3**: check for records that are not sslv3
- implied **ssl_version.!tls1.0**: check for records that are not tls1.0
- implied **ssl_version.!tls1.1**: check for records that are not tls1.1
- implied **ssl_version.!tls1.2**: check for records that are not tls1.2

10.85 stream_reassemble

What: detection option for stream reassembly control

Type: ips_option

Configuration:

- enum **stream_reassemble.action**: stop or start stream reassembly { disable|enable }
- enum **stream_reassemble.direction**: action applies to the given direction(s) { client|server|both }
- implied **stream_reassemble.noalert**: don’t alert when rule matches
- implied **stream_reassemble.fastpath**: optionally whitelist the remainder of the session

10.86 stream_size

What: detection option for stream size checking

Type: ips_option

Configuration:

- string **stream_size.~range**: size for comparison
- enum **stream_size.~direction**: compare applies to the given direction(s) { either|to_server|to_client|both }

10.87 tag

What: rule option to log additional packets

Type: ips_option

Configuration:

- enum **tag.~**: log all packets in session or all packets to or from host { session|host_src|host_dst }
- int **tag.packets**: tag this many packets { 1: }
- int **tag.seconds**: tag for this many seconds { 1: }
- int **tag.bytes**: tag for this many bytes { 1: }

10.88 tos

What: rule option to check type of service field

Type: ips_option

Configuration:

- string **tos.~range**: check if ip tos value is *value* | *min*<>*max* | <*max* | >*min*

10.89 ttl

What: rule option to check time to live field

Type: ips_option

Configuration:

- string **ttl.~range**: check if ip ttl field value is *value* | *min*<>*max* | <*max* | >*min*

10.90 window

What: rule option to check TCP window field

Type: ips_option

Configuration:

- string **window.~range**: check if tcp window field size is *size* | *min*<>*max* | <*max* | >*min*

11 Search Engine Modules

Search engines perform multipattern searching of packets and payload to find rules that should be evaluated. There are currently no specific modules, although there are several search engine plugins. Related configuration is done with the basic detection module.

12 SO Rule Modules

SO rules are dynamic rules that require custom coding to perform detection not possible with the existing rule options. These rules typically do not have associated modules.

13 Logger Modules

All output of events and packets is done by Loggers.

13.1 alert_csv

What: output event in csv format

Type: logger

Configuration:

- bool **alert_csv.file** = false: output to alert_csv.txt instead of stdout
- multi **alert_csv.fields** = timestamp pkt_num proto pkt_gen dgm_len dir src_ap dst_ap rule action: selected fields will be output in given order left to right { action | dir | dgm_len | dst_addr | dst_ap | dst_port | eth_dst | eth_len | eth_src | eth_type | gid | icmp_code | icmp_id | icmp_seq | icmp_type | iface | ip_id | ip_len | msg | pkt_gen | pkt_num | proto | rev | rule | sid | src_addr | src_ap | src_port | tcp_ack | tcp_flags | tcp_len | tcp_seq | tcp_win | timestamp | tos | ttl | udp_len }
- int **alert_csv.limit** = 0: set limit (0 is unlimited) { 0: }
- string **alert_csv.separator** = , : separate fields with this character sequence
- enum **alert_csv.units** = B: bytes | KB | MB | GB { B | K | M | G }

13.2 alert_fast

What: output event with brief text format

Type: logger

Configuration:

- bool **alert_fast.file** = false: output to alert_fast.txt instead of stdout
- bool **alert_fast.packet** = false: output packet dump with alert
- int **alert_fast.limit** = 0: set limit (0 is unlimited) { 0: }
- enum **alert_fast.units** = B: bytes | KB | MB | GB { B | K | M | G }

13.3 alert_full

What: output event with full packet dump

Type: logger

Configuration:

- bool **alert_full.file** = false: output to alert_full.txt instead of stdout
- int **alert_full.limit** = 0: set limit (0 is unlimited) { 0: }
- enum **alert_full.units** = B: limit is in bytes | KB | MB | GB { B | K | M | G }

13.4 alert_sfsocket

What: output event over socket

Type: logger

Configuration:

- string **alert_sfsocket.file**: name of unix socket file
- int **alert_sfsocket.rules[].gid** = 1: rule generator ID { 1: }
- int **alert_sfsocket.rules[].sid** = 1: rule signature ID { 1: }

13.5 alert_syslog

What: output event to syslog

Type: logger

Configuration:

- enum **alert_syslog.facility** = auth: part of priority applied to each message { auth | authpriv | daemon | user | local0 | local1 | local2 | local3 | local4 | local5 | local6 | local7 }
- enum **alert_syslog.level** = info: part of priority applied to each message { emerg | alert | crit | err | warning | notice | info | debug }
- multi **alert_syslog.options**: used to open the syslog connection { cons | ndelay | perror | pid }

13.6 log_codecs

What: log protocols in packet by layer

Type: logger

Configuration:

- bool **log_codecs.file** = false: output to log_codecs.txt instead of stdout
- bool **log_codecs.msg** = false: include alert msg

13.7 log_hext

What: output payload suitable for daq hext

Type: logger

Configuration:

- bool **log_hext.file** = false: output to log_hext.txt instead of stdout
- bool **log_hext.raw** = false: output all full packets if true, else just TCP payload
- int **log_hext.limit** = 0: set limit (0 is unlimited) { 0: }
- enum **log_hext.units** = B: bytes | KB | MB | GB { B | K | M | G }
- int **log_hext.width** = 20: set line width (0 is unlimited) { 0: }

13.8 log_pcap

What: log packet in pcap format

Type: logger

Configuration:

- int **log_pcap.limit** = 0: set limit (0 is unlimited) { 0: }
- enum **log_pcap.units** = B: bytes | KB | MB | GB { B | K | M | G }

13.9 unified2

What: output event and packet in unified2 format file

Type: logger

Configuration:

- int **unified2.limit** = 0: set limit (0 is unlimited) { 0: }
- enum **unified2.units** = B: limit multiplier { B | K | M | G }
- bool **unified2.nostamp** = true: append file creation time to name (in Unix Epoch format)
- bool **unified2.mpls_event_types** = false: include mpls labels in events
- bool **unified2.vlan_event_types** = false: include vlan IDs in events

14 DAQ Modules

The Data AcQuisition library (DAQ), provides pluggable packet I/O. The DAQ replaces direct calls to libraries like libpcap with an abstraction layer that facilitates operation on a variety of hardware and software interfaces without requiring changes to Snort. It is possible to select the DAQ type and mode when invoking Snort to perform pcap readback or inline operation, etc. The DAQ library may be useful for other packet processing applications and the modular nature allows you to build new modules for other platforms.

The DAQ library is provided as an external package on snort.org. There are a few additional modules provided with Snort 3. This section summarizes the important things you need to know to use these DAQ modules. There are also 3rd DAQ modules available.

14.1 Building the DAQ Library and DAQ Modules

The DAQ is bundled with Snort but must be built first using these steps:

```
./configure
make
sudo make install
```

This will build and install both static and dynamic DAQ modules.

Note that pcap >= 1.0.0 is required. pcap 1.1.1 is available at the time of this writing and is recommended.

Also, libdnet is required for IPQ and NFQ DAQs. If you get a relocation error trying to build those DAQs, you may need to reinstall libdnet and configure it with something like this:

```
./configure "CFLAGS=-fPIC -g -O2"
```

You may also experience problems trying to find the dynamic dnet library because it isn't always named properly. Try creating a link to the shared library (identified by its .x or .x.y etc. extension) with the same name but with ".so" inserted as follows:

```
$ ln -s libdnet.1.1 libdnet.so.1.1
$ ldconfig -Rv /usr/local/lib 2>&1 | grep dnet
  Adding /usr/local/lib/libdnet.so.1.1
```

Alternatively, you should be able to fix both issues as follows:

```
libtoolize --copy --force
aclocal -I config
autoheader
autoconf
automake --foreign
```

When the DAQ library is built, both static and dynamic flavors will be generated. The various DAQ modules will be built if the requisite headers and libraries are available. You can disable individual modules, etc. with options to configure. For the complete list of configure options, run:

```
./configure --help
```

14.2 PCAP Module

pcap is the default DAQ. If snort is run w/o any DAQ arguments, it will operate as it always did using this module. These are equivalent:

```
./snort -i <device>
./snort -r <file>

./snort --daq pcap --daq-mode passive -i <device>
./snort --daq pcap --daq-mode read-file -r <file>
```

You can specify the buffer size pcap uses with:

```
./snort --daq pcap --daq-var buffer_size=<#bytes>
```

- The pcap DAQ does not count filtered packets. *

14.3 AFPACKET Module

afpacket functions similar to the pcap DAQ but with better performance:

```
./snort --daq afpacket -i <device>
        [--daq-var buffer_size_mb=<#MB>]
        [--daq-var debug]
```

If you want to run afpacket in inline mode, you must craft the device string as one or more interface pairs, where each member of a pair is separated by a single colon and each pair is separated by a double colon like this:

```
eth0:eth1
```

or this:

```
eth0:eth1::eth2:eth3
```

By default, the afpacket DAQ allocates 128MB for packet memory. You can change this with:

```
--daq-var buffer_size_mb=<#MB>
```

Note that the total allocated is actually higher, here’s why. Assuming the default packet memory with a snaplen of 1518, the numbers break down like this:

- The frame size is 1518 (snaplen) + the size of the AFPacket header (66 bytes) = 1584 bytes.
- The number of frames is 128 MB / 1518 = 84733.
- The smallest block size that can fit at least one frame is 4 KB = 4096 bytes @ 2 frames per block.
- As a result, we need 84733 / 2 = 42366 blocks.
- Actual memory allocated is 42366 * 4 KB = 165.5 MB.

Note
Linux kernel version 2.6.31 or higher is required for the AFPacket DAQ module due to its dependency on both TPACKET v2 and PACKET_TX_RING support.

14.4 NFQ Module

NFQ is the new and improved way to process iptables packets:

```
./snort --daq nfq \
    [--daq-var device=<dev>] \
    [--daq-var proto=<proto>] \
    [--daq-var queue=<qid>]

<dev> ::= ip | eth0, etc; default is IP injection
<proto> ::= ip4 | ip6 |; default is ip4
<qid> ::= 0..65535; default is 0
```

This module can not run unprivileged so ./snort -u -g will produce a warning and won’t change user or group.

Notes on iptables are given below.

14.5 IPQ Module

IPQ is the old way to process iptables packets. It replaces the inline version available in pre-2.9 versions built with this:

```
./configure --enable-inline
```

Note that layer 2 resets are not supported with the IPQ DAQ:

```
config layer2resets[: <mac>]
```

Start the IPQ DAQ as follows:

```
./snort --daq ipq \
    [--daq-var device=<dev>] \
    [--daq-var proto=<proto>] \

<dev> ::= ip | eth0, etc; default is IP injection
<proto> ::= ip4 | ip6; default is ip4
```

This module can not run unprivileged so ./snort -u -g will produce a warning and won't change user or group.

Notes on iptables are given below.

14.6 IPFW Module

IPFW is available for BSD systems. It replaces the inline version available in pre-2.9 versions built with this:

```
./configure --enable-ipfw
```

This command line argument is no longer supported:

```
./snort -J <port#>
```

Instead, start Snort like this:

```
./snort --daq ipfw [--daq-var port=<port>]

<port> ::= 1..65535; default is 8000
```

- IPFW only supports ip4 traffic.

Notes on FreeBSD and OpenBSD are given below.

14.7 Dump Module

The dump DAQ allows you to test the various inline mode features available in Snort like injection and normalization.

```
./snort -i <device> --daq dump
./snort -r <pcap> --daq dump
```

By default a file named inline-out.pcap will be created containing all packets that passed through or were generated by snort. You can optionally specify a different name.

```
./snort --daq dump --daq-var file=<name>
```

dump uses the pcap daq for packet acquisition. It therefore does not count filtered packets (a pcap limitation).

Note that the dump DAQ inline mode is not an actual inline mode. Furthermore, you will probably want to have the pcap DAQ acquire in another mode like this:

```
./snort -r <pcap> -Q --daq dump --daq-var load-mode=read-file
./snort -i <device> -Q --daq dump --daq-var load-mode=passive
```

14.8 Netmap Module

The netmap project is a framework for very high speed packet I/O. It is available on both FreeBSD and Linux with varying amounts of preparatory setup required. Specific notes for each follow.

```
./snort --daq netmap -i <device>
        [--daq-var debug]
```

If you want to run netmap in inline mode, you must craft the device string as one or more interface pairs, where each member of a pair is separated by a single colon and each pair is separated by a double colon like this:

```
em1:em2
```

or this:

```
em1:em2::em3:em4
```

Inline operation performs Layer 2 forwarding with no MAC filtering, akin to the AFPacket module's behavior. All packets received on one interface in an inline pair will be forwarded out the other interface unless dropped by the reader and vice versa.

Important
The interfaces will need to be up and in promiscuous mode in order to function (*ifconfig em1 up promisc*). The DAQ module does not currently do either of these configuration steps for itself.

14.8.1 FreeBSD

In FreeBSD 10.0, netmap has been integrated into the core OS. In order to use it, you must recompile your kernel with the line

```
device netmap
```

added to your kernel config.

14.8.2 Linux

You will need to download the netmap source code from the project's repository:

```
https://code.google.com/p/netmap/
```

Follow the instructions on the project's homepage for compiling and installing the code:

```
http://info.iet.unipi.it/~luigi/netmap/
```

It will involve a standalone kernel module (netmap_lin) as well as patching and rebuilding the kernel module used to drive your network adapters. The following drivers are supported under Linux at the time of writing (June 2014):

```
e1000
e1000e
forcedeth
igb
ixgbe
r8169
virtio
```

TODO:

- Support for attaching to only a single ring (queue) on a network adapter.
- Support for VALE and netmap pipes.

14.9 Notes on iptables

These notes are just a quick reminder that you need to set up iptables to use the IPQ or NFQ DAQs. Doing so may cause problems with your network so tread carefully. The examples below are intentionally incomplete so please read the related documentation first.

Here is a blog post by Marty for historical reference:

```
http://archives.neohapsis.com/archives/snort/2000-11/0394.html
```

You can check this out for queue sizing tips:

```
http://www.inliniac.net/blog/2008/01/23/improving-snort_inlines-nfq-performance. ↩
    html
```

You might find useful IPQ info here:

```
http://snort-inline.sourceforge.net/
```

Use this to examine your iptables:

```
sudo /sbin/iptables -L
```

Use something like this to set up NFQ:

```
sudo /sbin/iptables
    -I <table> [<protocol stuff>] [<state stuff>]
    -j NFQUEUE --queue-num 1
```

Use something like this to set up IPQ:

```
sudo iptables -I FORWARD -j QUEUE
```

Use something like this to "disconnect" snort:

```
sudo /sbin/iptables -D <table> <rule pos>
```

Be sure to start Snort prior to routing packets through NFQ with iptables. Such packets will be dropped until Snort is started.

The queue-num is the number you must give Snort.

If you are running on a system with both NFQ and IPQ support, you may experience some start-up failures of the sort:

The solution seems to be to remove both modules from the kernel like this:

```
modprobe -r nfnetlink_queue
modprobe -r ip_queue
```

and then install the module you want:

```
modprobe ip_queue
```

or:

```
modprobe nfnetlink_queue
```

These DAQs should be run with a snaplen of 65535 since the kernel defrags the packets before queuing. Also, no need to configure frag3.

14.10 Notes on FreeBSD::IPFW

Check the online manual at:

```
http://www.freebsd.org/doc/handbook/firewalls-ipfw.html.
```

Here is a brief example to divert icmp packets to Snort at port 8000:

To enable support for divert sockets, place the following lines in the kernel configuration file:

```
options IPFIREWALL
options IPDIVERT
```

(The file in this case was: /usr/src/sys/i386/conf/GENERIC; which is platform dependent.)

You may need to also set these to use the loadable kernel modules:

```
/etc/rc.conf:
firewall_enable="YES"

/boot/loader.conf:
ipfw_load="YES"
ipdivert_load="YES"

$ dmesg | grep ipfw
ipfw2 (+ipv6) initialized, divert loadable, nat loadable, rule-based
forwarding disabled, default to deny, logging disabled

$ kldload -v ipdivert
Loaded ipdivert, id=4

$ ipfw add 75 divert 8000 icmp from any to any
00075 divert 8000 icmp from any to any

$ ipfw list
...
00075 divert 8000 icmp from any to any
00080 allow icmp from any to any
...
```

- Note that on FreeBSD, divert sockets don't work with bridges!

Please refer to the following articles for more information:

- https://forums.snort.org/forums/support/topics/snort-inline-on-freebsd-ipfw
- http://freebsd.rogness.net/snort_inline/

NAT gateway can be used with divert sockets if the network environment is conducive to using NAT.

The steps to set up NAT with ipfw are as follows:

1. Set up NAT with two interface em0 and em1 by adding the following to /etc/rc.conf. Here em0 is connected to external network and em1 to host-only LAN.

```
gateway_enable="YES"
natd_program="/sbin/natd"   # path to natd
natd_enable="YES"           # Enable natd (if firewall_enable == YES)
natd_interface="em0"       # Public interface or IP Address
natd_flags="-dynamic"       # Additional flags
defaultrouter=""
ifconfig_em0="DHCP"
ifconfig_em1="inet 192.168.1.2 netmask 255.255.255.0"
firewall_enable="YES"
firewall_script="/etc/rc.firewall"
firewall_type="simple"
```

2. Add the following divert rules to divert packets to Snort above and below the NAT rule in the "Simple" section of /etc/rc.firewall.

```
...
# Inspect outbound packets (those arriving on "inside" interface)
# before NAT translation.
${fwcmd} add divert 8000 all from any to any in via ${iif}
case ${natd_enable} in
[Yy][Ee][Ss])
    if [ -n "${natd_interface}" ]; then
        ${fwcmd} add divert natd all from any to any via ${natd_interface}
    fi
    ;;
esac
...
# Inspect inbound packets (those arriving on "outside" interface)
# after NAT translation that aren’t blocked for other reasons,
# after the TCP "established" rule.
${fwcmd} add divert 8000 all from any to any in via ${oif}
```

14.11 Notes on OpenBSD::IPFW

OpenBSD supports divert sockets as of 4.7, so we use the ipfw DAQ.

Here is one way to set things up:

1. Configure the system to forward packets:

```
$ sysctl net.inet.ip.forwarding=1
$ sysctl net.inet6.ip6.forwarding=1

(You can also put that in /etc/sysctl.conf to enable on boot.)
```

2. Set up interfaces

```
$ dhclient vic1
$ dhclient vic2
```

3. Set up packet filter rules:

```
$ echo "pass out on vic1 divert-packet port 9000 keep-state" > rules.txt
$ echo "pass out on vic2 divert-packet port 9000 keep-state" >> rules.txt
$ pfctl -v -f rules.txt
```

4. Analyze packets diverted to port 9000:

   ```
   $ ./snort --daq ipfw --daq-var port=9000
   ```

   - Note that on OpenBSD, divert sockets don’t work with bridges!

14.12 Socket Module

The socket module provides provides a stream socket server that will accept up to 2 simultaneous connections and bridge them together while also passing data to Snort for inspection. The first connection accepted is considered the client and the second connection accepted is considered the server. If there is only one connection, stream data can’t be forwarded but it is still inspected.

Each read from a socket of up to snaplen bytes is passed as a packet to Snort along with a DAQ_SktHdr_t pointer in DAQ_PktHdr_t→priv DAQ_SktHdr_t conveys IP4 address, ports, protocol, and direction. Socket packets can be configured to be TCP or UDP. The socket DAQ can be operated in inline mode and is able to block packets.

The socket DAQ uses DLT_SOCKET and requires that Snort load the socket codec which is included in the extra package.

To use the socket DAQ, start Snort like this:

```
./snort --plugin-path /path/to/lib/snort_extra \
    --daq socket [--daq-var port=<port>] [--daq-var proto=<proto>] [-Q]

<port> ::= 1..65535; default is 8000
<proto> ::= tcp | udp
```

- This module only supports ip4 traffic.
- This module is only supported by Snort 3. It is not compatible with Snort 2.
- This module is primarily for development and test.

14.13 File Module

The file module provides the ability to process files directly w/o having to extract them from pcaps. Use the file module with Snort’s stream_file to get file type identification and signature services. The usual IPS detection and logging etc. is available too.

You can process all the files in a directory recursively using 8 threads with these Snort options:

```
--pcap-dir path -z 8
```

- This module is only supported by Snort 3. It is not compatible with Snort 2.
- This module is primarily for development and test.

14.14 Hext Module

The hext module generates packets suitable for processing by Snort from hex/plain text. Raw packets include full headers and are processed normally. Otherwise the packets contain only payload and are accompanied with flow information (4-tuple) suitable for processing by stream_user.

The first character of the line determines it's purpose:

```
'$' command
'#' comment
'"' quoted string packet data
'x' hex packet data
' ' empty line separates packets
```

The available commands are:

```
$client <ip4> <port>
$server <ip4> <port>

$packet -> client
$packet -> server

$packet <addr> <port> -> <addr> <port>
```

Client and server are determined as follows. $packet → client indicates to the client (from server) and $packet → server indicates a packet to the server (from client). $packet followed by a 4-tuple uses the heuristic that the client is the side with the lower port number.

The default client and server are 192.168.1.1 12345 and 10.1.2.3 80 respectively. $packet commands with a 4-tuple do not change client and server set with the other $packet commands.

$packet commands should be followed by packet data, which may contain any combination of hex and strings. Data for a packet ends with the next command or a blank line. Data after a blank line will start another packet with the same tuple as the prior one.

Strings may contain the following escape sequences:

```
\r = 0x0D = carriage return
\n = 0x0A = new line
\t = 0x09 = tab
\\ = 0x5C = \
```

Format your input carefully; there is minimal error checking and little tolerance for arbitrary whitespace. You can use Snort's -L hext option to generate hext input from a pcap.

- This module only supports ip4 traffic.
- This module is only supported by Snort 3. It is not compatible with Snort 2.
- This module is primarily for development and test.

The hext DAQ also supports a raw mode which is activated by setting the data link type. For example, you can input full ethernet packets with --daq-var dlt=1 (Data link types are defined in the DAQ include sfbpf_dlt.h.) Combine that with the hext logger in raw mode for a quick (and dirty) way to edit pcaps. With --lua "log_hext = { raw = true }", the hext logger will dump the full packet in a way that can be read by the hext DAQ in raw mode. Here is an example:

```
# 3 [96]
```

```
x02 09 08 07 06 05 02 01 02 03 04 05 08 00 45 00 00 52 00 03  # ..............E..R ↩
    ..
x00 00 40 06 5C 90 0A 01 02 03 0A 09 08 07 BD EC 00 50 00 00  # ..@.\...........P ↩
    ..
x00 02 00 00 00 02 50 10 20 00 8A E1 00 00 47 45 54 20 2F 74  # ......P.  .....GET ↩
     /t
x72 69 67 67 65 72 2F 31 20 48 54 54 50 2F 31 2E 31 0D 0A 48  # rigger/1 HTTP ↩
    /1.1..H
x6F 73 74 3A 20 6C 6F 63 61 6C 68 6F 73 74 0D 0A              # ost: localhost..
```

A comment indicating packet number and size precedes each packet dump. Note that the commands are not applicable in raw mode and have no effect.

15 Snort 3 vs Snort 2

Snort 3 differs from Snort 2 in the following ways:

- command line and conf file syntax made more uniform
- removed unused and deprecated features
- remove as many barriers to successful run as possible (e.g.: no upper bounds on memcaps)
- assume the simplest mode of operation (e.g.: never assume input from or output to some hardcoded filename)
- all Snort 2 config options are grouped into Snort 3 modules

15.1 Build Options

- configure --with-lib{pcap,pcre}-* → --with-{pcap,pcre}-*
- control socket, cs_dir, and users were deleted
- POLICY_BY_ID_ONLY code was deleted
- hardened --enable-inline-init-failopen / INLINE_FAILOPEN

15.2 Command Line

- --pause loads config and waits for resume before processing packets
- --require-rule-sid is hardened
- --shell enables interactive Lua shell
- -T is assumed if no input given
- added --help-config prefix to dump all matching settings
- added --script-path
- added -L none|dump|pcap
- added -z <#> and --max-packet-threads <#>
- delete --enable-mpls-multicast, --enable-mpls-overlapping-ip, --max-mpls-labelchain-len, --mpls-payload-type
- deleted --pid-path and --no-interface-pidfile

- deleting command line options which will be available with --lua or some such including: -I, -h, -F, -p, --disable-inline-init-failopen
- hardened -n < 0
- removed --search-method
- replaced "unknown args are bpf" with --bpf
- replaced --dynamic-*-lib[-dir] with --plugin-path (with : separators)
- removed -b, -N, -Z and, --perfmon-file options

15.3 Conf File

- Snort 3 has a default unicode.map
- Snort 3 will not enforce an upper bound on memcaps and the like within 64 bits
- Snort 3 will supply a default *_global config if not specified (Snort 2 would fatal; e.g. http_inspect_server w/o http_inspect_global)
- address list syntax changes: [[and]] must be [[and]] to avoid Lua string parsing errors (unless in quoted string)
- because the Lua conf is live code, we lose file:line locations in app error messages (syntax errors from Lua have file:line)
- changed search-method names for consistency
- delete config include_vlan_in_alerts (not used in code)
- delete config so_rule_memcap (not used in code)
- deleted --disable-attribute-table-reload-thread
- deleted config decode_*_{alerts,drops} (use rules only)
- deleted config dump-dynamic-rules-path
- deleted config ipv6_frag (not actually used)
- deleted config threshold and ips rule threshold (→ event_filter)
- eliminated ac-split; must use ac-full-q split-any-any
- frag3 → defrag, arpspoof → arp_spoof, sfportscan → port_scan, perfmonitor → perf_monitor, bo → back_orifice
- limits like "1234K" are now "limit = 1234, units = *K*"
- lua field names are (lower) case sensitive; snort.conf largely wasn't
- module filenames are not configurable: always <log-dir>/<module-name><suffix> (suffix is determined by module)
- no positional parameters; all name = value
- perf_monitor configuration was simplified
- portscan.detect_ack_scans deleted (exact same as include_midstream)
- removed various run modes - now just one
- frag3 default policy is Linux not bsd
- lowmem* search methods are now in snort_examples
- deleted unused http_inspect stateful mode
- deleted stateless inspection from ftp and telnet

- deleted http and ftp alert options (now strictly rule based)
- preprocessor disabled settings deleted since no longer relevant
- sessions are always created; snort config stateful checks eliminated
- stream5_tcp: prune_log_max deleted; to be replaced with histogram
- stream5_tcp: max_active_responses, min_response_seconds moved to active.max_responses, min_interval

15.4 Rules

- all rules must have a sid
- deleted activate / dynamic rules
- deleted metadata engine shared
- deleted metadata: rule-flushing (with PDU flushing rule flushing can cause missed attacks, the opposite of its intent)
- deleted unused rule_state.action
- fastpattern_offset, fast_pattern_length
- no ; separated content suboptions
- offset, depth, distance, and within must use a space separator not colon (e.g. offset:5; becomes offset 5;)
- rule option sequence: <stub> soid <hidden>
- sid == 0 not allowed
- soid is now a non-metadata option
- content suboptions http_* are now full options and should be place before content
- the following pcre options have been deleted: use sticky buffers instead B, U, P, H, M, C, I, D, K, S, Y
- deleted uricontent ips rule option. uricontent:"foo" -→ http_uri; content:"foo"
- deleted urilen raw and norm; must use http_raw_uri and http_uri instead
- deleted unused http_encode option
- urilen replaced with generic bufferlen which applies to current sticky buffer
- added optional selector to http_header, e.g. http_header:User-Agent;
- multiline rules w/o \n
- #begin … #end comments

15.5 Output

- alert_fast includes packet data by default
- all text mode outputs default to stdout
- changed default logging mode to -L none
- deleted layer2resets and flexresp2_*
- deleted log_ascii
- general output guideline: don’t print zero counts

- Snort 3 queues decoder and inspector events to the main event queue before ips policy is selected; since some events may not be enabled, the queue needs to be sized larger than with Snort 2 which used an intermediate queue for decoder events.
- deleted the intermediate http and ftp_telnet event queues
- alert_unified2 and log_unified2 have been deleted

15.6 HTTP Profiles

This section describes the changes to the Http Inspect config option "profile".

Snort 2 allows users to select pre-defined HTTP server profiles using the config option "profile". The user can choose one of five predefined profiles. When defined, this option will set defaults for other config options within Http Inspect.

With Snort 3, the user has the flexibility of defining and fine tuning custom profiles along with the five predefined profiles.

Snort 2 conf

```
preprocessor http_inspect_server: server default \
    profile apache ports { 80 3128 } max_headers 200
```

Snort 3 conf

```
http_inspect = { profile = http_profile_apache }
http_inspect.profile.max_headers = 200

binder =
{
    {
        when = { proto = 'tcp', ports = '80 3128', },
        use = { type = 'http_inspect' },
    },
}
```

Note
The "profile" option now that points to a table "http_profile_apache" which is defined in "snort_defaults.lua" (as follows).

```
http_profile_apache =
{
    profile_type = 'apache',
    server_flow_depth = 300,
    client_flow_depth = 300,
    post_depth = -1,
    chunk_length = 500000,
    ascii = true,
    multi_slash = true,
    directory = true,
    webroot = true,
    utf_8 = true,
    apache_whitespace = true,
    non_strict = true,
    normalize_utf = true,
    normalize_javascript = false,
    max_header_length = 0,
    max_headers = 0,
    max_spaces = 200,
    max_javascript_whitespaces = 200,
    whitespace_chars ='0x9 0xb 0xc 0xd'
}
```

Note
The config option "max_headers" is set to 0 in the profile, but overwritten by "http_inspect.profile.max_headers = 200".

Conversion

snort2lua can convert the existing snort.conf with the "profile" option to Snort 3 compatible "profile". Please refer to the snort2Lua post for more details.

Examples

```
"profile all" ==> "profile = http_profile_default"
"profile apache" ==> "profile = http_profile_apache"
"profile iis" ==> "profile = http_profile_iis"
"profile iis_40" ==> "profile = http_profile_iis_40"
"profile iis_50" ==> "profile = http_profile_iis_50"
```

Defining custom profiles

The complete set of Http Inspect config options that a custom profile can configure can be found by running the following command:

```
snort --help-config http_inspect | grep http_inspect.profile
```

16 Snort2Lua

One of the major differences between Snort 2 and Snort 3 is the configuration. Snort 2 configuration files are written in Snort-specific syntax while Snort 3 configuration files are written in Lua. Snort2Lua is a program specifically designed to convert Snort 2 configuration files into Lua files that Snort 3 can understand.

Snort2Lua reads your legacy Snort conf file(s) and generates Snort 3 Lua and rules files. When running this program, the only mandatory option is to provide Snort2Lua with a Snort 2 configuration file. The default output file file is snort.lua, the default error file will be snort.rej, and the default rule file is the output file (default is snort.lua). When Snort2Lua finishes running, the resulting configuration file can be successfully run as the Snort3.0 configuration file. The sole exception to this rule is when Snort2Lua cannot find an included file. If that occurs, the file will still be included in the output file and you will need to manually adjust or comment the file name. Additionally, if the exit code is not zero, some of the information may not be successfully converted. Check the error file for all of the conversion problems.

Those errors can occur for a multitude of reasons and are not necessarily bad. For instance, Snort2Lua will only convert preprocessors that are currently supported. Therefore, any unsupported preprocessors or configuration options including DCERP, SIP, and SMTP, will cause an error in Snort2Lua since Snort 3 does not support those preprocessors. Additionally, any rule options associated with those preprocessors are also not supported. Finally, Snort2Lua expects a valid Snort 2 configuration. Therefore, if the configuration is invalid or has questionable syntax, Snort2Lua may fail to parse the configuration file or create an invalid Snort 3 configuration file.

There are a also few peculiarities of Snort2Lua that may be confusing to a first time user. Specifically, aside from an initial configuration file (which is specified from the command line or as the file in ‘config binding’), every file that is included into Snort 3 must be either a Lua file or a rule file; the file cannot contain both rules and Lua syntax. Therefore, when parsing a file specified with the ‘include’ command, Snort2Lua will output both a Lua file and a rule file. Additionally, any line that is a comment in a configuration file will be added in to a comments section at the bottom of the main configuration file. Finally, rules that contain unsupported options will be converted to the best of Snort2Lua’s capability and then printed as a comment in the rule file.

16.1 Snort2Lua Command Line

By default, Snort2Lua will attempt to parse every ‘include’ file and every ‘binding’ file. There is an option to change this functionality.

When specifying a rule file with one of the command line options, Snort2Lua will output all of the converted rules to that specified rule file. This is especially useful when you are only interesting in converting rules since there is no Lua syntax in rule files. There is also an option that tells Snort2Lua to output every rule for a given configuration into a single rule file. Similarly, there is an option pull all of the Lua syntax from every ‘include’ file into the output file.

There are currently three output modes: default, quiet, and differences. As expected, quiet mode produces a Snort configuration. All errors (aside from Fatal Snort2Lua errors), differences, and comments will omitted from the final output file. Default mode will print everything. That mean you will be able to see exactly what changes have occurred between Snort 2 and Snort 3 in addition to the new syntax, the original file’s comments, and all errors that have occurred. Finally, differences mode will not actually output a valid Snort 3 configuration. Instead, you can see the exact options from the input configuration that have changed.

16.1.1 Usage: snort2lua [OPTIONS]... -c <snort_conf> ...

Converts the Snort configuration file specified by the -c or --conf-file options into a Snort++ configuration file

Options:

- **-?** show usage
- **-h** this overview of snort2lua
- **-a** default option. print all data
- **-c <snort_conf>** The Snort <snort_conf> file to convert
- **-d** print the differences, and only the differences, between the Snort and Snort++ configurations to the <out_file>
- **-e <error_file>** output all errors to <error_file>
- **-i** if <snort_conf> file contains any <include_file> or <policy_file> (i.e. *include path/to/conf/other_conf*), do NOT parse those files
- **-m** add a remark to the end of every converted rule
- **-o <out_file>** output the new Snort++ lua configuration to <out_file>
- **-q** quiet mode. Only output valid confiration information to the <out_file>
- **-r <rule_file>** output any converted rule to <rule_file>
- **-s** when parsing <include_file>, write <include_file>’s rules to <rule_file>. Meaningles if *-i* provided
- **-t** when parsing <include_file>, write <include_file>’s information, excluding rules, to <out_file>. Meaningles if *-i* provided
- **-V** Print the current Snort2Lua version
- **--conf-file** Same as *-c*. A Snort <snort_conf> file which will be converted
- **--dont-parse-includes** Same as *-p*. if <snort_conf> file contains any <include_file> or <policy_file> (i.e. *include path/to/conf/other_conf*), do NOT parse those files
- **--error-file=<error_file>** Same as *-e*. output all errors to <error_file>
- **--help** Same as *-h*. this overview of snort2lua
- **--markup** print help in asciidoc compatible format

- **--ohi** Use Old Http Inspect format
- **--output-file=<out_file>** Same as *-o*. output the new Snort++ lua configuration to <out_file>
- **--print-all** Same as *-a*. default option. print all data
- **--print-differences** Same as *-d*. output the differences, and only the differences, between the Snort and Snort++ configurations to the <out_file>
- **--quiet** Same as *-q*. quiet mode. Only output valid confiration information to the <out_file>
- **--remark** same as *-m*. add a remark to the end of every converted rule
- **--rule-file=<rule_file>** Same as *-r*. output any converted rule to <rule_file>
- **--single-conf-file** Same as *-t*. when parsing <include_file>, write <include_file>'s information, excluding rules, to <out_file>
- **--single-rule-file** Same as *-s*. when parsing <include_file>, write <include_file>'s rules to <rule_file>.
- **--version** Same as *-V*. Print the current Snort2Lua version

Required option:

- A Snort configuration file to convert. Set with either *-c* or *--conf-file*

Default values:

- <out_file> = snort.lua
- <rule_file> = <out_file> = snort.lua. Rules are written to the *local_rules* variable in the <out_file>
- <error_file> = snort.rej. This file will not be created in quiet mode.

16.2 Known Problems

- Any Snort 2 'string' which is dependent on a variable will no longer have that variable in the Lua string.
- Snort2Lua currently does not handle variables well. First, that means variables will not always be parsed correctly. Second, sometimes a variables value will be outoput in the lua file rather than a variable For instance, if Snort2Lua attempted to convert the line *include $RULE_PATH/example.rule*, the output may ouput *include /etc/rules/example.rule* instead.
- When Snort2Lua parses a 'binding' configuration file, the rules and configuration will automatically be combined into the same file. Also, the new files name will automatically become the old file's name with a .lua extension. There is currently no way to specify or change that files name.
- If a rule's action is a custom ruletype, that rule action will be silently converted to the rultype's *type*. No warnings or errors are currently emmitted. Additionally, the custom ruletypes outputs will be silently discarded.
- If the original configuration contains a binding that points to another file and the binding file contains an error, Snort2Lua will output the number of rejects for the binding file in addition to the number of rejects in the main file. The two numbers will eventually be combined into one output.

16.3 Usage

Snort2Lua is included in the Snort 3 distribution. The Snort2Lua source code is located in the tools/snort2lua directory. The program is automatically built and installed.

Translating your configuration

To run Snort2Lua, the only requirement is a file containing Snort 2 syntax. Assuming your configuration file is named snort.conf, run the command

```
snort2lua -c snort.conf
```

Snort2Lua will output a file named snort.lua. Assuming your snort.conf file is a valid Snort 2 configuration file, than the resulting snort.lua file will always be a valid Snort 3 configuration file; any errors that occur are because Snort 3 currently does not support all of the Snort 2 options.

Every keyword from the Snort configuration can be found in the output file. If the option or keyword has changed, then a comment containing both the option or keyword's old name and new name will be present in the output file.

Translating a rule file

Snort2Lua can also accommodate translating individual rule files. Assuming the Snort 2 rule file is named snort.rules and you want the new rule file to be name updated.rules, run the command

```
snort2lua -c snort.rules -r updated.rules
```

Snort2Lua will output a file named updated.rules. That file, updated.rules, will always be a valid Snort 3 rule file. Any rule that contains unsupported options will be a comment in the output file.

Understanding the Output

Although Snort2Lua outputs very little to the console, there are several things that occur when Snort2Lua runs. This is a list of Snort2Lua outputs.

The console. Every line that Snort2Lua is unable to translate from the Snort 2.X format to the Snort 3 format is considered an error. Upon exiting, Snort2Lua will print the number of errors that occurred. Snort2Lua will also print the name of the error file.

The output file. As previously mentioned, Snort2Lua will create a Lua file with valid Snort 3 syntax. The default Lua file is named snort.lua. This file is the equivalent of your main Snort 2 configuration file.

The rule file. By default, all rules will be printed to the Lua file. However, if a rule file is specified on the command line, any rules found in the Snort 2 configuration will be written to the rule file instead

The error file. By default, the error file is snort.rej. It will only be created if errors exist. Every error referenced on the command line can be found in this file. There are two reasons an error can occur.

- The Snort 2 configuration file has invalid syntax. If Snort 2 cannot parse the configuration file, neither can Snort2Lua. In the example below, Snort2Lua could not convert the line *config bad_option*. Since that is not valid Snort 2 syntax, this is a syntax error.
- The Snort 2 configuration file contains preprocessors and rule options that are not supported in Snort 3. If Snort 2 can parse a line that Snort2Lua cannot parse, than Snort 3 does not support something in the line. As Snort 3 begins supporting these preprocessors and rule options, Snort2Lua will also begin translating these lines. One example of such an error is dcerpc2.

Additional .lua and .rules files. Every time Snort2Lua parses the include or binding keyword, the program will attempt to parse the file referenced by the keyword. Snort2Lua will then create one or two new files. The new files will have a .lua or .rules extension appended to the original filename.

17 Extending Snort

17.1 Plugins

Plugins have an associated API defined for each type, all of which share a common *header*, called the BaseApi. A dynamic library makes its plugins available by exporting the snort_plugins symbol, which is a null terminated array of BaseApi pointers.

The BaseApi includes type, name, API version, plugin version, and function pointers for constructing and destructing a Module. The specific API add various other data and functions for their given roles.

17.2 Modules

If we are defining a new Inspector called, say, gadget, it might be configured in snort.lua like this:

```
gadget =
{
    brain = true,
    claw = 3
}
```

When the gadget table is processed, Snort will look for a module called gadget. If that Module has an associated API, it will be used to configure a new instance of the plugin. In this case, a GadgetModule would be instantiated, brain and claw would be set, and the Module instance would be passed to the GadgetInspector constructor.

Module has three key virtual methods:

- **begin()** - called when Snort starts processing the associated Lua table. This is a good place to allocate any required data and set defaults.
- **set()** - called to set each parameter after validation.
- **end()** - called when Snort finishes processing the associated Lua table. This is where additional integrity checks of related parameters should be done.

The configured Module is passed to the plugin constructor which pulls the configuration data from the Module. For non-trivial configurations, the working paradigm is that Module hands a pointer to the configured data to the plugin instance which takes ownership.

Note that there is at most one instance of a given Module, even if multiple plugin instances are created which use that Module. (Multiple instances require Snort binding configuration.)

17.3 Inspectors

There are several types of inspector, which determines which inspectors are executed when:

- IT_BINDER - determines which inspectors apply to given flows
- IT_WIZARD - determines which service inspector to use if none explicitly bound
- IT_PACKET - used to process all packets before session and service processing (e.g. normalize)
- IT_NETWORK - processes packets w/o service (e.g. arp_spoof, back_orifice)
- IT_STREAM - for flow tracking, ip defrag, and tcp reassembly
- IT_SERVICE - for http, ftp, telnet, etc.
- IT_PROBE - process all packets after all the above (e.g. perf_monitor, port_scan)

17.4 Codecs

The Snort Codecs decipher raw packets. These Codecs are now completely pluggable; almost every Snort Codec can be built dynamically and replaced with an alternative, customized Codec. The pluggable nature has also made it easier to build new Codecs for protocols without having to touch the Snort code base.

The first step in creating a Codec is defining its class and protocol. Every Codec must inherit from the Snort Codec class defined in "framework/codec.h". The following is an example Codec named "example" and has an associated struct that is 14 bytes long.

```
#include <cstdint>
#include <arpa/inet.h>
#include “framework/codec.h”
#include "main/snort_types.h"

#define EX_NAME “example”
#define EX_HELP “example codec help string”

struct Example
{
    uint8_t dst[6];
    uint8_t src[6];
    uint16_t ethertype;

    static inline uint8_t size()
    { return 14; }
}

class ExCodec : public Codec
{
public:
    ExCodec() : Codec(EX_NAME) { }
    ~ExCodec() { }

    bool decode(const RawData&, CodecData&, DecodeData&) override;
    void get_protocol_ids(std::vector<uint16_t>&) override;
};
```

After defining ExCodec, the next step is adding the Codec’s decode functionality. The function below does this by implementing a valid decode function. The first parameter, which is the RawData struct, provides both a pointer to the raw data that has come from a wire and the length of that raw data. The function takes this information and validates that there are enough bytes for this protocol. If the raw data’s length is less than 14 bytes, the function returns false and Snort discards the packet; the packet is neither inspected nor processed. If the length is greater than 14 bytes, the function populates two fields in the CodecData struct, next_prot_id and lyr_len. The lyr_len field tells Snort the number of bytes that this layer contains. The next_prot_id field provides Snort the value of the next EtherType or IP protocol number.

```
bool ExCodec::decode(const RawData& raw, CodecData& codec, DecodeData&)
{
   if ( raw.len < Example::size() )
       return false;

    const Example* const ex = reinterpret_cast<const Example*>(raw.data);
    codec.next_prot_id = ntohs(ex->ethertype);
    codec.lyr_len = ex->size();
    return true;
}
```

For instance, assume this decode function receives the following raw data with a validated length of 32 bytes:

```
00 11 22 33 44 55 66 77    88 99 aa bb 08 00 45 00
00 38 00 01 00 00 40 06    5c ac 0a 01 02 03 0a 09
```

The Example struct's EtherType field is the 13 and 14 bytes. Therefore, this function tells Snort that the next protocol has an EtherType of 0x0800. Additionally, since the lyr_len is set to 14, Snort knows that the next protocol begins 14 bytes after the beginning of this protocol. The Codec with EtherType 0x0800, which happens to be the IPv4 Codec, will receive the following data with a validated length of 18 (== 32 – 14):

```
45 00 00 38 00 01 00 00    40 06 5c ac 0a 01 02 03
0a 09
```

How does Snort know that the IPv4 Codec has an EtherType of 0x0800? The Codec class has a second virtual function named get_protocol_ids(). When implementing the function, a Codec can register for any number of values between 0x0000 - 0xFFFF. Then, if the next_proto_id is set to a value for which this Codec has registered, this Codec's decode function will be called. As a general note, the protocol ids between [0, 0x00FF] are IP protocol numbers, [0x0100, 0x05FF] are custom types, and [0x0600, 0xFFFF] are EtherTypes.

For example, in the get_protocol_ids function below, the ExCodec registers for the protocols numbers 17, 787, and 2054. 17 happens to be the protocol number for UDP while 2054 is ARP's EtherType. Therefore, this Codec will now attempt to decode UDP and ARP data. Additionally, if any Codec sets the next_protocol_id to 787, ExCodec's decode function will be called. Some custom protocols are already defined in the file "protocols/protocol_ids.h"

```
void ExCodec::get_protocol_ids(std::vector<uint16_t>&v)
{
    v.push_back(0x0011); // == 17  == UDP
    v.push_back(0x1313); // == 787  == custom
    v.push_back(0x0806); // == 2054  == ARP
}
```

To register a Codec for Data Link Type's rather than protocols, the function get_data_link_type() can be similarly implemented.

The final step to creating a pluggable Codec is the snort_plugins array. This array is important because when Snort loads a dynamic library, the program only find plugins that are inside the snort_plugins array. In other words, if a plugin has not been added to the snort_plugins array, that plugin will not be loaded into Snort.

Although the details will not be covered in this post, the following code snippet is a basic CodecApi that Snort can load. This snippet can be copied and used with only three minor changes. First, in the function ctor, ExCodec should be replaced with the name of the Codec that is being built. Second, EX_NAME must match the Codec's name or Snort will be unable to load this Codec. Third, EX_HELP should be replaced with the general description of this Codec. Once this code snippet has been added, ExCodec is ready to be compiled and plugged into Snort.

```
static Codec* ctor(Module*)
{ return new ExCodec; }


static void dtor(Codec *cd)
{ delete cd; }


static const CodecApi ex_api =
{
    {
        PT_CODEC,
        EX_NAME,
        EX_HELP,
        CDAPI_PLUGIN_V0,
        0,
        nullptr,
        nullptr,
    },
    nullptr, // pointer to a function called during Snort's startup.
    nullptr, // pointer to a function called during Snort's exit.
    nullptr, // pointer to a function called during thread's startup.
    nullptr, // pointer to a function called during thread's destruction.
    ctor, // pointer to the codec constructor.
    dtor, // pointer to the codec destructor.
};

SO_PUBLIC const BaseApi* snort_plugins[] =
{
    &ex_api.base,
    nullptr
};
```

Two example Codecs are available in the extra directory on git and the extra tarball on the Snort page. One of those examples is the Token Ring Codec while the other example is the PIM Codec.

As a final note, there are four more virtual functions that a Codec should implement: encode, format, update, and log. If the functions are not implemented Snort will not throw any errors. However, Snort may also be unable to accomplish some of its basic functionality.

- encode is called whenever Snort actively responds and needs to builds a packet, i.e. whenever a rule using an IPS ACTION like react, reject, or rewrite is triggered. This function is used to build the response packet protocol by protocol.
- format is called when Snort is rebuilding a packet. For instance, every time Snort reassembles a TCP stream or IP fragment, format is called. Generally, this function either swaps any source and destination fields in the protocol or does nothing.
- update is similar to format in that it is called when Snort is reassembling a packet. Unlike format, this function only sets length fields.
- log is called when either the log_codecs logger or a custom logger that calls PacketManager::log_protocols is used when running Snort.

17.5 IPS Actions

Action plugins specify a builtin action in the API which is used to determine verdict. (Conversely, builtin actions don't have an associated plugin function.)

17.6 Developers Guide

Run doc/dev_guide.sh to generate /tmp/dev_guide.html, an annotated guide to the source tree.

17.7 Piglet Test Harness

In order to assist with plugin development, an experimental mode called "piglet" mode is provided. With piglet mode, you can call individual methods for a specific plugin. The piglet tests are specified as Lua scripts. Each piglet test script defines a test for a specific plugin.

Here is a minimal example of a piglet test script for the IPv4 Codec plugin:

```
plugin =
{
    type = "piglet",
    name = "codec::ipv4",
    use_defaults = true,
    test = function()
        local daq_header = DAQHeader.new()
        local raw_buffer = RawBuffer.new("some data")
        local codec_data = CodecData.new()
        local decode_data = DecodeData.new()

        return Codec.decode(
            daq_header,
            raw_buffer,
            codec_data,
            decode_data
        )
    end
}
```

To run snort in piglet mode, first build snort with the ENABLE_PIGLET option turned on (pass the flag -DENABLE_PIGLET:BOOL=ON in cmake).

Then, run the following command:

```
snort --script-path $test_scripts --piglet
```

(where $test_scripts is the directory containing your piglet tests).

The test runner will generate a check-like output, indicating the the results of each test script.

17.8 Piglet Lua API

This section documents the API that piglet exposes to Lua. Refer to the piglet directory in the source tree for examples of usage.

Note: Because of the differences between the Lua and C++ data model and type system, not all parameters map directly to the parameters of the underlying C\++ member functions. Every effort has been made to keep the mappings consist, but there are still some differences. They are documented below.

17.8.1 Plugin Instances

For each test, piglet instantiates plugin specified in the `name` field of the `plugin` table. The virtual methods of the instance are exposed in a table unique to each plugin type. The name of the table is the CamelCase name of the plugin type.

For example, codec plugins have a virtual method called `decode`. This method is called like this:

```
Codec.decode(...)
```

Codec

- `Codec.get_data_link_type() → { int, int, ... }`
- `Codec.get_protocol_ids() → { int, int, ... }`
- `Codec.decode(DAQHeader, RawBuffer, CodecData, DecodeData) → bool`
- `Codec.log(RawBuffer, uint[lyr_len])`
- `Codec.encode(RawBuffer, EncState, Buffer) → bool`
- `Codec.update(uint[flags_hi], uint[flags_lo], RawBuffer, uint[lyr_len] → int`
- `Codec.format(bool[reverse], RawBuffer, DecodeData)`

Differences:

- In `Codec.update()`, the `(uint64_t) flags` parameter has been split into `flags_hi` and `flags_lo`

Inspector

- `Inspector.configure()`
- `Inspector.tinit()`
- `Inspector.tterm()`
- `Inspector.likes(Packet)`
- `Inspector.eval(Packet)`
- `Inspector.clear(Packet)`
- `Inspector.get_buf_from_key(string[key], Packet, RawBuffer) → bool`
- `Inspector.get_buf_from_id(uint[id], Packet, RawBuffer) → bool`
- `Inspector.get_buf_from_type(uint[type], Packet, RawBuffer) → bool`
- `Inspector.get_splitter(bool[to_server]) → StreamSplitter`

Differences: * In `Inspector.configure()`, the `SnortConfig*` parameter is passed implicitly. * the overloaded `get_buf()` member function has been split into three separate methods.

IpsOption

- `IpsOption.hash() → int`
- `IpsOption.is_relative() → bool`
- `IpsOption.fp_research() → bool`
- `IpsOption.get_cursor_type() → int`
- `IpsOption.eval(Cursor, Packet) → int`
- `IpsOption.action(Packet)`

IpsAction

- `IpsAction.exec(Packet)`

Logger

- `Logger.open()`
- `Logger.close()`
- `Logger.reset()`
- `Logger.alert(Packet, string[message], Event)`
- `Logger.log(Packet, string[message], Event)`

SearchEngine

Currently, SearchEngine does not expose any methods.

SoRule

Currently, SoRule does not expose any methods.

Interface Objects

Many of the plugins take C++ classes and structs as arguments. These objects are exposed to the Lua API as Lua userdata. Exposed objects are instantiated by calling the `new` method from each object's method table.

For example, the DecodeData object can be instantiated and exposed to Lua like this:

```
local decode_data = DecodeData.new(...)
```

Each object also exposes useful methods for getting and setting member variables, and calling the C++ methods contained in the the object. These methods can be accessed using the `:` accessor syntax:

```
decode_data:set({ sp = 80, dp = 3500 })
```

Since this is just syntactic sugar for passing the object as the first parameter of the function `DecodeData.set`, an equivalent form is:

```
decode_data.set(decode_data, { sp = 80, dp = 3500 })
```

or even:

```
DecodeData.set(decode_data, { sp = 80, dp = 3500 })
```

Buffer

- `Buffer.new(string[data]) → Buffer`
- `Buffer.new(uint[length]) → Buffer`
- `Buffer.new(RawBuffer) → Buffer`
- `Buffer:allocate(uint[length]) → bool`
- `Buffer:clear()`

CodecData

- `CodecData.new() → CodecData`
- `CodecData.new(uint[next_prot_id]) → CodecData`
- `CodecData.new(fields) → CodecData`
- `CodecData:get() → fields`
- `CodecData:set(fields)`

`fields` is a table with the following contents:

- `next_prot_id`
- `lyr_len`
- `invalid_bytes`
- `proto_bits`
- `codec_flags`
- `ip_layer_cnt`
- `ip6_extension_count`

- `curr_ip6_extension`
- `ip6_csum_proto`

Cursor

- `Cursor.new()` → `Cursor`
- `Cursor.new(Packet)` → `Cursor`
- `Cursor.new(string[data])` → `Cursor`
- `Cursor.new(RawBuffer)` → `Cursor`
- `Cursor:reset()`
- `Cursor:reset(Packet)`
- `Cursor:reset(string[data])`
- `Cursor:reset(RawBuffer)`

DAQHeader

- `DAQHeader.new()` → `DAQHeader`
- `DAQHeader.new(fields)` → `DAQHeader`
- `DAQHeader:get()` → `fields`
- `DAQHeader:set(fields)`

`fields` is a table with the following contents:

- `caplen`
- `pktlen`
- `ingress_index`
- `egress_index`
- `ingress_group`
- `egress_group`
- `flags`
- `opaque`

DecodeData

- `DecodeData.new()` → `DecodeData`
- `DecodeData.new(fields)` → `DecodeData`
- `DecodeData:reset()`
- `DecodeData:get()` → `fields`
- `DecodeData:set(fields)`
- `DecodeData:set_ipv4_hdr(RawBuffer, uint[offset])`

`fields` is a table with the following contents:

- `sp`
- `dp`
- `decode_flags`
- `type`

EncState

- `EncState.new() → EncState`
- `EncState.new(uint[flags_lo]) → EncState`
- `EncState.new(uint[flags_lo], uint[flags_hi]) → EncState`
- `EncState.new(uint[flags_lo], uint[flags_hi], uint[next_proto]) → EncState`
- `EncState.new(uint[flags_lo], uint[flags_hi], uint[next_proto], uint[ttl]) → EncState`
- `EncState.new(uint[flags_lo], uint[flags_hi], uint[next_proto], uint[ttl], uint[dsize ]) → EncState`

Event

- `Event.new() → Event`
- `Event.new(fields) → Event`
- `Event:get() → fields`
- `Event:set(fields)`

`fields` is a table with the following contents:

- `event_id`
- `event_reference`
- `sig_info`
  - `generator`
  - `id`
  - `rev`
  - `class_id`
  - `priority`
  - `text_rule`
  - `num_services`

Flow

- `Flow.new() → Flow`
- `Flow:reset()`

Packet

- `Packet.new() → Packet`
- `Packet.new(string[data]) → Packet`
- `Packet.new(uint[size]) → Packet`
- `Packet.new(fields) → Packet`
- `Packet.new(RawBuffer) → Packet`
- `Packet.new(DAQHeader) → Packet`
- `Packet:set_decode_data(DecodeData)`
- `Packet:set_data(uint[offset], uint[length])`
- `Packet:set_flow(Flow)`
- `Packet:get() → fields`
- `Packet:set()`
- `Packet:set(string[data])`
- `Packet:set(uint[size])`
- `Packet:set(fields)`
- `Packet:set(RawBuffer)`
- `Packet:set(DAQHeader)`

`fields` is a table with the following contents:

- `packet_flags`
- `xtradata_mask`
- `proto_bits`
- `application_protocol_ordinal`
- `alt_dsize`
- `num_layers`
- `iplist_id`
- `user_policy_id`
- `ps_proto`

Note: `Packet.new()` and `Packet:set()` accept multiple arguments of the types described above in any order

RawBuffer

- `RawBuffer.new() → RawBuffer`
- `RawBuffer.new(uint[size]) → RawBuffer`
- `RawBuffer.new(string[data]) → RawBuffer`
- `RawBuffer:size() → int`
- `RawBuffer:resize(uint[size])`
- `RawBuffer:write(string[data])`

- `RawBuffer:write(string[data], uint[size])`
- `RawBuffer:read() → string`
- `RawBuffer:read(uint[end]) → string`
- `RawBuffer:read(uint[start], uint[end]) → string`

Note: calling `RawBuffer.new()` with no arguments returns a RawBuffer of size 0

StreamSplitter

- `StreamSplitter:scan(Flow, RawBuffer) → int, int`
- `StreamSplitter:scan(Flow, RawBuffer, uint[len]) → int, int`
- `StreamSplitter:scan(Flow, RawBuffer, uint[len], uint[flags]) → int, int`
- `StreamSplitter:reassemble(Flow, uint[total], uint[offset], RawBuffer) → int, RawBuffer`
- `StreamSplitter:reassemble(Flow, uint[total], uint[offset], RawBuffer, uint[len]) → int, RawBuffer`
- `StreamSplitter:reassemble(Flow, uint[total], uint[offset], RawBuffer, uint[len], uint[flags]) → int, RawBuffer`
- `StreamSplitter:finish(Flow) → bool`

Note: StreamSplitter does not have a `new()` method, it must be created by an inspector via `Inspector.get_splitter()`

18 Coding Style

All new code should try to follow these style guidelines. These are not yet firm so feedback is welcome to get something we can live with.

18.1 General

- Generally try to follow http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml, but there are some differences documented here.
- Each source directory should have a dev_notes.txt file summarizing the key points and design decisions for the code in that directory. These are built into the developers guide.
- Makefile.am and CMakeLists.txt should have the same files listed in alpha order. This makes it easier to maintain both build systems.
- All new code must come with unit tests providing 95% coverage or better.
- Generally, Catch is preferred for tests in the source file and CppUTest is preferred for test executables in a test subdirectory.

18.2 C++ Specific

- Do not use exceptions. Exception-safe code is non-trivial and we have ported legacy code that makes use of exceptions unwise. There are a few exceptions to this rule for the memory manager, shell, etc. Other code should handle errors as errors.
- Do not use dynamic_cast or RTTI. Although compilers are getting better all the time, there is a time and space cost to this that is easily avoided.

- Use smart pointers judiciously as they aren't free. If you would have to roll your own, then use a smart pointer. If you just need a dtor to delete something, write the dtor.
- Prefer *and* over && and *or* over || for new source files.
- Use nullptr instead of NULL.
- Use new, delete, and their [] counterparts instead of malloc and free except where realloc must be used. But try not to use realloc. New and delete can't return nullptr so no need to check. And Snort's memory manager will ensure that we live within our memory budget.
- Use references in lieu of pointers wherever possible.
- Use the order public, protected, private top to bottom in a class declaration.
- Keep inline functions in a class declaration very brief, preferably just one line. If you need a more complex inline function, move the definition below the class declaration.
- The goal is to have highly readable class declarations. The user shouldn't have to sift through implementation details to see what is available to the client.
- Any using statements in source files should be added only after all includes have been declared.

18.3 Naming

- Use camel case for namespaces, classes, and types like WhizBangPdfChecker.
- Use lower case identifiers with underscore separators, e.g. some_function() and my_var.
- Do not start or end variable names with an underscore. This has a good chance of conflicting with macro and/or system definitions.
- Use lower case filenames with underscores.

18.4 Comments

- Write comments sparingly with a mind towards future proofing. Often the comments can be obviated with better code. Clear code is better than a comment.
- Heed Tim Ottinger's Rule on Comments (https://disqus.com/by/tim_ottinger/):
  1. Comments should only say what the code is incapable of saying.
  2. Comments that repeat (or pre-state) what the code is doing must be removed.
  3. If the code CAN say what the comment is saying, it must be changed at least until rule #2 is in force.
- Function comment blocks are generally just noise that quickly becomes obsolete. If you absolutely must comment on parameters, put each on a separate line along with the comment. That way changing the signature may prompt a change to the comments too.
- Use FIXIT (not FIXTHIS or TODO or whatever) to mark things left for a day or even just a minute. That way we can find them easily and won't lose track of them.
- Presently using FIXIT-X where X = A | W | P | H | M | L, indicating analysis, warning, perf, high, med, or low priority. Place A and W comments on the exact warning line so we can match up comments and build output. Supporting comments can be added above.
- Put the copyright(s) and license in a comment block at the top of each source file (.h and .cc). Don't bother with trivial scripts and make foo. Some interesting Lua code should get a comment block too. Copy and paste exactly from src/main.h (don't reformat).

- Put author, description, etc. in separate comment(s) following the license. Do not put such comments in the middle of the license foo. Be sure to put the author line ahead of the header guard to exclude them from the developers guide. Use the following format, and include a mention to the original author if this is derived work:

  ```
  // ips_dnp3_obj.cc author Maya Dagon <mdagon@cisco.com>
  // based on work by Ryan Jordan
  ```

- Each header should have a comment immediately after the header guard to give an overview of the file so the reader knows what's going on.

18.5 Logging

- Messages intended for the user should not look like debug messages. Eg, the function name should not be included. It is generally unhelpful to include pointers.
- Most debug messages should just be deleted.
- Don't bang your error messages (no !). The user feels bad enough about the problem already w/o you shouting at him.

18.6 Types

- Use logical types to make the code clearer and to help the compiler catch problems. typedef uint16_t Port; bool foo(Port) is way better than int foo(int port).
- Use forward declarations (e.g. struct SnortConfig;) instead of void*.
- Try not to use extern data unless absolutely necessary and then put the extern in an appropriate header. Exceptions for things used in exactly one place like BaseApi pointers.
- Use const liberally. In most cases, const char* s = "foo" should be const char* const s = "foo". The former goes in the initialized data section and the latter in read only data section.
- But use const char s[] = "foo" instead of const char* s = "foo" when possible. The latter form allocates a pointer variable and the data while the former allocates only the data.
- Use static wherever possible to minimize public symbols and eliminate unneeded relocations.
- Declare functions virtual only in the parent class introducing the function (not in a derived class that is overriding the function). This makes it clear which class introduces the function.
- Declare functions as override if they are intended to override a function. This makes it possible to find derived implementations that didn't get updated and therefore won't get called due a change in the parent signature.
- Use bool functions instead of int unless there is truly a need for multiple error returns. The C-style use of zero for success and -1 for error is less readable and often leads to messy code that either ignores the various errors anyway or needlessly and ineffectively tries to do something aobut them. Generally that code is not updated if new errors are added.

18.7 Macros (aka defines)

- In many cases, even in C++, use #define name "value" instead of a const char* const name = "value" because it will eliminate a symbol from the binary.
- Use inline functions instead of macros where possible (pretty much all cases except where stringification is necessary). Functions offer better typing, avoid re-expansions, and a debugger can break there.
- All macros except simple const values should be wrapped in () and all args should be wrapped in () too to avoid surprises upon expansion. Example:

  ```
  #define SEQ_LT(a,b)  ((int)((a) - (b)) <  0)
  ```

- Multiline macros should be blocked (i.e. inside { }) to avoid if-else type surprises.

18.8 Formatting

- Try to keep all source files under 2500 lines. 3000 is the max allowed. If you need more lines, chances are that the code needs to be refactored.
- Indent 4 space chars … no tabs!
- If you need to indent many times, something could be rewritten or restructured to make it clearer. Fewer indents is generally easier to write, easier to read, and overall better code.
- Braces go on the line immediately following a new scope (function signature, if, else, loop, switch, etc.
- Use consistent spacing and line breaks. Always indent 4 spaces from the breaking line. Keep lines less than 100 chars; it greatly helps readability.

```
No:
    calling_a_func_with_a_long_name(arg1,
                                    arg2,
                                    arg3);

Yes:
    calling_a_func_with_a_long_name(
        arg1, arg2, arg3);
```

- Put function signature on one line, except when breaking for the arg list:

```
No:
    inline
    bool foo()
    { // ...

Yes:
    inline bool foo()
    { // ...
```

- Put conditional code on the line following the if so it is easy to break on the conditional block:

```
No:
    if ( test ) foo();

Yes:
    if ( test )
        foo();
```

18.9 Headers

- Don’t hesitate to create a new header if it is needed. Don’t lump unrelated stuff into an header because it is convenient.
- Write header guards like this (leading underscores are reserved for system stuff). In my_header.h:

```
#ifndef MY_HEADER_H
#define MY_HEADER_H
// ...
#endif
```

- Includes from a different directory should specify parent directory. This makes it clear exactly what is included and avoids the primordial soup that results from using -I this -I that -I the_other_thing … .

```
// given:
src/foo/foo.cc
src/bar/bar.cc
src/bar/baz.cc

// in baz.cc
#include "bar.h"

// in foo.cc
#include "bar/bar.h"
```

- Includes within installed headers should specify parent directory.
- Just because it is a #define doesn't mean it goes in a header. Everything should be scoped as tightly as possible. Shared implementation declarations should go in a separate header from the interface. And so on.
- A .cc should include its own .h before any others (including system headers). This ensures that the header stands on its own and can be used by clients without include prerequisites and the developer will be the first to find a dependency problem.
- Include required headers, all required headers, and nothing but required headers. Don't just clone a bunch of headers because it is convenient.
- Try to keep includes in alpha order. This makes it easier to maintain, avoid duplicates, etc.
- Any file depending on #ifdefs should include config.h as shown below. A .h should include it before any other includes, and a .cc should include it immediately after the include of its own .h.

  ```
  #ifdef HAVE_CONFIG_H
  #include "config.h"
  #endif
  ```

- Do not put using statements in headers unless they are tightly scoped.

18.10 Warnings

- With g++, use at least these compiler flags:

  ```
  -Wall -Wextra -pedantic -Wformat -Wformat-security
  -Wunused-but-set-variable -Wno-deprecated-declarations
  -fsanitize=address -fno-omit-frame-pointer
  ```

- With clang, use at least these compiler flags:

  ```
  -Wall -Wextra -pedantic -Wformat -Wformat-security
  -Wno-deprecated-declarations
  -fsanitize=address -fno-omit-frame-pointer
  ```

- Two macros (PADDING_GUARD_BEGIN and PADDING_GUARD_END) are provided by utils/cpp_macros.h. These should be used to surround any structure used as a hash key with a raw comparator or that would otherwise suffer from unintentional padding. A compiler warning will be generated if any structure definition is automatically padded between the macro invocations.
- Then Fix All Warnings and Aborts. None Allowed.

18.11 Uncrustify

Currently using uncrustify from at https://github.com/bengardner/uncrustify to reformat legacy code and anything that happens to need a makeover at some point.

The working config is crusty.cfg in the top level directory. It does well but will munge some things. Specially formatted INDENT-OFF comments were added in 2 places to avoid a real mess.

You can use uncrustify something like this:

```
uncrustify -c crusty.cfg --replace file.cc
```

19 Reference

19.1 Build Options

The options listed below must be explicitly enabled so they are built into the Snort binary. For a full list of build options, run ./configure --help.

- **--enable-shell**: enable building local and remote command line shell support.

These options are built only if the required libraries and headers are present. There is no need to explicitly enable.

- **lzma**: for decompression of SWF and PDF files.
- **openssl**: for SHA and MD5 file signatures and the protected_content rule option.
- *intel-soft-cpm": an optional pattern matcher based on a library from Intel.
- hyperscan for the regex rule option and hyperscan search engine.

If you need to use headers and/or libraries in non-standard locations, you can use these options:

- **--with-pkg-includes**: specify the directory containing the package headers.
- **--with-pkg-libraries**: specify the directory containing the package libraries.

These can be use for pcap, luajit, pcre, dnet, daq, lzma, openssl, intel-soft-cpm, and hyperscan packages. For more information on these libraries see the Getting Started section of the manual.

19.2 Environment Variables

- **HOSTTYPE**: optional string that is output with the version at end of line.
- **LUA_PATH**: you must export as follows so LuaJIT can find required files.

  ```
  LUA_PATH=$install_dir/include/snort/lua/\?.lua\;\;
  ```

- **SNORT_IGNORE**: the list of symbols Snort should ignore when parsing the Lua conf. Unknown symbols not in SNORT_IGNORE will cause warnings with --warn-unknown or fatals with --warn-unknown --pedantic.
- **SNORT_LUA_PATH**: an optional path where Snort can find supplemental conf files such as classification.lua.
- **SNORT_PROMPT**: the character sequence that is printed at startup, shutdown, and in the shell. The default is the mini-pig: o")~ .
- **SNORT_PLUGIN_PATH**: an optional path where Snort can find supplemental shared libraries. This is only used when Snort is building manuals. Modules in supplemental shared libraries will be added to the manuals.

19.3 Command Line Options

- **--alert-before-pass** process alert, drop, sdrop, or reject before pass; default is pass before alert, drop,…
- **-A** <mode> set alert mode: none, cmg, or alert_*
- **-B** <mask> obfuscated IP addresses in alerts and packet dumps using CIDR mask
- **--bpf** <filter options> are standard BPF options, as seen in TCPDump
- **--c2x** output hex for given char (see also --x2c)
- **--catch-test** comma separated list of cat unit test tags or *all*
- **-c** <conf> use this configuration
- **-C** print out payloads with character data only (no hex)
- **--create-pidfile** create PID file, even when not in Daemon mode
- **--daq-dir** <dir> tell snort where to find desired DAQ
- **--daq-list** list packet acquisition modules available in optional dir, default is static modules only
- **--daq** <type> select packet acquisition module (default is pcap)
- **--daq-var** <name=value> specify extra DAQ configuration variable
- **-d** dump the Application Layer
- **--dirty-pig** don’t flush packets on shutdown
- **-D** run Snort in background (daemon) mode
- **--dump-builtin-rules** [<module prefix>] output stub rules for selected modules
- **--dump-defaults** [<module prefix>] output module defaults in Lua format (optional)
- **--dump-dynamic-rules** output stub rules for all loaded rules libraries
- **--dump-version** output the version, the whole version, and only the version
- **-e** display the second layer header info
- **--enable-inline-test** enable Inline-Test Mode Operation
- **-f** turn off fflush() calls after binary log writes
- **-G** <0xid> (same as --logid) (0:65535)
- **-g** <gname> run snort gid as <gname> group (or gid) after initialization
- **--help-commands** [<module prefix>] output matching commands (optional)
- **--help-config** [<module prefix>] output matching config options (optional)
- **--help-counts** [<module prefix>] output matching peg counts (optional)
- **--help** list command line options
- **--help-module** <module> output description of given module
- **--help-modules** list all available modules with brief help
- **--help-options** <option prefix> output matching command line option quick help (same as -?) (optional)
- **--help-plugins** list all available plugins with brief help
- **--help-signals** dump available control signals

- **-H** make hash tables deterministic
- **--id-subdir** create/use instance subdirectories in logdir instead of instance filename prefix
- **--id-zero** use id prefix / subdirectory even with one packet thread
- **-i** <iface>… list of interfaces
- **-j** <port> to listen for telnet connections
- **-k** <mode> checksum mode; default is all (all|noip|notcp|noudp|noicmp|none)
- **--list-buffers** output available inspection buffers
- **--list-builtin** <module prefix> output matching builtin rules (optional)
- **--list-gids** [<module prefix>] output matching generators (optional)
- **--list-modules** [<module type>] list all known modules of given type (optional)
- **--list-plugins** list all known plugins
- **-l** <logdir> log to this directory instead of current directory
- **-L** <mode> logging mode (none, dump, pcap, or log_*)
- **--logid** <0xid> log Identifier to uniquely id events for multiple snorts (same as -G) (0:65535)
- **--lua** <chunk> extend/override conf with chunk; may be repeated
- **--markup** output help in asciidoc compatible format
- **--max-packet-threads** <count> configure maximum number of packet threads (same as -z) (0:)
- **-M** log messages to syslog (not alerts)
- **-m** <umask> set umask = <umask> (0:)
- **-n** <count> stop after count packets (0:)
- **--nolock-pidfile** do not try to lock Snort PID file
- **--nostamps** don’t include timestamps in log file names
- **-O** obfuscate the logged IP addresses
- **-?** <option prefix> output matching command line option quick help (same as --help-options) (optional)
- **--pause** wait for resume/quit command before processing packets/terminating
- **--pcap-dir** <dir> a directory to recurse to look for pcaps - read mode is implied
- **--pcap-file** <file> file that contains a list of pcaps to read - read mode is implied
- **--pcap-filter** <filter> filter to apply when getting pcaps from file or directory
- **--pcap-list** <list> a space separated list of pcaps to read - read mode is implied
- **--pcap-loop** <count> read all pcaps <count> times; 0 will read until Snort is terminated (-1:)
- **--pcap-no-filter** reset to use no filter when getting pcaps from file or directory
- **--pcap-reload** if reading multiple pcaps, reload snort config between pcaps
- **--pcap-show** print a line saying what pcap is currently being read
- **--pedantic** warnings are fatal
- **--piglet** enable piglet test harness mode

- **--plugin-path** <path> where to find plugins
- **--process-all-events** process all action groups
- **-Q** enable inline mode operation
- **-q** quiet mode - Don't show banner and status report
- **-r** <pcap>… (same as --pcap-list)
- **-R** <rules> include this rules file in the default policy
- **--rule** <rules> to be added to configuration; may be repeated
- **--rule-to-hex** output so rule header to stdout for text rule on stdin
- **--rule-to-text** output plain so rule header to stdout for text rule on stdin
- **--run-prefix** <pfx> prepend this to each output file
- **--script-path** <path> to a luajit script or directory containing luajit scripts
- **--shell** enable the interactive command line
- **--show-plugins** list module and plugin versions
- **--skip** <n> skip 1st n packets (0:)
- **--snaplen** <snap> set snaplen of packet (same as -s) (68:65535)
- **-s** <snap> (same as --snaplen); default is 1514 (68:65535)
- **--stdin-rules** read rules from stdin until EOF or a line starting with END is read
- **-S** <x=v> set config variable x equal to value v
- **-t** <dir> chroots process to <dir> after initialization
- **--treat-drop-as-alert** converts drop, sdrop, and reject rules into alert rules during startup
- **--treat-drop-as-ignore** use drop, sdrop, and reject rules to ignore session traffic when not inline
- **-T** test and report on the current Snort configuration
- **-u** <uname> run snort as <uname> or <uid> after initialization
- **-U** use UTC for timestamps
- **-v** be verbose
- **--version** show version number (same as -V)
- **-V** (same as --version)
- **--warn-all** enable all warnings
- **--warn-conf** warn about configuration issues
- **--warn-daq** warn about DAQ issues, usually related to mode
- **--warn-flowbits** warn about flowbits that are checked but not set and vice-versa
- **--warn-hosts** warn about host table issues
- **--warn-plugins** warn about issues that prevent plugins from loading
- **--warn-rules** warn about duplicate rules and rule parsing issues
- **--warn-scripts** warn about issues discovered while processing Lua scripts

- **--warn-symbols** warn about unknown symbols in your Lua config
- **--warn-vars** warn about variable definition and usage issues
- **-W** lists available interfaces
- **--x2c** output ASCII char for given hex (see also --c2x)
- **--x2s** output ASCII string for given byte code (see also --x2c)
- **-X** dump the raw packet data starting at the link layer
- **-x** same as --pedantic
- **-y** include year in timestamp in the alert and log files
- **-z** <count> maximum number of packet threads (same as --max-packet-threads); 0 gets the number of CPU cores reported by the system; default is 1 (0:)

19.4 Configuration

- string **ack.~range**: check if tcp ack value is *value* | *min<>max* | *<max* | *>min*
- int **active.attempts** = 0: number of TCP packets sent per response (with varying sequence numbers) { 0:20 }
- string **active.device**: use *ip* for network layer responses or *eth0* etc for link layer
- string **active.dst_mac**: use format *01:23:45:67:89:ab*
- int **active.max_responses** = 0: maximum number of responses { 0: }
- int **active.min_interval** = 255: minimum number of seconds between responses { 1: }
- multi **alert_csv.fields** = timestamp pkt_num proto pkt_gen dgm_len dir src_ap dst_ap rule action: selected fields will be output in given order left to right { action | dir | dgm_len | dst_addr | dst_ap | dst_port | eth_dst | eth_len | eth_src | eth_type | gid | icmp_code | icmp_id | icmp_seq | icmp_type | iface | ip_id | ip_len | msg | pkt_gen | pkt_num | proto | rev | rule | sid | src_addr | src_ap | src_port | tcp_ack | tcp_flags | tcp_len | tcp_seq | tcp_win | timestamp | tos | ttl | udp_len }
- bool **alert_csv.file** = false: output to alert_csv.txt instead of stdout
- int **alert_csv.limit** = 0: set limit (0 is unlimited) { 0: }
- string **alert_csv.separator** = , : separate fields with this character sequence
- enum **alert_csv.units** = B: bytes | KB | MB | GB { B | K | M | G }
- bool **alert_fast.file** = false: output to alert_fast.txt instead of stdout
- int **alert_fast.limit** = 0: set limit (0 is unlimited) { 0: }
- bool **alert_fast.packet** = false: output packet dump with alert
- enum **alert_fast.units** = B: bytes | KB | MB | GB { B | K | M | G }
- bool **alert_full.file** = false: output to alert_full.txt instead of stdout
- int **alert_full.limit** = 0: set limit (0 is unlimited) { 0: }
- enum **alert_full.units** = B: limit is in bytes | KB | MB | GB { B | K | M | G }
- bool **alerts.alert_with_interface_name** = false: include interface in alert info (fast, full, or syslog only)
- bool **alerts.default_rule_state** = true: enable or disable ips rules
- int **alerts.detection_filter_memcap** = 1048576: set available bytes of memory for detection_filters { 0: }

- int **alerts.event_filter_memcap** = 1048576: set available bytes of memory for event_filters { 0: }
- string **alert_sfsocket.file**: name of unix socket file
- int **alert_sfsocket.rules[].gid** = 1: rule generator ID { 1: }
- int **alert_sfsocket.rules[].sid** = 1: rule signature ID { 1: }
- string **alerts.order** = pass drop alert log: change the order of rule action application
- int **alerts.rate_filter_memcap** = 1048576: set available bytes of memory for rate_filters { 0: }
- string **alerts.reference_net**: set the CIDR for homenet (for use with -l or -B, does NOT change $HOME_NET in IDS mode)
- bool **alerts.stateful** = false: don't alert w/o established session (note: rule action still taken)
- string **alerts.tunnel_verdicts**: let DAQ handle non-allow verdicts for GTP|Teredo|6in4|4in6 traffic
- enum **alert_syslog.facility** = auth: part of priority applied to each message { auth | authpriv | daemon | user | local0 | local1 | local2 | local3 | local4 | local5 | local6 | local7 }
- enum **alert_syslog.level** = info: part of priority applied to each message { emerg | alert | crit | err | warning | notice | info | debug }
- multi **alert_syslog.options**: used to open the syslog connection { cons | ndelay | perror | pid }
- string **appid.app_detector_dir**: directory to load appid detectors from
- int **appid.app_stats_period** = 300: time period for collecting and logging appid statistics { 0: }
- int **appid.app_stats_rollover_size** = 20971520: max file size for appid stats before rolling over the log file { 0: }
- int **appid.app_stats_rollover_time** = 86400: max time period for collection appid stats before rolling over the log file { 0: }
- string **appid.conf**: RNA configuration file
- bool **appid.debug** = false: enable appid debug logging
- bool **appid.dump_ports** = false: enable dump of appid port information
- int **appid.instance_id** = 0: instance id - need more details for what this is { 0: }
- bool **appid.log_stats** = false: enable logging of appid statistics
- int **appid.memcap** = 0: disregard - not implemented { 0: }
- string **appids.~**: appid option
- addr **appid.session_log_filter.dst_ip** = 0.0.0.0/32: destination ip address in CIDR format
- port **appid.session_log_filter.dst_port**: destination port { 1: }
- bool **appid.session_log_filter.log_all_sessions** = false: enable logging for all appid sessions
- string **appid.session_log_filter.protocol**: ip protocol
- addr **appid.session_log_filter.src_ip** = 0.0.0.0/32: source ip address in CIDR format
- port **appid.session_log_filter.src_port**: source port { 1: }
- string **appid.thirdparty_appid_dir**: directory to load thirdparty appid detectors from
- ip4 **arp_spoof.hosts[].ip**: host ip address
- mac **arp_spoof.hosts[].mac**: host mac address
- int **asn1.absolute_offset**: absolute offset from the beginning of the packet { 0: }
- implied **asn1.bitstring_overflow**: detects invalid bitstring encodings that are known to be remotely exploitable

- implied **asn1.double_overflow**: detects a double ASCII encoding that is larger than a standard buffer
- int **asn1.oversize_length**: compares ASN.1 type lengths with the supplied argument { 0: }
- implied **asn1.print**: dump decode data to console; always true
- int **asn1.relative_offset**: relative offset from the cursor
- int **attribute_table.max_hosts** = 1024: maximum number of hosts in attribute table { 32:207551 }
- int **attribute_table.max_metadata_services** = 8: maximum number of services in rule metadata { 1:256 }
- int **attribute_table.max_services_per_host** = 8: maximum number of services per host entry in attribute table { 1:65535 }
- int **base64_decode.bytes**: number of base64 encoded bytes to decode { 1: }
- int **base64_decode.offset** = 0: bytes past start of buffer to start decoding { 0: }
- implied **base64_decode.relative**: apply offset to cursor instead of start of buffer
- enum **binder[].use.action** = inspect: what to do with matching traffic { reset | block | allow | inspect }
- string **binder[].use.file**: use configuration in given file
- string **binder[].use.name**: symbol name (defaults to type)
- string **binder[].use.service**: override automatic service identification
- string **binder[].use.type**: select module for binding
- bit_list **binder[].when.ifaces**: list of interface indices { 255 }
- addr_list **binder[].when.nets**: list of networks
- int **binder[].when.policy_id** = 0: unique ID for selection of this config by external logic { 0: }
- bit_list **binder[].when.ports**: list of ports { 65535 }
- enum **binder[].when.proto**: protocol { any | ip | icmp | tcp | udp | user | file }
- enum **binder[].when.role** = any: use the given configuration on one or any end of a session { client | server | any }
- string **binder[].when.service**: override default configuration
- bit_list **binder[].when.vlans**: list of VLAN IDs { 4095 }
- string **bufferlen.~range**: len | min<>max | <max | >min
- int **byte_extract.align** = 0: round the number of converted bytes up to the next 2- or 4-byte boundary { 0:4 }
- implied **byte_extract.big**: big endian
- int **byte_extract.~count**: number of bytes to pick up from the buffer { 1:10 }
- implied **byte_extract.dce**: dcerpc2 determines endianness
- implied **byte_extract.dec**: convert from decimal string
- implied **byte_extract.hex**: convert from hex string
- implied **byte_extract.little**: little endian
- int **byte_extract.multiplier** = 1: scale extracted value by given amount { 1:65535 }
- string **byte_extract.~name**: name of the variable that will be used in other rule options
- implied **byte_extract.oct**: convert from octal string
- int **byte_extract.~offset**: number of bytes into the buffer to start processing { -65535:65535 }

- implied **byte_extract.relative**: offset from cursor instead of start of buffer
- implied **byte_extract.string**: convert from string
- int **byte_jump.align** = 0: round the number of converted bytes up to the next 2- or 4-byte boundary { 0:4 }
- implied **byte_jump.big**: big endian
- int **byte_jump.~count**: number of bytes to pick up from the buffer { 1:10 }
- implied **byte_jump.dce**: dcerpc2 determines endianness
- implied **byte_jump.dec**: convert from decimal string
- implied **byte_jump.from_beginning**: jump from start of buffer instead of cursor
- implied **byte_jump.hex**: convert from hex string
- implied **byte_jump.little**: little endian
- int **byte_jump.multiplier** = 1: scale extracted value by given amount { 1:65535 }
- implied **byte_jump.oct**: convert from octal string
- string **byte_jump.~offset**: variable name or number of bytes into the buffer to start processing
- int **byte_jump.post_offset** = 0: also skip forward or backwards (positive of negative value) this number of bytes { -65535:65535 }
- implied **byte_jump.relative**: offset from cursor instead of start of buffer
- implied **byte_jump.string**: convert from string
- implied **byte_test.big**: big endian
- string **byte_test.~compare**: variable name or value to test the converted result against
- int **byte_test.~count**: number of bytes to pick up from the buffer { 1:10 }
- implied **byte_test.dce**: dcerpc2 determines endianness
- implied **byte_test.dec**: convert from decimal string
- implied **byte_test.hex**: convert from hex string
- implied **byte_test.little**: little endian
- implied **byte_test.oct**: convert from octal string
- string **byte_test.~offset**: variable name or number of bytes into the payload to start processing
- string **byte_test.~operator**: variable name or number of bytes into the buffer to start processing
- implied **byte_test.relative**: offset from cursor instead of start of buffer
- implied **byte_test.string**: convert from string
- string **classifications[].name**: name used with classtype rule option
- int **classifications[].priority** = 1: default priority for class { 0: }
- string **classifications[].text**: description of class
- string **classtype.~**: classification for this rule
- string **content.~data**: data to match
- string **content.depth**: var or maximum number of bytes to search from beginning of buffer

- string **content.distance**: var or number of bytes from cursor to start search
- int **content.fast_pattern_length**: maximum number of characters from this content the fast pattern matcher should use { 1: }
- int **content.fast_pattern_offset** = 0: number of leading characters of this content the fast pattern matcher should exclude { 0: }
- implied **content.fast_pattern**: use this content in the fast pattern matcher instead of the content selected by default
- implied **content.nocase**: case insensitive match
- string **content.offset**: var or number of bytes from start of buffer to start search
- string **content.within**: var or maximum number of bytes to search from cursor
- implied **cvs.invalid-entry**: looks for an invalid Entry string
- string **daq.input_spec**: input specification
- int **daq.instances[].id**: instance ID (required) { 0: }
- string **daq.instances[].input_spec**: input specification
- string **daq.instances[].variables[].str**: string parameter
- string **daq.module**: DAQ module to use
- string **daq.module_dirs[].str**: string parameter
- bool **daq.no_promisc** = false: whether to put DAQ device into promiscuous mode
- int **daq.snaplen**: set snap length (same as -s) { 0:65535 }
- string **daq.variables[].str**: string parameter
- implied **dce_iface.any_frag**: match on any fragment
- string **dce_iface.uuid**: match given dcerpc uuid
- string **dce_iface.version**: interface version
- string **dce_opnum.~**: match given dcerpc operation number, range or list
- bool **dce_smb.disable_defrag** = false: Disable DCE/RPC defragmentation
- int **dce_smb.max_frag_len** = 65535: Maximum fragment size for defragmentation { 1514:65535 }
- enum **dce_smb.policy** = WinXP: Target based policy to use { Win2000 | WinXP | WinVista | Win2003 | Win2008 | Win7 | Samba | Samba-3.0.37 | Samba-3.0.22 | Samba-3.0.20 }
- int **dce_smb.reassemble_threshold** = 0: Minimum bytes received before performing reassembly { 0:65535 }
- int **dce_smb.smb_file_depth** = 16384: SMB file depth for file data { -1: }
- enum **dce_smb.smb_file_inspection** = off: SMB file inspection { off | on | only }
- enum **dce_smb.smb_fingerprint_policy** = none: Target based SMB policy to use { none | client | server | both }
- string **dce_smb.smb_invalid_shares**: SMB shares to alert on
- bool **dce_smb.smb_legacy_mode** = false: inspect only SMBv1
- int **dce_smb.smb_max_chain** = 3: SMB max chain size { 0:255 }
- int **dce_smb.smb_max_compound** = 3: SMB max compound size { 0:255 }
- multi **dce_smb.valid_smb_versions** = all: Valid SMB versions { v1 | v2 | all }
- bool **dce_tcp.disable_defrag** = false: Disable DCE/RPC defragmentation

- int **dce_tcp.max_frag_len** = 65535: Maximum fragment size for defragmentation { 1514:65535 }
- enum **dce_tcp.policy** = WinXP: Target based policy to use { Win2000 | WinXP | WinVista | Win2003 | Win2008 | Win7 | Samba | Samba-3.0.37 | Samba-3.0.22 | Samba-3.0.20 }
- int **dce_tcp.reassemble_threshold** = 0: Minimum bytes received before performing reassembly { 0:65535 }
- bool **dce_udp.disable_defrag** = false: Disable DCE/RPC defragmentation
- int **dce_udp.max_frag_len** = 65535: Maximum fragment size for defragmentation { 1514:65535 }
- int **detection.asn1** = 256: maximum decode nodes { 1: }
- int **detection_filter.count**: hits in interval before allowing the rule to fire { 1: }
- int **detection_filter.seconds**: length of interval to count hits { 1: }
- enum **detection_filter.track**: track hits by source or destination IP address { by_src | by_dst }
- bool **detection.pcre_enable** = true: disable pcre pattern matching
- int **detection.pcre_match_limit** = 1500: limit pcre backtracking, -1 = max, 0 = off { -1:1000000 }
- int **detection.pcre_match_limit_recursion** = 1500: limit pcre stack consumption, -1 = max, 0 = off { -1:10000 }
- bool **dnp3.check_crc** = false: validate checksums in DNP3 link layer frames
- string **dnp3_func.~**: match dnp3 function code or name
- string **dnp3_ind.~**: match given dnp3 indicator flags
- int **dnp3_obj.group** = 0: match given dnp3 object header group { 0:255 }
- int **dnp3_obj.var** = 0: match given dnp3 object header var { 0:255 }
- string **dsize.~range**: check if packet payload size is *size* | *min<>max* | *<max* | *>min*
- bool **esp.decode_esp** = false: enable for inspection of esp traffic that has authentication but not encryption
- int **event_filter[].count** = 0: number of events in interval before tripping; -1 to disable { -1: }
- int **event_filter[].gid** = 1: rule generator ID { 0: }
- string **event_filter[].ip**: restrict filter to these addresses according to track
- int **event_filter[].seconds** = 0: count interval { 0: }
- int **event_filter[].sid** = 1: rule signature ID { 0: }
- enum **event_filter[].track**: filter only matching source or destination addresses { by_src | by_dst }
- enum **event_filter[].type**: 1st count events | every count events | once after count events { limit | threshold | both }
- int **event_queue.log** = 3: maximum events to log { 1: }
- int **event_queue.max_queue** = 8: maximum events to queue { 1: }
- enum **event_queue.order_events** = content_length: criteria for ordering incoming events { priority|content_length }
- bool **event_queue.process_all_events** = false: process just first action group or all action groups
- string **file_connector.connector**: connector name
- enum **file_connector.direction**: usage { receive | transmit | duplex }
- enum **file_connector.format**: file format { binary | text }
- string **file_connector.name**: channel name

- int **file_id.block_timeout** = 86400: stop blocking after this many seconds { 0: }
- bool **file_id.block_timeout_lookup** = false: block if lookup times out
- int **file_id.capture_block_size** = 32768: file capture block size in bytes { 8: }
- int **file_id.capture_max_size** = 1048576: stop file capture beyond this point { 0: }
- int **file_id.capture_memcap** = 100: memcap for file capture in megabytes { 0: }
- int **file_id.capture_min_size** = 0: stop file capture if file size less than this { 0: }
- bool **file_id.enable_capture** = false: enable file capture
- bool **file_id.enable_signature** = false: enable signature calculation
- bool **file_id.enable_type** = false: enable type ID
- bool **file_id.file_policy[].use.enable_file_capture** = false: true/false → enable/disable file capture
- bool **file_id.file_policy[].use.enable_file_signature** = false: true/false → enable/disable file signature
- bool **file_id.file_policy[].use.enable_file_type** = false: true/false → enable/disable file type identification
- enum **file_id.file_policy[].use.verdict** = unknown: what to do with matching traffic { unknown | log | stop | block | reset }
- int **file_id.file_policy[].when.file_type_id** = 0: unique ID for file type in file magic rule { 0: }
- string **file_id.file_policy[].when.sha256**: SHA 256
- string **file_id.file_rules[].category**: file type category
- int **file_id.file_rules[].id** = 0: file type id { 0: }
- string **file_id.file_rules[].magic[].content**: file magic content
- int **file_id.file_rules[].magic[].offset** = 0: file magic offset { 0: }
- string **file_id.file_rules[].msg**: information about the file type
- int **file_id.file_rules[].rev** = 0: rule revision { 0: }
- string **file_id.file_rules[].type**: file type name
- string **file_id.file_rules[].version**: file type version
- int **file_id.lookup_timeout** = 2: give up on lookup after this many seconds { 0: }
- int **file_id.max_files_cached** = 65536: maximal number of files cached in memory { 8: }
- int **file_id.show_data_depth** = 100: print this many octets { 0: }
- int **file_id.signature_depth** = 10485760: stop signature at this point { 0: }
- bool **file_id.trace_signature** = false: enable runtime dump of signature info
- bool **file_id.trace_stream** = false: enable runtime dump of file data
- bool **file_id.trace_type** = false: enable runtime dump of type info
- int **file_id.type_depth** = 1460: stop type ID at this point { 0: }
- bool **file_log.log_pkt_time** = true: log the packet time when event generated
- bool **file_log.log_sys_time** = false: log the system time when event generated
- string **file_type.~**: list of file type IDs to match
- string **flags.~mask_flags**: these flags are don't cares

- string **flags.~test_flags**: these flags are tested
- string **flowbits.~arg1**: bits or group
- string **flowbits.~arg2**: group if arg1 is bits
- string **flowbits.~command**: set|reset|isset|etc.
- implied **flow.established**: match only during data transfer phase
- implied **flow.from_client**: same as to_server
- implied **flow.from_server**: same as to_client
- implied **flow.no_frag**: match on raw packets only
- implied **flow.no_stream**: match on raw packets only
- implied **flow.not_established**: match only outside data transfer phase
- implied **flow.only_frag**: match on defragmented packets only
- implied **flow.only_stream**: match on reassembled packets only
- implied **flow.stateless**: match regardless of stream state
- implied **flow.to_client**: match on server responses
- implied **flow.to_server**: match on client requests
- string **fragbits.~flags**: these flags are tested
- string **fragoffset.~range**: check if ip fragment offset value is *value* | *min*<>*max* | <*max* | >*min*
- bool **ftp_client.bounce** = false: check for bounces
- addr **ftp_client.bounce_to[].address** = 1.0.0.0/32: allowed ip address in CIDR format
- port **ftp_client.bounce_to[].last_port**: optional allowed range from port to last_port inclusive { 0: }
- port **ftp_client.bounce_to[].port** = 20: allowed port { 1: }
- bool **ftp_client.ignore_telnet_erase_cmds** = false: ignore erase character and erase line commands when normalizing
- int **ftp_client.max_resp_len** = -1: maximum ftp response accepted by client { -1: }
- bool **ftp_client.telnet_cmds** = false: detect telnet escape sequences on ftp control channel
- bool **ftp_server.check_encrypted** = false: check for end of encryption
- string **ftp_server.chk_str_fmt**: check the formatting of the given commands
- string **ftp_server.cmd_validity[].command**: command string
- string **ftp_server.cmd_validity[].format**: format specification
- int **ftp_server.cmd_validity[].length** = 0: specify non-default maximum for command { 0: }
- string **ftp_server.data_chan_cmds**: check the formatting of the given commands
- string **ftp_server.data_rest_cmds**: check the formatting of the given commands
- string **ftp_server.data_xfer_cmds**: check the formatting of the given commands
- int **ftp_server.def_max_param_len** = 100: default maximum length of commands handled by server; 0 is unlimited { 1: }
- string **ftp_server.directory_cmds[].dir_cmd**: directory command
- int **ftp_server.directory_cmds[].rsp_code** = 200: expected successful response code for command { 200: }

- string **ftp_server.encr_cmds**: check the formatting of the given commands
- bool **ftp_server.encrypted_traffic** = false: check for encrypted telnet and ftp
- string **ftp_server.file_get_cmds**: check the formatting of the given commands
- string **ftp_server.file_put_cmds**: check the formatting of the given commands
- string **ftp_server.ftp_cmds**: specify additional commands supported by server beyond RFC 959
- bool **ftp_server.ignore_data_chan** = false: do not inspect ftp data channels
- bool **ftp_server.ignore_telnet_erase_cmds** = false: ignore erase character and erase line commands when normalizing
- string **ftp_server.login_cmds**: check the formatting of the given commands
- bool **ftp_server.print_cmds** = false: print command configurations on start up
- bool **ftp_server.telnet_cmds** = false: detect telnet escape sequences of ftp control channel
- int **gid.~**: generator id { 1: }
- string **gtp_info.~**: info element to match
- int **gtp_inspect[].infos[].length** = 0: information element type code { 0:255 }
- string **gtp_inspect[].infos[].name**: information element name
- int **gtp_inspect[].infos[].type** = 0: information element type code { 0:255 }
- string **gtp_inspect[].messages[].name**: message name
- int **gtp_inspect[].messages[].type** = 0: message type code { 0:255 }
- int **gtp_inspect[].version** = 2: gtp version { 0:2 }
- string **gtp_type.~**: list of types to match
- int **gtp_version.~**: version to match { 0:2 }
- bool **high_availability.daq_channel** = false: enable use of daq data plane channel
- bool **high_availability.enable** = false: enable high availability
- real **high_availability.min_age** = 1.0: minimum session life before HA updates { 0.0:100.0 }
- real **high_availability.min_sync** = 1.0: minimum interval between HA updates { 0.0:100.0 }
- bit_list **high_availability.ports**: side channel message port list { 65535 }
- int **host_cache[].size**: size of host cache
- enum **hosts[].frag_policy**: defragmentation policy { first | linux | bsd | bsd_right | last | windows | solaris }
- addr **hosts[].ip** = 0.0.0.0/32: hosts address / cidr
- string **hosts[].services[].name**: service identifier
- port **hosts[].services[].port**: port number
- enum **hosts[].services[].proto** = tcp: ip protocol { tcp | udp }
- enum **hosts[].tcp_policy**: tcp reassembly policy { first | last | linux | old_linux | bsd | macos | solaris | irix | hpux11 | hpux10 | windows | win_2003 | vista | proxy }
- enum **host_tracker[].frag_policy**: defragmentation policy { first | linux | bsd | bsd_right | last | windows | solaris }
- addr **host_tracker[].ip** = 0.0.0.0/32: hosts address / cidr

- string **host_tracker[].services[].name**: service identifier
- port **host_tracker[].services[].port**: port number
- enum **host_tracker[].services[].proto** = tcp: ip protocol { tcp | udp }
- enum **host_tracker[].tcp_policy**: tcp reassembly policy { first | last | linux | old_linux | bsd | macos | solaris | irix | hpux11 | hpux10 | windows | win_2003 | vista | proxy }
- implied **http_cookie.request**: match against the cookie from the request message even when examining the response
- implied **http_cookie.with_body**: parts of this rule examine HTTP message body
- implied **http_cookie.with_trailer**: parts of this rule examine HTTP message trailers
- string **http_header.field**: restrict to given header. Header name is case insensitive.
- implied **http_header.request**: match against the headers from the request message even when examining the response
- implied **http_header.with_body**: parts of this rule examine HTTP message body
- implied **http_header.with_trailer**: parts of this rule examine HTTP message trailers
- bool **http_inspect.backslash_to_slash** = false: replace \ with / when normalizing URIs
- bit_list **http_inspect.bad_characters**: alert when any of specified bytes are present in URI after percent decoding { 255 }
- string **http_inspect.ignore_unreserved**: do not alert when the specified unreserved characters are percent-encoded in a URI.Unreserved characters are 0-9, a-z, A-Z, period, underscore, tilde, and minus. { (optional) }
- bool **http_inspect.iis_double_decode** = false: perform double decoding of percent encodings to normalize characters
- int **http_inspect.iis_unicode_code_page** = 1252: code page to use from the IIS unicode map file { 0:65535 }
- bool **http_inspect.iis_unicode** = false: use IIS unicode code point mapping to normalize characters
- string **http_inspect.iis_unicode_map_file**: file containing code points for IIS unicode. { (optional) }
- int **http_inspect.max_javascript_whitespaces** = 200: maximum consecutive whitespaces allowed within the Javascript obfuscated data { 1:65535 }
- bool **http_inspect.normalize_javascript** = false: normalize javascript in response bodies
- bool **http_inspect.normalize_utf** = true: normalize charset utf encodings in response bodies
- int **http_inspect.oversize_dir_length** = 300: maximum length for URL directory { 1:65535 }
- bool **http_inspect.percent_u** = false: normalize %uNNNN and %UNNNN encodings
- bool **http_inspect.plus_to_space** = true: replace + with <sp> when normalizing URIs
- int **http_inspect.print_amount** = 1200: number of characters to print from a Field { 1:1000000 }
- bool **http_inspect.print_hex** = false: nonprinting characters printed in [HH] format instead of using an asterisk
- int **http_inspect.request_depth** = -1: maximum request message body bytes to examine (-1 no limit) { -1: }
- int **http_inspect.response_depth** = -1: maximum response message body bytes to examine (-1 no limit) { -1: }
- bool **http_inspect.show_pegs** = true: display peg counts with test output
- bool **http_inspect.simplify_path** = true: reduce URI directory path to simplest form
- bool **http_inspect.test_input** = false: read HTTP messages from text file
- bool **http_inspect.test_output** = false: print out HTTP section data
- bool **http_inspect.unzip** = true: decompress gzip and deflate message bodies

- bool **http_inspect.utf8_bare_byte** = false: when doing UTF-8 character normalization include bytes that were not percent encoded
- bool **http_inspect.utf8** = true: normalize 2-byte and 3-byte UTF-8 characters to a single byte
- implied **http_method.with_body**: parts of this rule examine HTTP message body
- implied **http_method.with_trailer**: parts of this rule examine HTTP message trailers
- implied **http_raw_cookie.request**: match against the cookie from the request message even when examining the response
- implied **http_raw_cookie.with_body**: parts of this rule examine HTTP message body
- implied **http_raw_cookie.with_trailer**: parts of this rule examine HTTP message trailers
- implied **http_raw_header.request**: match against the headers from the request message even when examining the response
- implied **http_raw_header.with_body**: parts of this rule examine HTTP message body
- implied **http_raw_header.with_trailer**: parts of this rule examine HTTP message trailers
- implied **http_raw_request.with_body**: parts of this rule examine HTTP message body
- implied **http_raw_request.with_trailer**: parts of this rule examine HTTP message trailers
- implied **http_raw_status.with_body**: parts of this rule examine HTTP message body
- implied **http_raw_status.with_trailer**: parts of this rule examine HTTP message trailers
- implied **http_raw_trailer.request**: match against the trailers from the request message even when examining the response
- implied **http_raw_trailer.with_body**: parts of this rule examine HTTP response message body (must be combined with request)
- implied **http_raw_trailer.with_header**: parts of this rule examine HTTP response message headers (must be combined with request)
- implied **http_raw_uri.fragment**: match against fragment section of URI only
- implied **http_raw_uri.host**: match against host section of URI only
- implied **http_raw_uri.path**: match against path section of URI only
- implied **http_raw_uri.port**: match against port section of URI only
- implied **http_raw_uri.query**: match against query section of URI only
- implied **http_raw_uri.scheme**: match against scheme section of URI only
- implied **http_raw_uri.with_body**: parts of this rule examine HTTP message body
- implied **http_raw_uri.with_trailer**: parts of this rule examine HTTP message trailers
- implied **http_stat_code.with_body**: parts of this rule examine HTTP message body
- implied **http_stat_code.with_trailer**: parts of this rule examine HTTP message trailers
- implied **http_stat_msg.with_body**: parts of this rule examine HTTP message body
- implied **http_stat_msg.with_trailer**: parts of this rule examine HTTP message trailers
- string **http_trailer.field**: restrict to given trailer
- implied **http_trailer.request**: match against the trailers from the request message even when examining the response
- implied **http_trailer.with_body**: parts of this rule examine HTTP message body (must be combined with request)

- implied **http_trailer.with_header**: parts of this rule examine HTTP response message headers (must be combined with request)
- implied **http_uri.fragment**: match against fragment section of URI only
- implied **http_uri.host**: match against host section of URI only
- implied **http_uri.path**: match against path section of URI only
- implied **http_uri.port**: match against port section of URI only
- implied **http_uri.query**: match against query section of URI only
- implied **http_uri.scheme**: match against scheme section of URI only
- implied **http_uri.with_body**: parts of this rule examine HTTP message body
- implied **http_uri.with_trailer**: parts of this rule examine HTTP message trailers
- implied **http_version.request**: match against the version from the request message even when examining the response
- implied **http_version.with_body**: parts of this rule examine HTTP message body
- implied **http_version.with_trailer**: parts of this rule examine HTTP message trailers
- string **icmp_id.~range**: check if icmp id is *id* | *min*<>*max* | <*max* | >*min*
- string **icmp_seq.~range**: check if icmp sequence number is *seq* | *min*<>*max* | <*max* | >*min*
- string **icode.~range**: check if ICMP code is *code* | *min*<>*max* | <*max* | >*min*
- string **id.~range**: check if the IP ID is *id* | *min*<>*max* | <*max* | >*min*
- int **imap.b64_decode_depth** = 1460: base64 decoding depth { -1:65535 }
- int **imap.bitenc_decode_depth** = 1460: non-Encoded MIME attachment extraction depth { -1:65535 }
- int **imap.qp_decode_depth** = 1460: quoted Printable decoding depth { -1:65535 }
- int **imap.uu_decode_depth** = 1460: Unix-to-Unix decoding depth { -1:65535 }
- select **ipopts.~opt**: output format { rr|eol|nop|ts|sec|esec|lsrr|lsrre|ssrr|satid|any }
- string **ip_proto.~proto**: [!|>|<] name or number
- bool **ips.enable_builtin_rules** = false: enable events from builtin rules w/o stubs
- int **ips.id** = 0: correlate unified2 events with configuration { 0:65535 }
- string **ips.include**: legacy snort rules and includes
- enum **ips.mode**: set policy mode { tap | inline | inline-test }
- string **ips.rules**: snort rules and includes
- string **isdataat.~length**: num | !num
- implied **isdataat.relative**: offset from cursor instead of start of buffer
- string **itype.~range**: check if icmp type is *type* | *min*<>*max* | <*max* | >*min*
- enum **latency.packet.action** = none: event action if packet times out and is fastpathed { none | alert | log | alert_and_log }
- bool **latency.packet.fastpath** = false: fastpath expensive packets (max_time exceeded)
- int **latency.packet.max_time** = 500: set timeout for packet latency thresholding (usec) { 0: }
- enum **latency.rule.action** = none: event action for rule latency enable and suspend events { none | alert | log | alert_and_log }

- int **latency.rule.max_suspend_time** = 30000: set max time for suspending a rule (ms, 0 means permanently disable rule) { 0: }
- int **latency.rule.max_time** = 500: set timeout for rule evaluation (usec) { 0: }
- bool **latency.rule.suspend** = false: temporarily suspend expensive rules
- int **latency.rule.suspend_threshold** = 5: set threshold for number of timeouts before suspending a rule { 1: }
- bool **log_codecs.file** = false: output to log_codecs.txt instead of stdout
- bool **log_codecs.msg** = false: include alert msg
- bool **log_hext.file** = false: output to log_hext.txt instead of stdout
- int **log_hext.limit** = 0: set limit (0 is unlimited) { 0: }
- bool **log_hext.raw** = false: output all full packets if true, else just TCP payload
- enum **log_hext.units** = B: bytes | KB | MB | GB { B | K | M | G }
- int **log_hext.width** = 20: set line width (0 is unlimited) { 0: }
- int **log_pcap.limit** = 0: set limit (0 is unlimited) { 0: }
- enum **log_pcap.units** = B: bytes | KB | MB | GB { B | K | M | G }
- string **md5.~hash**: data to match
- int **md5.length**: number of octets in plain text { 1:65535 }
- string **md5.offset**: var or number of bytes from start of buffer to start search
- implied **md5.relative** = false: offset from cursor instead of start of buffer
- int **memory.cap** = 0: set the per-packet-thread cap on memory (bytes, 0 to disable) { 0: }
- bool **memory.soft** = false: always succeed in allocating memory, even if above the cap
- int **memory.threshold** = 0: set the per-packet-thread threshold for preemptive cleanup actions (percent, 0 to disable) { 0: }
- string **metadata.***: additional parameters not used by snort
- string **metadata.service**: service name
- string **modbus_func.~**: function code to match
- int **modbus_unit.~**: modbus unit ID { 0:255 }
- bool **mpls.enable_mpls_multicast** = false: enables support for MPLS multicast
- bool **mpls.enable_mpls_overlapping_ip** = false: enable if private network addresses overlap and must be differentiated by MPLS label(s)
- int **mpls.max_mpls_stack_depth** = -1: set MPLS stack depth { -1: }
- enum **mpls.mpls_payload_type** = ip4: set encapsulated payload type { eth | ip4 | ip6 }
- string **msg.~**: message describing rule
- multi **network.checksum_drop** = none: drop if checksum is bad { all | ip | noip | tcp | notcp | udp | noudp | icmp | noicmp | none }
- multi **network.checksum_eval** = none: checksums to verify { all | ip | noip | tcp | notcp | udp | noudp | icmp | noicmp | none }
- bool **network.decode_drops** = false: enable dropping of packets by the decoder
- int **network.id** = 0: correlate unified2 events with configuration { 0:65535 }

- int **network.layers** = 40: the maximum number of protocols that Snort can correctly decode { 3:255 }
- int **network.max_ip6_extensions** = 0: the maximum number of IP6 options Snort will process for a given IPv6 layer before raising 116:456 (0 = unlimited) { 0:255 }
- int **network.max_ip_layers** = 0: the maximum number of IP layers Snort will process for a given packet before raising 116:293 (0 = unlimited) { 0:255 }
- int **network.min_ttl** = 1: alert / normalize packets with lower ttl / hop limit (you must enable rules and / or normalization also) { 1:255 }
- int **network.new_ttl** = 1: use this value for responses and when normalizing { 1:255 }
- bool **normalizer.icmp4** = false: clear reserved flag
- bool **normalizer.icmp6** = false: clear reserved flag
- bool **normalizer.ip4.base** = true: clear options
- bool **normalizer.ip4.df** = false: clear don’t frag flag
- bool **normalizer.ip4.rf** = false: clear reserved flag
- bool **normalizer.ip4.tos** = false: clear tos / differentiated services byte
- bool **normalizer.ip4.trim** = false: truncate excess payload beyond datagram length
- bool **normalizer.ip6** = false: clear reserved flag
- string **normalizer.tcp.allow_codes**: don’t clear given option codes
- multi **normalizer.tcp.allow_names**: don’t clear given option names { sack | echo | partial_order | conn_count | alt_checksum | md5 }
- bool **normalizer.tcp.base** = true: clear reserved bits and option padding and fix urgent pointer / flags issues
- bool **normalizer.tcp.block** = true: allow packet drops during TCP normalization
- select **normalizer.tcp.ecn** = off: clear ecn for all packets | sessions w/o ecn setup { off | packet | stream }
- bool **normalizer.tcp.ips** = false: ensure consistency in retransmitted data
- bool **normalizer.tcp.opts** = true: clear all options except mss, wscale, timestamp, and any explicitly allowed
- bool **normalizer.tcp.pad** = true: clear any option padding bytes
- bool **normalizer.tcp.req_pay** = true: clear the urgent pointer and the urgent flag if there is no payload
- bool **normalizer.tcp.req_urg** = true: clear the urgent pointer if the urgent flag is not set
- bool **normalizer.tcp.req_urp** = true: clear the urgent flag if the urgent pointer is not set
- bool **normalizer.tcp.rsv** = true: clear the reserved bits in the TCP header
- bool **normalizer.tcp.trim** = false: enable all of the TCP trim options
- bool **normalizer.tcp.trim_mss** = false: trim data to MSS
- bool **normalizer.tcp.trim_rst** = false: remove any data from RST packet
- bool **normalizer.tcp.trim_syn** = false: remove data on SYN
- bool **normalizer.tcp.trim_win** = false: trim data to window
- bool **normalizer.tcp.urp** = true: adjust urgent pointer if beyond segment length
- bool **output.dump_chars_only** = false: turns on character dumps (same as -C)

- bool **output.dump_payload** = false: dumps application layer (same as -d)
- bool **output.dump_payload_verbose** = false: dumps raw packet starting at link layer (same as -X)
- int **output.event_trace.max_data** = 0: maximum amount of packet data to capture { 0:65535 }
- string **output.logdir** = .: where to put log files (same as -l)
- bool **output.log_ipv6_extra_data** = false: log IPv6 source and destination addresses as unified2 extra data records
- bool **output.obfuscate** = false: obfuscate the logged IP addresses (same as -O)
- bool **output.obfuscate_pii** = false: mask all but the last 4 characters of credit card and social security numbers
- bool **output.quiet** = false: suppress non-fatal information (still show alerts, same as -q)
- bool **output.show_year** = false: include year in timestamp in the alert and log files (same as -y)
- int **output.tagged_packet_limit** = 256: maximum number of packets tagged for non-packet metrics { 0: }
- bool **output.verbose** = false: be verbose (same as -v)
- bool **packet_capture.enable** = false: initially enable packet dumping
- string **packet_capture.filter**: bpf filter to use for packet dump
- bool **packets.address_space_agnostic** = false: determines whether DAQ address space info is used to track fragments and connections
- string **packets.bpf_file**: file with BPF to select traffic for Snort
- int **packets.limit** = 0: maximum number of packets to process before stopping (0 is unlimited) { 0: }
- int **packets.skip** = 0: number of packets to skip before before processing { 0: }
- bool **packets.vlan_agnostic** = false: determines whether VLAN info is used to track fragments and connections
- string **pcre.~re**: Snort regular expression
- bool **perf_monitor.base** = true: enable base statistics { nullptr }
- bool **perf_monitor.cpu** = false: enable cpu statistics { nullptr }
- bool **perf_monitor.flow** = false: enable traffic statistics
- bool **perf_monitor.flow_ip** = false: enable statistics on host pairs
- int **perf_monitor.flow_ip_memcap** = 52428800: maximum memory in bytes for flow tracking { 8200: }
- int **perf_monitor.flow_ports** = 1023: maximum ports to track { 0:65535 }
- enum **perf_monitor.format** = csv: output format for stats { csv | text }
- int **perf_monitor.max_file_size** = 1073741824: files will be rolled over if they exceed this size { 4096: }
- string **perf_monitor.modules[].name**: name of the module
- string **perf_monitor.modules[].pegs**: list of statistics to track or empty for all counters
- enum **perf_monitor.output** = file: output location for stats { file | console }
- int **perf_monitor.packets** = 10000: minimum packets to report { 0: }
- int **perf_monitor.seconds** = 60: report interval { 1: }
- bool **perf_monitor.summary** = false: output summary at shutdown
- int **pop.b64_decode_depth** = 1460: base64 decoding depth { -1:65535 }

- int **pop.bitenc_decode_depth** = 1460: Non-Encoded MIME attachment extraction depth { -1:65535 }
- int **pop.qp_decode_depth** = 1460: Quoted Printable decoding depth { -1:65535 }
- int **pop.uu_decode_depth** = 1460: Unix-to-Unix decoding depth { -1:65535 }
- int **port_scan_global.memcap** = 1048576: maximum tracker memory in bytes { 1: }
- string **port_scan.ignore_scanned**: list of CIDRs with optional ports to ignore if the destination of scan alerts
- string **port_scan.ignore_scanners**: list of CIDRs with optional ports to ignore if the source of scan alerts
- bool **port_scan.include_midstream** = false: list of CIDRs with optional ports
- bool **port_scan.logfile** = false: write scan events to file
- multi **port_scan.protos** = all: choose the protocols to monitor { tcp | udp | icmp | ip | all }
- multi **port_scan.scan_types** = all: choose type of scans to look for { portscan | portsweep | decoy_portscan | distributed_portscan | all }
- enum **port_scan.sense_level** = medium: choose the level of detection { low | medium | high }
- string **port_scan.watch_ip**: list of CIDRs with optional ports to watch
- int **priority.~**: relative severity level; 1 is highest priority { 1: }
- string **process.chroot**: set chroot directory (same as -t)
- bool **process.daemon** = false: fork as a daemon (same as -D)
- bool **process.dirty_pig** = false: shutdown without internal cleanup
- string **process.set_gid**: set group ID (same as -g)
- string **process.set_uid**: set user ID (same as -u)
- string **process.threads[].cpuset**: pin the associated thread to this cpuset
- int **process.threads[].thread** = 0: set cpu affinity for the <cur_thread_num> thread that runs { 0: }
- string **process.umask**: set process umask (same as -m)
- bool **process.utc** = false: use UTC instead of local time for timestamps
- int **profiler.memory.count** = 0: limit results to count items per level (0 = no limit) { 0: }
- int **profiler.memory.max_depth** = -1: limit depth to max_depth (-1 = no limit) { -1: }
- bool **profiler.memory.show** = true: show module memory profile stats
- enum **profiler.memory.sort** = total_used: sort by given field { none | allocations | total_used | avg_allocation }
- int **profiler.modules.count** = 0: limit results to count items per level (0 = no limit) { 0: }
- int **profiler.modules.max_depth** = -1: limit depth to max_depth (-1 = no limit) { -1: }
- bool **profiler.modules.show** = true: show module time profile stats
- enum **profiler.modules.sort** = total_time: sort by given field { none | checks | avg_check | total_time }
- int **profiler.rules.count** = 0: print results to given level (0 = all) { 0: }
- bool **profiler.rules.show** = true: show rule time profile stats
- enum **profiler.rules.sort** = total_time: sort by given field { none | checks | avg_check | total_time | matches | no_matches | avg_match | avg_no_match }
- string **rate_filter[].apply_to**: restrict filter to these addresses according to track

- int **rate_filter[].count** = 1: number of events in interval before tripping { 0: }
- int **rate_filter[].gid** = 1: rule generator ID { 0: }
- enum **rate_filter[].new_action** = alert: take this action on future hits until timeout { log | pass | alert | drop | block | reset }
- int **rate_filter[].seconds** = 1: count interval { 0: }
- int **rate_filter[].sid** = 1: rule signature ID { 0: }
- int **rate_filter[].timeout** = 1: count interval { 0: }
- enum **rate_filter[].track** = by_src: filter only matching source or destination addresses { by_src | by_dst | by_rule }
- bool **react.msg** = false: use rule msg in response page instead of default message
- string **react.page**: file containing HTTP response (headers and body)
- string **reference.~id**: reference id
- string **reference.~scheme**: reference scheme
- string **references[].name**: name used with reference rule option
- string **references[].url**: where this reference is defined
- implied **regex.dotall**: matching a . will not exclude newlines
- implied **regex.multiline**: ^ and $ anchors match any newlines in data
- implied **regex.nocase**: case insensitive match
- string **regex.~re**: hyperscan regular expression
- implied **regex.relative**: start search from end of last match instead of start of buffer
- enum **reject.control**: send icmp unreachable(s) { network|host|port|all }
- enum **reject.reset**: send tcp reset to one or both ends { source|dest|both }
- string **rem.~**: comment
- string **replace.~**: byte code to replace with
- string **reputation.blacklist**: blacklist file name with ip lists
- int **reputation.memcap** = 500: maximum total MB of memory allocated { 1:4095 }
- enum **reputation.nested_ip** = inner: ip to use when there is IP encapsulation { inner|outer|all }
- enum **reputation.priority** = whitelist: defines priority when there is a decision conflict during run-time { blacklist|whitelist }
- bool **reputation.scan_local** = false: inspect local address defined in RFC 1918
- string **reputation.whitelist**: whitelist file name with ip lists
- enum **reputation.white** = unblack: specify the meaning of whitelist { unblack|trust }
- int **rev.~**: revision { 1: }
- int **rpc.~app**: application number
- int **rpc.proc**: procedure number or * for any
- int **rpc.ver**: version number or * for any
- bool **rule_state.enable** = true: enable or disable rule in all policies
- int **rule_state.gid** = 0: rule generator ID { 0: }

- int **rule_state.sid** = 0: rule signature ID { 0: }
- string **sd_pattern.~pattern**: The pattern to search for
- int **sd_pattern.threshold**: number of matches before alerting { 1 }
- int **search_engine.bleedover_port_limit** = 1024: maximum ports in rule before demotion to any-any port group { 1: }
- bool **search_engine.bleedover_warnings_enabled** = false: print warning if a rule is demoted to any-any port group
- bool **search_engine.debug** = false: print verbose fast pattern info
- bool **search_engine.debug_print_nocontent_rule_tests** = false: print rule group info during packet evaluation
- bool **search_engine.debug_print_rule_group_build_details** = false: print rule group info during compilation
- bool **search_engine.debug_print_rule_groups_compiled** = false: prints compiled rule group information
- bool **search_engine.debug_print_rule_groups_uncompiled** = false: prints uncompiled rule group information
- bool **search_engine.enable_single_rule_group** = false: put all rules into one group
- bool **search_engine.inspect_stream_inserts** = false: inspect reassembled payload - disabling is good for performance, bad for detection
- int **search_engine.max_pattern_len** = 0: truncate patterns when compiling into state machine (0 means no maximum) { 0: }
- int **search_engine.max_queue_events** = 5: maximum number of matching fast pattern states to queue per packet
- dynamic **search_engine.search_method** = ac_bnfa: set fast pattern algorithm - choose available search engine { ac_banded | ac_bnfa | ac_full | ac_sparse | ac_sparse_bands | ac_std | hyperscan }
- bool **search_engine.search_optimize** = true: tweak state machine construction for better performance
- bool **search_engine.show_fast_patterns** = false: print fast pattern info for each rule
- bool **search_engine.split_any_any** = false: evaluate any-any rules separately to save memory
- string **seq.~range**: check if tcp sequence number value is *value* | *min<>max* | *<max* | *>min*
- enum **session.~mode**: output format { printable|binary|all }
- string **sha256.~hash**: data to match
- int **sha256.length**: number of octets in plain text { 1:65535 }
- string **sha256.offset**: var or number of bytes from start of buffer to start search
- implied **sha256.relative** = false: offset from cursor instead of start of buffer
- string **sha512.~hash**: data to match
- int **sha512.length**: number of octets in plain text { 1:65535 }
- string **sha512.offset**: var or number of bytes from start of buffer to start search
- implied **sha512.relative** = false: offset from cursor instead of start of buffer
- string **side_channel.connector**: connector handle
- string **side_channel.connectors[].connector**: connector handle
- bit_list **side_channel.ports**: side channel message port list { 65535 }
- int **sid.~**: signature id { 1: }
- bool **sip.ignore_call_channel** = false: enables the support for ignoring audio/video data channel
- int **sip.max_call_id_len** = 256: maximum call id field size { 0:65535 }

- int **sip.max_contact_len** = 256: maximum contact field size { 0:65535 }
- int **sip.max_content_len** = 1024: maximum content length of the message body { 0:65535 }
- int **sip.max_dialogs** = 4: maximum number of dialogs within one stream session { 1:4194303 }
- int **sip.max_from_len** = 256: maximum from field size { 0:65535 }
- int **sip.max_requestName_len** = 20: maximum request name field size { 0:65535 }
- int **sip.max_sessions** = 10000: maximum number of sessions that can be allocated { 1024:4194303 }
- int **sip.max_to_len** = 256: maximum to field size { 0:65535 }
- int **sip.max_uri_len** = 256: maximum request uri field size { 0:65535 }
- int **sip.max_via_len** = 1024: maximum via field size { 0:65535 }
- string **sip_method.*method**: sip method
- string **sip.methods** = invite cancel ack bye register options: list of methods to check in sip messages
- int **sip_stat_code.*code**: stat code { 1:999 }
- string **smtp.alt_max_command_line_len[].command**: command string
- int **smtp.alt_max_command_line_len[].length** = 0: specify non-default maximum for command { 0: }
- string **smtp.auth_cmds**: commands that initiate an authentication exchange
- int **smtp.b64_decode_depth** = 25: depth used to decode the base64 encoded MIME attachments { -1:65535 }
- string **smtp.binary_data_cmds**: commands that initiate sending of data and use a length value after the command
- int **smtp.bitenc_decode_depth** = 25: depth used to extract the non-encoded MIME attachments { -1:65535 }
- string **smtp.data_cmds**: commands that initiate sending of data with an end of data delimiter
- int **smtp.email_hdrs_log_depth** = 1464: depth for logging email headers { 0:20480 }
- bool **smtp.ignore_data** = false: ignore data section of mail
- bool **smtp.ignore_tls_data** = false: ignore TLS-encrypted data when processing rules
- string **smtp.invalid_cmds**: alert if this command is sent from client side
- bool **smtp.log_email_hdrs** = false: log the SMTP email headers extracted from SMTP data
- bool **smtp.log_filename** = false: log the MIME attachment filenames extracted from the Content-Disposition header within the MIME body
- bool **smtp.log_mailfrom** = false: log the sender's email address extracted from the MAIL FROM command
- bool **smtp.log_rcptto** = false: log the recipient's email address extracted from the RCPT TO command
- int **smtp.max_auth_command_line_len** = 1000: max auth command Line Length { 0:65535 }
- int **smtp.max_command_line_len** = 0: max Command Line Length { 0:65535 }
- int **smtp.max_header_line_len** = 0: max SMTP DATA header line { 0:65535 }
- int **smtp.max_response_line_len** = 0: max SMTP response line { 0:65535 }
- string **smtp.normalize_cmds**: list of commands to normalize
- enum **smtp.normalize** = none: turns on/off normalization { none | cmds | all }
- int **smtp.qp_decode_depth** = 25: quoted-Printable decoding depth { -1:65535 }

- int **smtp.uu_decode_depth** = 25: unix-to-Unix decoding depth { -1:65535 }
- string **smtp.valid_cmds**: list of valid commands
- enum **smtp.xlink2state** = alert: enable/disable xlink2state alert { disable | alert | drop }
- implied **snort.--alert-before-pass**: process alert, drop, sdrop, or reject before pass; default is pass before alert, drop,...
- string **snort.-A**: <mode> set alert mode: none, cmg, or alert_*
- addr **snort.-B** = 255.255.255.255/32: <mask> obfuscated IP addresses in alerts and packet dumps using CIDR mask
- string **snort.--bpf**: <filter options> are standard BPF options, as seen in TCPDump
- string **snort.--c2x**: output hex for given char (see also --x2c)
- string **snort.--catch-test**: comma separated list of cat unit test tags or *all*
- string **snort.-c**: <conf> use this configuration
- implied **snort.-C**: print out payloads with character data only (no hex)
- implied **snort.--create-pidfile**: create PID file, even when not in Daemon mode
- string **snort.--daq-dir**: <dir> tell snort where to find desired DAQ
- implied **snort.--daq-list**: list packet acquisition modules available in optional dir, default is static modules only
- string **snort.--daq**: <type> select packet acquisition module (default is pcap)
- string **snort.--daq-var**: <name=value> specify extra DAQ configuration variable
- implied **snort.-d**: dump the Application Layer
- implied **snort.--dirty-pig**: don't flush packets on shutdown
- implied **snort.-D**: run Snort in background (daemon) mode
- implied **snort.--dump-builtin-rules**: [<module prefix>] output stub rules for selected modules
- string **snort.--dump-defaults**: [<module prefix>] output module defaults in Lua format { (optional) }
- implied **snort.--dump-dynamic-rules**: output stub rules for all loaded rules libraries
- implied **snort.--dump-version**: output the version, the whole version, and only the version
- implied **snort.-e**: display the second layer header info
- implied **snort.--enable-inline-test**: enable Inline-Test Mode Operation
- implied **snort.-f**: turn off fflush() calls after binary log writes
- int **snort.-G**: <0xid> (same as --logid) { 0:65535 }
- string **snort.-g**: <gname> run snort gid as <gname> group (or gid) after initialization
- string **snort.--help-commands**: [<module prefix>] output matching commands { (optional) }
- string **snort.--help-config**: [<module prefix>] output matching config options { (optional) }
- string **snort.--help-counts**: [<module prefix>] output matching peg counts { (optional) }
- implied **snort.--help**: list command line options
- string **snort.--help-module**: <module> output description of given module
- implied **snort.--help-modules**: list all available modules with brief help
- string **snort.--help-options**: <option prefix> output matching command line option quick help (same as -?) { (optional) }

- implied **snort.--help-plugins**: list all available plugins with brief help
- implied **snort.--help-signals**: dump available control signals
- implied **snort.-H**: make hash tables deterministic
- implied **snort.--id-subdir**: create/use instance subdirectories in logdir instead of instance filename prefix
- implied **snort.--id-zero**: use id prefix / subdirectory even with one packet thread
- string **snort.-i**: <iface>… list of interfaces
- port **snort.-j**: <port> to listen for telnet connections
- enum **snort.-k** = all: <mode> checksum mode; default is all { all|noip|notcp|noudp|noicmp|none }
- implied **snort.--list-buffers**: output available inspection buffers
- string **snort.--list-builtin**: <module prefix> output matching builtin rules { (optional) }
- string **snort.--list-gids**: [<module prefix>] output matching generators { (optional) }
- string **snort.--list-modules**: [<module type>] list all known modules of given type { (optional) }
- implied **snort.--list-plugins**: list all known plugins
- string **snort.-l**: <logdir> log to this directory instead of current directory
- string **snort.-L**: <mode> logging mode (none, dump, pcap, or log_*)
- int **snort.--logid**: <0xid> log Identifier to uniquely id events for multiple snorts (same as -G) { 0:65535 }
- string **snort.--lua**: <chunk> extend/override conf with chunk; may be repeated
- implied **snort.--markup**: output help in asciidoc compatible format
- int **snort.--max-packet-threads** = 1: <count> configure maximum number of packet threads (same as -z) { 0: }
- implied **snort.-M**: log messages to syslog (not alerts)
- int **snort.-m**: <umask> set umask = <umask> { 0: }
- int **snort.-n**: <count> stop after count packets { 0: }
- implied **snort.--nolock-pidfile**: do not try to lock Snort PID file
- implied **snort.--nostamps**: don't include timestamps in log file names
- implied **snort.-O**: obfuscate the logged IP addresses
- string **snort.-?**: <option prefix> output matching command line option quick help (same as --help-options) { (optional) }
- implied **snort.--pause**: wait for resume/quit command before processing packets/terminating
- string **snort.--pcap-dir**: <dir> a directory to recurse to look for pcaps - read mode is implied
- string **snort.--pcap-file**: <file> file that contains a list of pcaps to read - read mode is implied
- string **snort.--pcap-filter**: <filter> filter to apply when getting pcaps from file or directory
- string **snort.--pcap-list**: <list> a space separated list of pcaps to read - read mode is implied
- int **snort.--pcap-loop**: <count> read all pcaps <count> times; 0 will read until Snort is terminated { -1: }
- implied **snort.--pcap-no-filter**: reset to use no filter when getting pcaps from file or directory
- implied **snort.--pcap-reload**: if reading multiple pcaps, reload snort config between pcaps
- implied **snort.--pcap-show**: print a line saying what pcap is currently being read

- implied **snort.--pedantic**: warnings are fatal
- implied **snort.--piglet**: enable piglet test harness mode
- string **snort.--plugin-path**: <path> where to find plugins
- implied **snort.--process-all-events**: process all action groups
- implied **snort.-Q**: enable inline mode operation
- implied **snort.-q**: quiet mode - Don’t show banner and status report
- string **snort.-r**: <pcap>… (same as --pcap-list)
- string **snort.-R**: <rules> include this rules file in the default policy
- string **snort.--rule**: <rules> to be added to configuration; may be repeated
- implied **snort.--rule-to-hex**: output so rule header to stdout for text rule on stdin
- implied **snort.--rule-to-text**: output plain so rule header to stdout for text rule on stdin
- string **snort.--run-prefix**: <pfx> prepend this to each output file
- int **snort.-s** = 1514: <snap> (same as --snaplen); default is 1514 { 68:65535 }
- string **snort.--script-path**: <path> to a luajit script or directory containing luajit scripts
- implied **snort.--shell**: enable the interactive command line
- implied **snort.--show-plugins**: list module and plugin versions
- int **snort.--skip**: <n> skip 1st n packets { 0: }
- int **snort.--snaplen** = 1514: <snap> set snaplen of packet (same as -s) { 68:65535 }
- implied **snort.--stdin-rules**: read rules from stdin until EOF or a line starting with END is read
- string **snort.-S**: <x=v> set config variable x equal to value v
- string **snort.-t**: <dir> chroots process to <dir> after initialization
- implied **snort.--treat-drop-as-alert**: converts drop, sdrop, and reject rules into alert rules during startup
- implied **snort.--treat-drop-as-ignore**: use drop, sdrop, and reject rules to ignore session traffic when not inline
- implied **snort.-T**: test and report on the current Snort configuration
- string **snort.-u**: <uname> run snort as <uname> or <uid> after initialization
- implied **snort.-U**: use UTC for timestamps
- implied **snort.-v**: be verbose
- implied **snort.--version**: show version number (same as -V)
- implied **snort.-V**: (same as --version)
- implied **snort.--warn-all**: enable all warnings
- implied **snort.--warn-conf**: warn about configuration issues
- implied **snort.--warn-daq**: warn about DAQ issues, usually related to mode
- implied **snort.--warn-flowbits**: warn about flowbits that are checked but not set and vice-versa
- implied **snort.--warn-hosts**: warn about host table issues
- implied **snort.--warn-plugins**: warn about issues that prevent plugins from loading

- implied **snort.--warn-rules**: warn about duplicate rules and rule parsing issues
- implied **snort.--warn-scripts**: warn about issues discovered while processing Lua scripts
- implied **snort.--warn-symbols**: warn about unknown symbols in your Lua config
- implied **snort.--warn-vars**: warn about variable definition and usage issues
- implied **snort.-W**: lists available interfaces
- int **snort.--x2c**: output ASCII char for given hex (see also --c2x)
- string **snort.--x2s**: output ASCII string for given byte code (see also --x2c)
- implied **snort.-X**: dump the raw packet data starting at the link layer
- implied **snort.-x**: same as --pedantic
- implied **snort.-y**: include year in timestamp in the alert and log files
- int **snort.-z** = 1: <count> maximum number of packet threads (same as --max-packet-threads); 0 gets the number of CPU cores reported by the system; default is 1 { 0: }
- string **so.~func**: name of eval function
- string **soid.~**: SO rule ID has <gid>|<sid> format, like 3|12345
- int **ssh.max_client_bytes** = 19600: number of unanswered bytes before alerting on challenge-response overflow or CRC32 { 0:65535 }
- int **ssh.max_encrypted_packets** = 25: ignore session after this many encrypted packets { 0:65535 }
- int **ssh.max_server_version_len** = 80: limit before alerting on secure CRT server version string overflow { 0:255 }
- int **ssl.max_heartbeat_length** = 0: maximum length of heartbeat record allowed { 0:65535 }
- implied **ssl_state.client_hello**: check for client hello
- implied **ssl_state.!client_hello**: check for records that are not client hello
- implied **ssl_state.client_keyx**: check for client keyx
- implied **ssl_state.!client_keyx**: check for records that are not client keyx
- implied **ssl_state.!server_hello**: check for records that are not server hello
- implied **ssl_state.server_hello**: check for server hello
- implied **ssl_state.!server_keyx**: check for records that are not server keyx
- implied **ssl_state.server_keyx**: check for server keyx
- implied **ssl_state.!unknown**: check for records that are not unknown
- implied **ssl_state.unknown**: check for unknown record
- bool **ssl.trust_servers** = false: disables requirement that application (encrypted) data must be observed on both sides
- implied **ssl_version.!sslv2**: check for records that are not sslv2
- implied **ssl_version.sslv2**: check for sslv2
- implied **ssl_version.!sslv3**: check for records that are not sslv3
- implied **ssl_version.sslv3**: check for sslv3
- implied **ssl_version.!tls1.0**: check for records that are not tls1.0
- implied **ssl_version.tls1.0**: check for tls1.0

- implied **ssl_version.!tls1.1**: check for records that are not tls1.1
- implied **ssl_version.tls1.1**: check for tls1.1
- implied **ssl_version.!tls1.2**: check for records that are not tls1.2
- implied **ssl_version.tls1.2**: check for tls1.2
- int **stream.file_cache.idle_timeout** = 180: maximum inactive time before retiring session tracker { 1: }
- int **stream.file_cache.max_sessions** = 128: maximum simultaneous sessions tracked before pruning { 2: }
- int **stream.file_cache.pruning_timeout** = 30: minimum inactive time before being eligible for pruning { 1: }
- bool **stream_file.upload** = false: indicate file transfer direction
- int **stream.icmp_cache.idle_timeout** = 180: maximum inactive time before retiring session tracker { 1: }
- int **stream.icmp_cache.max_sessions** = 65536: maximum simultaneous sessions tracked before pruning { 2: }
- int **stream.icmp_cache.pruning_timeout** = 30: minimum inactive time before being eligible for pruning { 1: }
- int **stream_icmp.session_timeout** = 30: session tracking timeout { 1:86400 }
- int **stream.ip_cache.idle_timeout** = 180: maximum inactive time before retiring session tracker { 1: }
- int **stream.ip_cache.max_sessions** = 16384: maximum simultaneous sessions tracked before pruning { 2: }
- int **stream.ip_cache.pruning_timeout** = 30: minimum inactive time before being eligible for pruning { 1: }
- bool **stream.ip_frags_only** = false: don’t process non-frag flows
- int **stream_ip.max_frags** = 8192: maximum number of simultaneous fragments being tracked { 1: }
- int **stream_ip.max_overlaps** = 0: maximum allowed overlaps per datagram; 0 is unlimited { 0: }
- int **stream_ip.min_frag_length** = 0: alert if fragment length is below this limit before or after trimming { 0: }
- int **stream_ip.min_ttl** = 1: discard fragments with ttl below the minimum { 1:255 }
- enum **stream_ip.policy** = linux: fragment reassembly policy { first | linux | bsd | bsd_right | last | windows | solaris }
- int **stream_ip.session_timeout** = 30: session tracking timeout { 1:86400 }
- int **stream_ip.trace**: mask for enabling debug traces in module
- enum **stream_reassemble.action**: stop or start stream reassembly { disable|enable }
- enum **stream_reassemble.direction**: action applies to the given direction(s) { client|server|both }
- implied **stream_reassemble.fastpath**: optionally whitelist the remainder of the session
- implied **stream_reassemble.noalert**: don’t alert when rule matches
- enum **stream_size.~direction**: compare applies to the given direction(s) { either|to_server|to_client|both }
- string **stream_size.~range**: size for comparison
- int **stream.tcp_cache.idle_timeout** = 180: maximum inactive time before retiring session tracker { 1: }
- int **stream.tcp_cache.max_sessions** = 262144: maximum simultaneous sessions tracked before pruning { 2: }
- int **stream.tcp_cache.pruning_timeout** = 30: minimum inactive time before being eligible for pruning { 1: }
- int **stream_tcp.flush_factor** = 0: flush upon seeing a drop in segment size after given number of non-decreasing segments { 0: }
- int **stream_tcp.footprint** = 0: use zero for production, non-zero for testing at given size { 0: }

- bool **stream_tcp.ignore_any_rules** = false: process tcp content rules w/o ports only if rules with ports are present
- int **stream_tcp.max_pdu** = 16384: maximum reassembled PDU size { 1460:65535 }
- int **stream_tcp.max_window** = 0: maximum allowed tcp window { 0:1073725440 }
- int **stream_tcp.overlap_limit** = 0: maximum number of allowed overlapping segments per session { 0:255 }
- enum **stream_tcp.policy** = bsd: determines operating system characteristics like reassembly { first | last | linux | old_linux | bsd | macos | solaris | irix | hpux11 | hpux10 | windows | win_2003 | vista | proxy }
- int **stream_tcp.queue_limit.max_bytes** = 1048576: don't queue more than given bytes per session and direction { 0: }
- int **stream_tcp.queue_limit.max_segments** = 2621: don't queue more than given segments per session and direction { 0: }
- bool **stream_tcp.reassemble_async** = true: queue data for reassembly before traffic is seen in both directions
- int **stream_tcp.require_3whs** = -1: don't track midstream sessions after given seconds from start up; -1 tracks all { -1:86400 }
- int **stream_tcp.session_timeout** = 30: session tracking timeout { 1:86400 }
- bool **stream_tcp.show_rebuilt_packets** = false: enable cmg like output of reassembled packets
- int **stream_tcp.small_segments.count** = 0: limit number of small segments queued { 0:2048 }
- int **stream_tcp.small_segments.maximum_size** = 0: limit number of small segments queued { 0:2048 }
- int **stream.udp_cache.idle_timeout** = 180: maximum inactive time before retiring session tracker { 1: }
- int **stream.udp_cache.max_sessions** = 131072: maximum simultaneous sessions tracked before pruning { 2: }
- int **stream.udp_cache.pruning_timeout** = 30: minimum inactive time before being eligible for pruning { 1: }
- bool **stream_udp.ignore_any_rules** = false: process udp content rules w/o ports only if rules with ports are present
- int **stream_udp.session_timeout** = 30: session tracking timeout { 1:86400 }
- int **stream.user_cache.idle_timeout** = 180: maximum inactive time before retiring session tracker { 1: }
- int **stream.user_cache.max_sessions** = 1024: maximum simultaneous sessions tracked before pruning { 2: }
- int **stream.user_cache.pruning_timeout** = 30: minimum inactive time before being eligible for pruning { 1: }
- int **stream_user.session_timeout** = 30: session tracking timeout { 1:86400 }
- int **suppress[].gid** = 0: rule generator ID { 0: }
- string **suppress[].ip**: restrict suppression to these addresses according to track
- int **suppress[].sid** = 0: rule signature ID { 0: }
- enum **suppress[].track**: suppress only matching source or destination addresses { by_src | by_dst }
- int **tag.bytes**: tag for this many bytes { 1: }
- enum **tag.~**: log all packets in session or all packets to or from host { session|host_src|host_dst }
- int **tag.packets**: tag this many packets { 1: }
- int **tag.seconds**: tag for this many seconds { 1: }
- string **tcp_connector.address**: address
- port **tcp_connector.base_port**: base port number
- string **tcp_connector.connector**: connector name
- enum **tcp_connector.setup**: stream establishment { call | answer }

- int **telnet.ayt_attack_thresh** = -1: alert on this number of consecutive telnet AYT commands { -1: }
- bool **telnet.check_encrypted** = false: check for end of encryption
- bool **telnet.encrypted_traffic** = false: check for encrypted telnet and ftp
- bool **telnet.normalize** = false: eliminate escape sequences
- string **tos.~range**: check if ip tos value is *value* | *min<>max* | *<max* | *>min*
- string **ttl.~range**: check if ip ttl field value is *value* | *min<>max* | *<max* | *>min*
- bool **udp.deep_teredo_inspection** = false: look for Teredo on all UDP ports (default is only 3544)
- bool **udp.enable_gtp** = false: decode GTP encapsulations
- bit_list **udp.gtp_ports** = 2152 3386: set GTP ports { 65535 }
- int **unified2.limit** = 0: set limit (0 is unlimited) { 0: }
- bool **unified2.mpls_event_types** = false: include mpls labels in events
- bool **unified2.nostamp** = true: append file creation time to name (in Unix Epoch format)
- enum **unified2.units** = B: limit multiplier { B | K | M | G }
- bool **unified2.vlan_event_types** = false: include vlan IDs in events
- string **window.~range**: check if tcp window field size is *size* | *min<>max* | *<max* | *>min*
- multi **wizard.curses**: enable service identification based on internal algorithm { dce_smb | dce_udp | dce_tcp }
- bool **wizard.hexes[].client_first** = true: which end initiates data transfer
- select **wizard.hexes[].proto** = tcp: protocol to scan { tcp | udp }
- string **wizard.hexes[].service**: name of service
- string **wizard.hexes[].to_client[].hex**: sequence of data with wild chars (?)
- string **wizard.hexes[].to_server[].hex**: sequence of data with wild chars (?)
- bool **wizard.spells[].client_first** = true: which end initiates data transfer
- select **wizard.spells[].proto** = tcp: protocol to scan { tcp | udp }
- string **wizard.spells[].service**: name of service
- string **wizard.spells[].to_client[].spell**: sequence of data with wild cards (*)
- string **wizard.spells[].to_server[].spell**: sequence of data with wild cards (*)

19.5 Counts

- **appid.aim clients**: count of aim clients discovered
- **appid.battlefield flows**: count of battle field flows discovered
- **appid.bgp flows**: count of bgp flows discovered
- **appid.bit clients**: count of bittorrent clients discovered
- **appid.bit flows**: count of bittorrent flows discovered
- **appid.bittracker clients**: count of bittorrent tracker clients discovered
- **appid.bootp flows**: count of bootp flows discovered

- **appid.dcerpc tcp flows**: count of dce rpc flows over tcp discovered
- **appid.dcerpc udp flows**: count of dce rpc flows over udp discovered
- **appid.direct connect flows**: count of direct connect flows discovered
- **appid.dns tcp flows**: count of dns flows over tcp discovered
- **appid.dns udp flows**: count of dns flows over udp discovered
- **appid.ftp flows**: count of ftp flows discovered
- **appid.ftps flows**: count of ftps flows discovered
- **appid.http flows**: count of http flows discovered
- **appid.ignored packets**: count of packets ignored
- **appid.imap flows**: count of imap service flows discovered
- **appid.imaps flows**: count of imap TLS service flows discovered
- **appid.irc flows**: count of irc service flows discovered
- **appid.kerberos clients**: count of kerberos clients discovered
- **appid.kerberos flows**: count of kerberos service flows discovered
- **appid.kerberos users**: count of kerberos users discovered
- **appid.lpr flows**: count of lpr service flows discovered
- **appid.mdns flows**: count of mdns service flows discovered
- **appid.msn clients**: count of msn clients discovered
- **appid.mysql flows**: count of mysql service flows discovered
- **appid.netbios dgm flows**: count of netbios-dgm service flows discovered
- **appid.netbios ns flows**: count of netbios-ns service flows discovered
- **appid.netbios ssn flows**: count of netbios-ssn service flows discovered
- **appid.nntp flows**: count of nntp flows discovered
- **appid.ntp flows**: count of ntp flows discovered
- **appid.packets**: count of packets received
- **appid.pop flows**: count of pop service flows discovered
- **appid.processed packets**: count of packets processed
- **appid.radius flows**: count of radius flows discovered
- **appid.rexec flows**: count of rexec flows discovered
- **appid.rfb flows**: count of rfb flows discovered
- **appid.rlogin flows**: count of rlogin flows discovered
- **appid.rpc flows**: count of rpc flows discovered
- **appid.rshell flows**: count of rshell flows discovered
- **appid.rsync flows**: count of rsync service flows discovered
- **appid.rtmp flows**: count of rtmp flows discovered

- **appid.rtp clients**: count of rtp clients discovered
- **appid.sip clients**: count of SIP clients discovered
- **appid.sip flows**: count of SIP flows discovered
- **appid.smtp aol clients**: count of AOL smtp clients discovered
- **appid.smtp applemail clients**: count of Apple Mail smtp clients discovered
- **appid.smtp eudora clients**: count of Eudora smtp clients discovered
- **appid.smtp eudora pro clients**: count of Eudora Pro smtp clients discovered
- **appid.smtp evolution clients**: count of Evolution smtp clients discovered
- **appid.smtp flows**: count of smtp flows discovered
- **appid.smtp kmail clients**: count of KMail smtp clients discovered
- **appid.smtp lotus notes clients**: count of Lotus Notes smtp clients discovered
- **appid.smtp microsoft outlook clients**: count of Microsoft Outlook smtp clients discovered
- **appid.smtp microsoft outlook express clients**: count of Microsoft Outlook Express smtp clients discovered
- **appid.smtp microsoft outlook imo clients**: count of Microsoft Outlook IMO smtp clients discovered
- **appid.smtp mutt clients**: count of Mutt smtp clients discovered
- **appid.smtps flows**: count of smtps flows discovered
- **appid.smtp thunderbird clients**: count of Thunderbird smtp clients discovered
- **appid.snmp flows**: count of snmp flows discovered
- **appid.ssh clients**: count of ssh clients discovered
- **appid.ssh flows**: count of ssh flows discovered
- **appid.ssl flows**: count of ssl flows discovered
- **appid.telnet flows**: count of telnet flows discovered
- **appid.tftp flows**: count of tftp flows discovered
- **appid.timbuktu flows**: count of timbuktu flows discovered
- **appid.tns clients**: count of tns clients discovered
- **appid.tns flows**: count of tns flows discovered
- **appid.vnc clients**: count of vnc clients discovered
- **appid.yahoo messenger clients**: count of Yahoo Messenger clients discovered
- **arp_spoof.packets**: total packets
- **back_orifice.packets**: total packets
- **binder.allows**: allow bindings
- **binder.blocks**: block bindings
- **binder.inspects**: inspect bindings
- **binder.packets**: initial bindings
- **binder.resets**: reset bindings

- **daq.allow**: total allow verdicts
- **daq.analyzed**: total packets analyzed from DAQ
- **daq.blacklist**: total blacklist verdicts
- **daq.block**: total block verdicts
- **daq.dropped**: packets dropped
- **daq.filtered**: packets filtered out
- **daq.idle**: attempts to acquire from DAQ without available packets
- **daq.ignore**: total ignore verdicts
- **daq.injected**: active responses or replacements
- **daq.internal blacklist**: packets blacklisted internally due to lack of DAQ support
- **daq.internal whitelist**: packets whitelisted internally due to lack of DAQ support
- **daq.outstanding**: packets unprocessed
- **daq.pcaps**: total files and interfaces processed
- **daq.received**: total packets received from DAQ
- **daq.replace**: total replace verdicts
- **daq.skipped**: packets skipped at startup
- **daq.whitelist**: total whitelist verdicts
- **dce_smb.Alter context responses**: total connection-oriented alter context responses
- **dce_smb.Alter contexts**: total connection-oriented alter contexts
- **dce_smb.Auth3s**: total connection-oriented auth3s
- **dce_smb.Bind acks**: total connection-oriented binds acks
- **dce_smb.Bind naks**: total connection-oriented bind naks
- **dce_smb.Binds**: total connection-oriented binds
- **dce_smb.Cancels**: total connection-oriented cancels
- **dce_smb.Client frags reassembled**: total connection-oriented client fragments reassembled
- **dce_smb.Client max fragment size**: connection-oriented client maximum fragment size
- **dce_smb.Client min fragment size**: connection-oriented client minimum fragment size
- **dce_smb.Client segs reassembled**: total connection-oriented client segments reassembled
- **dce_smb.Client segs reassembled**: total smb client segments reassembled
- **dce_smb.events**: total events
- **dce_smb.Faults**: total connection-oriented faults
- **dce_smb.Files processed**: total smb files processed
- **dce_smb.Ignored bytes**: total ignored bytes
- **dce_smb.Max outstanding requests**: total smb maximum outstanding requests
- **dce_smb.MS RPC/HTTP PDUs**: total connection-oriented MS requests to send RPC over HTTP

- **dce_smb.Orphaned**: total connection-oriented orphaned
- **dce_smb.Other requests**: total connection-oriented other requests
- **dce_smb.Other responses**: total connection-oriented other responses
- **dce_smb.Packets**: total smb packets
- **dce_smb.PDUs**: total connection-oriented PDUs
- **dce_smb.Rejects**: total connection-oriented rejects
- **dce_smb.Request fragments**: total connection-oriented request fragments
- **dce_smb.Requests**: total connection-oriented requests
- **dce_smb.Response fragments**: total connection-oriented response fragments
- **dce_smb.Responses**: total connection-oriented responses
- **dce_smb.Server frags reassembled**: total connection-oriented server fragments reassembled
- **dce_smb.Server max fragment size**: connection-oriented server maximum fragment size
- **dce_smb.Server min fragment size**: connection-oriented server minimum fragment size
- **dce_smb.Server segs reassembled**: total connection-oriented server segments reassembled
- **dce_smb.Server segs reassembled**: total smb server segments reassembled
- **dce_smb.Sessions**: total smb sessions
- **dce_smb.Shutdowns**: total connection-oriented shutdowns
- **dce_smb.SMBv2 close**: total number of SMBv2 close packets seen
- **dce_smb.SMBv2 create**: total number of SMBv2 create packets seen
- **dce_smb.SMBv2 read**: total number of SMBv2 read packets seen
- **dce_smb.SMBv2 set info**: total number of SMBv2 set info packets seen
- **dce_smb.SMBv2 tree connect**: total number of SMBv2 tree connect packets seen
- **dce_smb.SMBv2 tree disconnect**: total number of SMBv2 tree disconnect packets seen
- **dce_smb.SMBv2 write**: total number of SMBv2 write packets seen
- **dce_tcp.Alter context responses**: total connection-oriented alter context responses
- **dce_tcp.Alter contexts**: total connection-oriented alter contexts
- **dce_tcp.Auth3s**: total connection-oriented auth3s
- **dce_tcp.Bind acks**: total connection-oriented binds acks
- **dce_tcp.Bind naks**: total connection-oriented bind naks
- **dce_tcp.Binds**: total connection-oriented binds
- **dce_tcp.Cancels**: total connection-oriented cancels
- **dce_tcp.Client frags reassembled**: total connection-oriented client fragments reassembled
- **dce_tcp.Client max fragment size**: connection-oriented client maximum fragment size
- **dce_tcp.Client min fragment size**: connection-oriented client minimum fragment size
- **dce_tcp.Client segs reassembled**: total connection-oriented client segments reassembled

- **dce_tcp.events**: total events
- **dce_tcp.Faults**: total connection-oriented faults
- **dce_tcp.MS RPC/HTTP PDUs**: total connection-oriented MS requests to send RPC over HTTP
- **dce_tcp.Orphaned**: total connection-oriented orphaned
- **dce_tcp.Other requests**: total connection-oriented other requests
- **dce_tcp.Other responses**: total connection-oriented other responses
- **dce_tcp.PDUs**: total connection-oriented PDUs
- **dce_tcp.Rejects**: total connection-oriented rejects
- **dce_tcp.Request fragments**: total connection-oriented request fragments
- **dce_tcp.Requests**: total connection-oriented requests
- **dce_tcp.Response fragments**: total connection-oriented response fragments
- **dce_tcp.Responses**: total connection-oriented responses
- **dce_tcp.Server frags reassembled**: total connection-oriented server fragments reassembled
- **dce_tcp.Server max fragment size**: connection-oriented server maximum fragment size
- **dce_tcp.Server min fragment size**: connection-oriented server minimum fragment size
- **dce_tcp.Server segs reassembled**: total connection-oriented server segments reassembled
- **dce_tcp.Shutdowns**: total connection-oriented shutdowns
- **dce_tcp.tcp packets**: total tcp packets
- **dce_tcp.tcp sessions**: total tcp sessions
- **dce_udp.Acks**: total connection-less acks
- **dce_udp.Cancel acks**: total connection-less cancel acks
- **dce_udp.Cancels**: total connection-less cancels
- **dce_udp.Client facks**: total connection-less client facks
- **dce_udp.events**: total events
- **dce_udp.Faults**: total connection-less faults
- **dce_udp.Fragments**: total connection-less fragments
- **dce_udp.Frags reassembled**: total connection-less fragments reassembled
- **dce_udp.Max fragment size**: connection-less maximum fragment size
- **dce_udp.Max seqnum**: max connection-less seqnum
- **dce_udp.No calls**: total connection-less no calls
- **dce_udp.Other requests**: total connection-less other requests
- **dce_udp.Other responses**: total connection-less other responses
- **dce_udp.Ping**: total connection-less ping
- **dce_udp.Rejects**: total connection-less rejects
- **dce_udp.Requests**: total connection-less requests

- **dce_udp.Responses**: total connection-less responses
- **dce_udp.Server facks**: total connection-less server facks
- **dce_udp.udp packets**: total udp packets
- **dce_udp.udp sessions**: total udp sessions
- **dce_udp.Working**: total connection-less working
- **detection.alert limit**: events previously triggered on same PDU
- **detection.alerts**: alerts not including IP reputation
- **detection.alt searches**: alt fast pattern searches in packet data
- **detection.analyzed**: packets sent to detection
- **detection.body searches**: fast pattern searches in body buffer
- **detection.cooked searches**: fast pattern searches in cooked packet data
- **detection.event limit**: events filtered
- **detection.file searches**: fast pattern searches in file buffer
- **detection.hard evals**: non-fast pattern rule evaluations
- **detection.header searches**: fast pattern searches in header buffer
- **detection.key searches**: fast pattern searches in key buffer
- **detection.logged**: logged packets
- **detection.log limit**: events queued but not logged
- **detection.match limit**: fast pattern matches not processed
- **detection.passed**: passed packets
- **detection.pkt searches**: fast pattern searches in packet data
- **detection.queue limit**: events not queued because queue full
- **detection.raw searches**: fast pattern searches in raw packet data
- **detection.total alerts**: alerts including IP reputation
- **dnp3.dnp3 application pdus**: total dnp3 application pdus
- **dnp3.dnp3 link layer frames**: total dnp3 link layer frames
- **dnp3.tcp pdus**: total tcp pdus
- **dnp3.total packets**: total packets
- **dnp3.udp packets**: total udp packets
- **dns.packets**: total packets processed
- **dns.requests**: total dns requests
- **dns.responses**: total dns responses
- **file_connector.messages**: total messages
- **file_id.cache failures**: number of file cache add failures
- **file_id.total file data**: number of file data bytes processed

- **file_id.total files**: number of files processed
- **file_log.total events**: total file events
- **ftp_data.packets**: total packets
- **ftp_server.packets**: total packets
- **gtp_inspect.events**: requests
- **gtp_inspect.sessions**: total sessions processed
- **gtp_inspect.unknown infos**: unknown information elements
- **gtp_inspect.unknown types**: unknown message types
- **host_cache.lru cache adds**: lru cache added new entry
- **host_cache.lru cache clears**: lru cache clear API calls
- **host_cache.lru cache find hits**: lru cache found entry in cache
- **host_cache.lru cache find misses**: lru cache did not find entry in cache
- **host_cache.lru cache prunes**: lru cache pruned entry to make space for new entry
- **host_cache.lru cache removes**: lru cache found entry and removed it
- **host_cache.lru cache replaces**: lru cache replaced existing entry
- **host_tracker.service adds**: host service adds
- **host_tracker.service finds**: host service finds
- **host_tracker.service removes**: host service removes
- **http_inspect.chunked**: chunked message bodies
- **http_inspect.CONNECT requests**: CONNECT requests inspected
- **http_inspect.DELETE requests**: DELETE requests inspected
- **http_inspect.flows**: HTTP connections inspected
- **http_inspect.GET requests**: GET requests inspected
- **http_inspect.HEAD requests**: HEAD requests inspected
- **http_inspect.inspections**: total message sections inspected
- **http_inspect.OPTIONS requests**: OPTIONS requests inspected
- **http_inspect.other requests**: other request methods inspected
- **http_inspect.POST requests**: POST requests inspected
- **http_inspect.PUT requests**: PUT requests inspected
- **http_inspect.reassembles**: TCP segments combined into HTTP messages
- **http_inspect.request bodies**: POST, PUT, and other requests with message bodies
- **http_inspect.requests**: HTTP request messages inspected
- **http_inspect.responses**: HTTP response messages inspected
- **http_inspect.scans**: TCP segments scanned looking for HTTP messages
- **http_inspect.TRACE requests**: TRACE requests inspected

- **http_inspect.URI coding**: URIs with character coding problems
- **http_inspect.URI normalizations**: URIs needing to be normalization
- **http_inspect.URI path**: URIs with path problems
- **icmp4.bad checksum**: non-zero icmp checksums
- **icmp6.bad checksum (ip4)**: nonzero ipcm4 checksums
- **icmp6.bad checksum (ip6)**: nonzero ipcm6 checksums
- **imap.b64 attachments**: total base64 attachments decoded
- **imap.b64 decoded bytes**: total base64 decoded bytes
- **imap.non-encoded attachments**: total non-encoded attachments extracted
- **imap.non-encoded bytes**: total non-encoded extracted bytes
- **imap.packets**: total packets processed
- **imap.qp attachments**: total quoted-printable attachments decoded
- **imap.qp decoded bytes**: total quoted-printable decoded bytes
- **imap.sessions**: total imap sessions
- **imap.uu attachments**: total uu attachments decoded
- **imap.uu decoded bytes**: total uu decoded bytes
- **ipv4.bad checksum**: nonzero ip checksums
- **latency.max usecs**: maximum usecs elapsed
- **latency.packet timeouts**: packets that timed out
- **latency.rule eval timeouts**: rule evals that timed out
- **latency.rule tree enables**: rule tree re-enables
- **latency.total packets**: total packets monitored
- **latency.total rule evals**: total rule evals monitored
- **latency.total usecs**: total usecs elapsed
- **modbus.frames**: total Modbus messages
- **modbus.sessions**: total sessions processed
- **mpls.total bytes**: total mpls labeled bytes processed
- **mpls.total packets**: total mpls labeled packets processed
- **normalizer.icmp4 echo**: icmp4 ping normalizations
- **normalizer.icmp6 echo**: icmp6 echo normalizations
- **normalizer.ip4 df**: don’t frag bit normalizations
- **normalizer.ip4 opts**: ip4 options cleared
- **normalizer.ip4 rf**: reserved flag bit clears
- **normalizer.ip4 tos**: type of service normalizations
- **normalizer.ip4 trim**: eth packets trimmed to datagram size

- **normalizer.ip4 ttl**: time-to-live normalizations
- **normalizer.ip6 hops**: ip6 hop limit normalizations
- **normalizer.ip6 options**: ip6 options cleared
- **normalizer.tcp block**: blocked segments
- **normalizer.tcp ecn pkt**: packets with ECN bits cleared
- **normalizer.tcp ecn session**: ECN bits cleared
- **normalizer.tcp ips data**: normalized segments
- **normalizer.tcp nonce**: packets with nonce bit cleared
- **normalizer.tcp options**: packets with options cleared
- **normalizer.tcp paddding**: packets with padding cleared
- **normalizer.tcp req pay**: cleared urgent pointer and urgent flag when there is no payload
- **normalizer.tcp req urg**: cleared urgent pointer when urgent flag is not set
- **normalizer.tcp req urp**: cleared the urgent flag if the urgent pointer is not set
- **normalizer.tcp reserved**: packets with reserved bits cleared
- **normalizer.tcp syn options**: SYN only options cleared from non-SYN packets
- **normalizer.tcp trim mss**: data trimmed to MSS
- **normalizer.tcp trim rst**: RST packets with data trimmed
- **normalizer.tcp trim syn**: tcp segments trimmed on SYN
- **normalizer.tcp trim win**: data trimed to window
- **normalizer.tcp ts ecr**: timestamp cleared on non-ACKs
- **normalizer.tcp ts nop**: timestamp options cleared
- **normalizer.tcp urgent ptr**: packets without data with urgent pointer cleared
- **normalizer.test icmp4 echo**: test icmp4 ping normalizations
- **normalizer.test icmp6 echo**: test icmp6 echo normalizations
- **normalizer.test ip4 df**: test don’t frag bit normalizations
- **normalizer.test ip4 opts**: test ip4 options cleared
- **normalizer.test ip4 rf**: test reserved flag bit clears
- **normalizer.test ip4 tos**: test type of service normalizations
- **normalizer.test ip4 trim**: test eth packets trimmed to datagram size
- **normalizer.test ip4 ttl**: test time-to-live normalizations
- **normalizer.test ip6 hops**: test ip6 hop limit normalizations
- **normalizer.test ip6 options**: test ip6 options cleared
- **normalizer.test tcp block**: test blocked segments
- **normalizer.test tcp ecn pkt**: test packets with ECN bits cleared
- **normalizer.test tcp ecn session**: test ECN bits cleared

- **normalizer.test tcp ips data**: test normalized segments
- **normalizer.test tcp nonce**: test packets with nonce bit cleared
- **normalizer.test tcp options**: test packets with options cleared
- **normalizer.test tcp paddding**: test packets with padding cleared
- **normalizer.test tcp req pay**: test cleared urgent pointer and urgent flag when there is no payload
- **normalizer.test tcp req urg**: test cleared urgent pointer when urgent flag is not set
- **normalizer.test tcp req urp**: test cleared the urgent flag if the urgent pointer is not set
- **normalizer.test tcp reserved**: test packets with reserved bits cleared
- **normalizer.test tcp syn options**: test SYN only options cleared from non-SYN packets
- **normalizer.test tcp trim mss**: test data trimmed to MSS
- **normalizer.test tcp trim rst**: test RST packets with data trimmed
- **normalizer.test tcp trim syn**: test tcp segments trimmed on SYN
- **normalizer.test tcp trim win**: test data trimed to window
- **normalizer.test tcp ts ecr**: test timestamp cleared on non-ACKs
- **normalizer.test tcp ts nop**: test timestamp options cleared
- **normalizer.test tcp urgent ptr**: test packets without data with urgent pointer cleared
- **packet_capture.captured**: packets matching dumped after matching filter
- **packet_capture.processed**: packets processed against filter
- **perf_monitor.packets**: total packets
- **pop.b64 attachments**: total base64 attachments decoded
- **pop.b64 decoded bytes**: total base64 decoded bytes
- **pop.non-encoded attachments**: total non-encoded attachments extracted
- **pop.non-encoded bytes**: total non-encoded extracted bytes
- **pop.packets**: total packets processed
- **pop.qp attachments**: total quoted-printable attachments decoded
- **pop.qp decoded bytes**: total quoted-printable decoded bytes
- **pop.sessions**: total pop sessions
- **pop.uu attachments**: total uu attachments decoded
- **pop.uu decoded bytes**: total uu decoded bytes
- **port_scan_global.packets**: total packets
- **reputation.blacklisted**: number of packets blacklisted
- **reputation.memory allocated**: total memory allocated
- **reputation.monitored**: number of packets monitored
- **reputation.packets**: total packets processed
- **reputation.whitelisted**: number of packets whitelisted

- **rpc_decode.packets**: total packets
- **sd_pattern.below threshold**: sd_pattern matched but missed threshold
- **sd_pattern.pattern not found**: sd_pattern did not not match
- **sd_pattern.terminated**: hyperscan terminated
- **search_engine.max queued**: maximum fast pattern matches queued for further evaluation
- **search_engine.non-qualified events**: total non-qualified events
- **search_engine.qualified events**: total qualified events
- **search_engine.total flushed**: fast pattern matches discarded due to overflow
- **search_engine.total inserts**: total fast pattern hits
- **search_engine.total unique**: total unique fast pattern hits
- **sip.1xx**: 1xx
- **sip.2xx**: 2xx
- **sip.3xx**: 3xx
- **sip.4xx**: 4xx
- **sip.5xx**: 5xx
- **sip.6xx**: 6xx
- **sip.7xx**: 7xx
- **sip.8xx**: 8xx
- **sip.9xx**: 9xx
- **sip.ack**: ack
- **sip.bye**: bye
- **sip.cancel**: cancel
- **sip.dialogs**: total dialogs
- **sip.events**: events generated
- **sip.ignored channels**: total channels ignored
- **sip.ignored sessions**: total sessions ignored
- **sip.info**: info
- **sip.invite**: invite
- **sip.join**: join
- **sip.message**: message
- **sip.notify**: notify
- **sip.options**: options
- **sip.packets**: total packets
- **sip.prack**: prack
- **sip.refer**: refer

- **sip.register**: register
- **sip.sessions**: total sessions
- **sip.subscribe**: subscribe
- **sip.total requests**: total requests
- **sip.total responses**: total responses
- **sip.update**: update
- **smtp.b64 attachments**: total base64 attachments decoded
- **smtp.b64 decoded bytes**: total base64 decoded bytes
- **smtp.concurrent sessions**: total concurrent smtp sessions
- **smtp.max concurrent sessions**: maximum concurrent smtp sessions
- **smtp.non-encoded attachments**: total non-encoded attachments extracted
- **smtp.non-encoded bytes**: total non-encoded extracted bytes
- **smtp.packets**: total packets processed
- **smtp.qp attachments**: total quoted-printable attachments decoded
- **smtp.qp decoded bytes**: total quoted-printable decoded bytes
- **smtp.sessions**: total smtp sessions
- **smtp.uu attachments**: total uu attachments decoded
- **smtp.uu decoded bytes**: total uu decoded bytes
- **snort.attribute table hosts**: total number of hosts in table
- **snort.attribute table reloads**: number of times hosts table was reloaded
- **snort.conf reloads**: number of times configuration was reloaded
- **snort.local commands**: total local commands processed
- **snort.remote commands**: total remote commands processed
- **snort.signals**: total signals processed
- **ssh.packets**: total packets
- **ssl.alert**: total ssl alert records
- **ssl.bad handshakes**: total bad handshakes
- **ssl.certificate**: total ssl certificates
- **ssl.change cipher**: total change cipher records
- **ssl.client application**: total client application records
- **ssl.client hello**: total client hellos
- **ssl.client key exchange**: total client key exchanges
- **ssl.decoded**: ssl packets decoded
- **ssl.detection disabled**: total detection disabled
- **ssl.finished**: total handshakes finished

- **ssl.handshakes completed**: total completed ssl handshakes
- **ssl.packets**: total packets processed
- **ssl.server application**: total server application records
- **ssl.server done**: total server done
- **ssl.server hello**: total server hellos
- **ssl.server key exchange**: total server key exchanges
- **ssl.sessions ignored**: total sessions ignore
- **ssl.unrecognized records**: total unrecognized records
- **stream.file excess prunes**: file sessions pruned due to excess
- **stream.file flows**: total file sessions
- **stream.file ha prunes**: file sessions pruned by high availability sync
- **stream.file idle prunes**: file sessions pruned due to timeout
- **stream.file memcap prunes**: file sessions pruned due to memcap
- **stream.file preemptive prunes**: file sessions pruned during preemptive pruning
- **stream.file total prunes**: total file sessions pruned
- **stream.file uni prunes**: file uni sessions pruned
- **stream_icmp.created**: icmp session trackers created
- **stream.icmp excess prunes**: icmp sessions pruned due to excess
- **stream.icmp flows**: total icmp sessions
- **stream.icmp ha prunes**: icmp sessions pruned by high availability sync
- **stream.icmp idle prunes**: icmp sessions pruned due to timeout
- **stream_icmp.max**: max icmp sessions
- **stream.icmp memcap prunes**: icmp sessions pruned due to memcap
- **stream.icmp preemptive prunes**: icmp sessions pruned during preemptive pruning
- **stream_icmp.prunes**: icmp session prunes
- **stream_icmp.released**: icmp session trackers released
- **stream_icmp.sessions**: total icmp sessions
- **stream_icmp.timeouts**: icmp session timeouts
- **stream.icmp total prunes**: total icmp sessions pruned
- **stream.icmp uni prunes**: icmp uni sessions pruned
- **stream_ip.alerts**: alerts generated
- **stream_ip.anomalies**: anomalies detected
- **stream_ip.created**: ip session trackers created
- **stream_ip.current frags**: current fragments
- **stream_ip.discards**: fragments discarded

- **stream_ip.drops**: fragments dropped
- **stream.ip excess prunes**: ip sessions pruned due to excess
- **stream.ip flows**: total ip sessions
- **stream_ip.fragmented bytes**: total fragmented bytes
- **stream_ip.frag timeouts**: datagrams abandoned
- **stream.ip ha prunes**: ip sessions pruned by high availability sync
- **stream.ip idle prunes**: ip sessions pruned due to timeout
- **stream_ip.max frags**: max fragments
- **stream_ip.max**: max ip sessions
- **stream.ip memcap prunes**: ip sessions pruned due to memcap
- **stream_ip.memory used**: current memory usage in bytes
- **stream_ip.nodes deleted**: fragments deleted from tracker
- **stream_ip.nodes inserted**: fragments added to tracker
- **stream_ip.overlaps**: overlapping fragments
- **stream.ip preemptive prunes**: ip sessions pruned during preemptive pruning
- **stream_ip.prunes**: ip session prunes
- **stream_ip.reassembled bytes**: total reassembled bytes
- **stream_ip.reassembled**: reassembled datagrams
- **stream_ip.released**: ip session trackers released
- **stream_ip.sessions**: total ip sessions
- **stream_ip.timeouts**: ip session timeouts
- **stream_ip.total frags**: total fragments
- **stream.ip total prunes**: total ip sessions pruned
- **stream_ip.trackers added**: datagram trackers created
- **stream_ip.trackers cleared**: datagram trackers cleared
- **stream_ip.trackers completed**: datagram trackers completed
- **stream_ip.trackers freed**: datagram trackers released
- **stream.ip uni prunes**: ip uni sessions pruned
- **stream_tcp.3way trackers**: tcp session tracking started on ack
- **stream_tcp.client cleanups**: number of times data from server was flushed when session released
- **stream_tcp.closing**: number of sessions currently closing
- **stream_tcp.created**: tcp session trackers created
- **stream_tcp.data trackers**: tcp session tracking started on data
- **stream_tcp.discards**: tcp packets discarded
- **stream_tcp.established**: number of sessions currently established

- **stream_tcp.events**: events generated
- **stream.tcp excess prunes**: tcp sessions pruned due to excess
- **stream.tcp flows**: total tcp sessions
- **stream_tcp.gaps**: missing data between PDUs
- **stream.tcp ha prunes**: tcp sessions pruned by high availability sync
- **stream.tcp idle prunes**: tcp sessions pruned due to timeout
- **stream_tcp.ignored**: tcp packets ignored
- **stream_tcp.initializing**: number of sessions currently initializing
- **stream_tcp.internal events**: 135:X events generated
- **stream_tcp.max bytes**: number of times the maximum queued byte limit was reached
- **stream_tcp.max**: max tcp sessions
- **stream_tcp.max segs**: number of times the maximum queued segment limit was reached
- **stream.tcp memcap prunes**: tcp sessions pruned due to memcap
- **stream_tcp.memory**: current memory in use
- **stream_tcp.overlaps**: overlapping segments queued
- **stream.tcp preemptive prunes**: tcp sessions pruned during preemptive pruning
- **stream_tcp.prunes**: tcp session prunes
- **stream_tcp.rebuilt buffers**: rebuilt PDU sections
- **stream_tcp.rebuilt bytes**: total rebuilt bytes
- **stream_tcp.rebuilt packets**: total reassembled PDUs
- **stream_tcp.released**: tcp session trackers released
- **stream_tcp.resyns**: SYN received on established session
- **stream_tcp.segs queued**: total segments queued
- **stream_tcp.segs released**: total segments released
- **stream_tcp.segs split**: tcp segments split when reassembling PDUs
- **stream_tcp.segs used**: queued tcp segments applied to reassembled PDUs
- **stream_tcp.server cleanups**: number of times data from client was flushed when session released
- **stream_tcp.sessions**: total tcp sessions
- **stream_tcp.syn-ack trackers**: tcp session tracking started on syn-ack
- **stream_tcp.syn trackers**: tcp session tracking started on syn
- **stream_tcp.timeouts**: tcp session timeouts
- **stream.tcp total prunes**: total tcp sessions pruned
- **stream.tcp uni prunes**: tcp uni sessions pruned
- **stream_tcp.untracked**: tcp packets not tracked
- **stream_udp.created**: udp session trackers created

- **stream.udp excess prunes**: udp sessions pruned due to excess
- **stream.udp flows**: total udp sessions
- **stream.udp ha prunes**: udp sessions pruned by high availability sync
- **stream.udp idle prunes**: udp sessions pruned due to timeout
- **stream_udp.max**: max udp sessions
- **stream.udp memcap prunes**: udp sessions pruned due to memcap
- **stream.udp preemptive prunes**: udp sessions pruned during preemptive pruning
- **stream_udp.prunes**: udp session prunes
- **stream_udp.released**: udp session trackers released
- **stream_udp.sessions**: total udp sessions
- **stream_udp.timeouts**: udp session timeouts
- **stream.udp total prunes**: total udp sessions pruned
- **stream.udp uni prunes**: udp uni sessions pruned
- **stream.user excess prunes**: user sessions pruned due to excess
- **stream.user flows**: total user sessions
- **stream.user ha prunes**: user sessions pruned by high availability sync
- **stream.user idle prunes**: user sessions pruned due to timeout
- **stream.user memcap prunes**: user sessions pruned due to memcap
- **stream.user preemptive prunes**: user sessions pruned during preemptive pruning
- **stream.user total prunes**: total user sessions pruned
- **stream.user uni prunes**: user uni sessions pruned
- **tcp.bad checksum (ip4)**: nonzero tcp over ip checksums
- **tcp.bad checksum (ip6)**: nonzero tcp over ipv6 checksums
- **tcp_connector.messages**: total messages
- **telnet.packets**: total packets
- **udp.bad checksum (ip4)**: nonzero udp over ipv4 checksums
- **udp.bad checksum (ip6)**: nonzero udp over ipv6 checksums
- **wizard.tcp hits**: tcp identifications
- **wizard.tcp scans**: tcp payload scans
- **wizard.udp hits**: udp identifications
- **wizard.udp scans**: udp payload scans
- **wizard.user hits**: user identifications
- **wizard.user scans**: user payload scans

19.6 Generators

- **105**: back_orifice
- **106**: rpc_decode
- **112**: arp_spoof
- **116**: arp
- **116**: auth
- **116**: ciscometadata
- **116**: decode
- **116**: erspan2
- **116**: erspan3
- **116**: esp
- **116**: eth
- **116**: fabricpath
- **116**: gre
- **116**: gtp
- **116**: icmp4
- **116**: icmp6
- **116**: igmp
- **116**: ipv4
- **116**: ipv6
- **116**: mpls
- **116**: pgm
- **116**: pppoe
- **116**: tcp
- **116**: udp
- **116**: vlan
- **119**: http_inspect
- **122**: port_scan
- **123**: stream_ip
- **124**: smtp
- **125**: ftp_server
- **126**: telnet
- **128**: ssh
- **129**: stream_tcp
- **131**: dns

- **133**: dce_smb
- **133**: dce_tcp
- **133**: dce_udp
- **134**: latency
- **136**: reputation
- **137**: ssl
- **140**: sip
- **141**: imap
- **142**: pop
- **143**: gtp_inspect
- **144**: modbus
- **145**: dnp3

19.7 Builtin Rules

- **105:1** (back_orifice) BO traffic detected
- **105:2** (back_orifice) BO client traffic detected
- **105:3** (back_orifice) BO server traffic detected
- **105:4** (back_orifice) BO Snort buffer attack
- **106:1** (rpc_decode) fragmented RPC records
- **106:2** (rpc_decode) multiple RPC records
- **106:3** (rpc_decode) large RPC record fragment
- **106:4** (rpc_decode) incomplete RPC segment
- **106:5** (rpc_decode) zero-length RPC fragment
- **112:1** (arp_spoof) unicast ARP request
- **112:2** (arp_spoof) ethernet/ARP mismatch request for source
- **112:3** (arp_spoof) ethernet/ARP mismatch request for destination
- **112:4** (arp_spoof) attempted ARP cache overwrite attack
- **116:1** (ipv4) not IPv4 datagram
- **116:2** (ipv4) IPv4 header length < minimum
- **116:3** (ipv4) IPv4 datagram length < header field
- **116:4** (ipv4) IPv4 options found with bad lengths
- **116:5** (ipv4) truncated IPv4 options
- **116:6** (ipv4) IPv4 datagram length > captured length
- **116:45** (tcp) TCP packet length is smaller than 20 bytes
- **116:46** (tcp) TCP data offset is less than 5

- **116:47** (tcp) TCP header length exceeds packet length
- **116:54** (tcp) TCP options found with bad lengths
- **116:55** (tcp) truncated TCP options
- **116:56** (tcp) T/TCP detected
- **116:57** (tcp) obsolete TCP options found
- **116:58** (tcp) experimental TCP options found
- **116:59** (tcp) TCP window scale option found with length > 14
- **116:95** (udp) truncated UDP header
- **116:96** (udp) invalid UDP header, length field < 8
- **116:97** (udp) short UDP packet, length field > payload length
- **116:98** (udp) long UDP packet, length field < payload length
- **116:105** (icmp4) ICMP header truncated
- **116:106** (icmp4) ICMP timestamp header truncated
- **116:107** (icmp4) ICMP address header truncated
- **116:109** (arp) truncated ARP
- **116:120** (pppoe) bad PPPOE frame detected
- **116:130** (vlan) bad VLAN frame
- **116:131** (vlan) bad LLC header
- **116:132** (vlan) bad extra LLC info
- **116:150** (decode) loopback IP
- **116:151** (decode) same src/dst IP
- **116:160** (gre) GRE header length > payload length
- **116:161** (gre) multiple encapsulations in packet
- **116:162** (gre) invalid GRE version
- **116:163** (gre) invalid GRE header
- **116:164** (gre) invalid GRE v.1 PPTP header
- **116:165** (gre) GRE trans header length > payload length
- **116:170** (mpls) bad MPLS frame
- **116:171** (mpls) MPLS label 0 appears in non-bottom header
- **116:172** (mpls) MPLS label 1 appears in bottom header
- **116:173** (mpls) MPLS label 2 appears in non-bottom header
- **116:174** (mpls) MPLS label 3 appears in header
- **116:175** (mpls) MPLS label 4, 5,.. or 15 appears in header
- **116:176** (mpls) too many MPLS headers
- **116:250** (icmp4) ICMP original IP header truncated

- **116:251** (icmp4) ICMP version and original IP header versions differ
- **116:252** (icmp4) ICMP original datagram length < original IP header length
- **116:253** (icmp4) ICMP original IP payload < 64 bits
- **116:254** (icmp4) ICMP original IP payload > 576 bytes
- **116:255** (icmp4) ICMP original IP fragmented and offset not 0
- **116:270** (ipv6) IPv6 packet below TTL limit
- **116:271** (ipv6) IPv6 header claims to not be IPv6
- **116:272** (ipv6) IPv6 truncated extension header
- **116:273** (ipv6) IPv6 truncated header
- **116:274** (ipv6) IPv6 datagram length < header field
- **116:275** (ipv6) IPv6 datagram length > captured length
- **116:276** (ipv6) IPv6 packet with destination address ::0
- **116:277** (ipv6) IPv6 packet with multicast source address
- **116:278** (ipv6) IPv6 packet with reserved multicast destination address
- **116:279** (ipv6) IPv6 header includes an undefined option type
- **116:280** (ipv6) IPv6 address includes an unassigned multicast scope value
- **116:281** (ipv6) IPv6 header includes an invalid value for the *next header* field
- **116:282** (ipv6) IPv6 header includes a routing extension header followed by a hop-by-hop header
- **116:283** (ipv6) IPv6 header includes two routing extension headers
- **116:285** (icmp6) ICMPv6 packet of type 2 (message too big) with MTU field < 1280
- **116:286** (icmp6) ICMPv6 packet of type 1 (destination unreachable) with non-RFC 2463 code
- **116:287** (icmp6) ICMPv6 router solicitation packet with a code not equal to 0
- **116:288** (icmp6) ICMPv6 router advertisement packet with a code not equal to 0
- **116:289** (icmp6) ICMPv6 router solicitation packet with the reserved field not equal to 0
- **116:290** (icmp6) ICMPv6 router advertisement packet with the reachable time field set > 1 hour
- **116:291** (ipv6) IPV6 tunneled over IPv4, IPv6 header truncated, possible Linux kernel attack
- **116:292** (ipv6) IPv6 header has destination options followed by a routing header
- **116:293** (decode) two or more IP (v4 and/or v6) encapsulation layers present
- **116:294** (esp) truncated encapsulated security payload header
- **116:295** (ipv6) IPv6 header includes an option which is too big for the containing header
- **116:296** (ipv6) IPv6 packet includes out-of-order extension headers
- **116:297** (gtp) two or more GTP encapsulation layers present
- **116:298** (gtp) GTP header length is invalid
- **116:400** (tcp) XMAS attack detected
- **116:401** (tcp) Nmap XMAS attack detected

- **116:402** (tcp) DOS NAPTHA vulnerability detected
- **116:403** (tcp) SYN to multicast address
- **116:404** (ipv4) IPv4 packet with zero TTL
- **116:405** (ipv4) IPv4 packet with bad frag bits (both MF and DF set)
- **116:406** (udp) invalid IPv6 UDP packet, checksum zero
- **116:407** (ipv4) IPv4 packet frag offset + length exceed maximum
- **116:408** (ipv4) IPv4 packet from *current net* source address
- **116:409** (ipv4) IPv4 packet to *current net* dest address
- **116:410** (ipv4) IPv4 packet from multicast source address
- **116:411** (ipv4) IPv4 packet from reserved source address
- **116:412** (ipv4) IPv4 packet to reserved dest address
- **116:413** (ipv4) IPv4 packet from broadcast source address
- **116:414** (ipv4) IPv4 packet to broadcast dest address
- **116:415** (icmp4) ICMP4 packet to multicast dest address
- **116:416** (icmp4) ICMP4 packet to broadcast dest address
- **116:418** (icmp4) ICMP4 type other
- **116:419** (tcp) TCP urgent pointer exceeds payload length or no payload
- **116:420** (tcp) TCP SYN with FIN
- **116:421** (tcp) TCP SYN with RST
- **116:422** (tcp) TCP PDU missing ack for established session
- **116:423** (tcp) TCP has no SYN, ACK, or RST
- **116:424** (eth) truncated ethernet header
- **116:425** (ipv4) truncated IPv4 header
- **116:426** (icmp4) truncated ICMP4 header
- **116:427** (icmp6) truncated ICMP6 header
- **116:428** (ipv4) IPv4 packet below TTL limit
- **116:429** (ipv6) IPv6 packet has zero hop limit
- **116:430** (ipv4) IPv4 packet both DF and offset set
- **116:431** (icmp6) ICMPv6 type not decoded
- **116:432** (icmp6) ICMPv6 packet to multicast address
- **116:433** (tcp) DDOS shaft SYN flood
- **116:434** (icmp4) ICMP ping Nmap
- **116:435** (icmp4) ICMP icmpenum v1.1.1
- **116:436** (icmp4) ICMP redirect host
- **116:437** (icmp4) ICMP redirect net

- **116:438** (icmp4) ICMP traceroute ipopts
- **116:439** (icmp4) ICMP source quench
- **116:440** (icmp4) broadscan smurf scanner
- **116:441** (icmp4) ICMP destination unreachable communication administratively prohibited
- **116:442** (icmp4) ICMP destination unreachable communication with destination host is administratively prohibited
- **116:443** (icmp4) ICMP destination unreachable communication with destination network is administratively prohibited
- **116:444** (ipv4) IPv4 option set
- **116:445** (udp) large UDP packet (> 4000 bytes)
- **116:446** (tcp) TCP port 0 traffic
- **116:447** (udp) UDP port 0 traffic
- **116:448** (ipv4) IPv4 reserved bit set
- **116:449** (decode) unassigned/reserved IP protocol
- **116:450** (decode) bad IP protocol
- **116:451** (icmp4) ICMP path MTU denial of service attempt
- **116:452** (icmp4) Linux ICMP header DOS attempt
- **116:453** (ipv6) ISATAP-addressed IPv6 traffic spoofing attempt
- **116:454** (pgm) PGM nak list overflow attempt
- **116:455** (igmp) DOS IGMP IP options validation attempt
- **116:456** (ipv6) too many IPv6 extension headers
- **116:457** (icmp6) ICMPv6 packet of type 1 (destination unreachable) with non-RFC 4443 code
- **116:458** (ipv6) bogus fragmentation packet, possible BSD attack
- **116:459** (decode) fragment with zero length
- **116:460** (icmp6) ICMPv6 node info query/response packet with a code greater than 2
- **116:461** (ipv6) IPv6 routing type 0 extension header
- **116:462** (erspan2) ERSpan header version mismatch
- **116:463** (erspan2) captured length < ERSpan type2 header length
- **116:464** (erspan3) captured < ERSpan type3 header length
- **116:465** (auth) truncated authentication header
- **116:466** (auth) bad authentication header length
- **116:467** (fabricpath) truncated FabricPath header
- **116:468** (ciscometadata) truncated Cisco Metadata header
- **116:469** (ciscometadata) invalid Cisco Metadata option length
- **116:470** (ciscometadata) invalid Cisco Metadata option type
- **116:471** (ciscometadata) invalid Cisco Metadata SGT
- **116:472** (decode) too many protocols present

- **119:1** (http_inspect) ascii encoding
- **119:2** (http_inspect) double decoding attack
- **119:3** (http_inspect) u encoding
- **119:4** (http_inspect) bare byte unicode encoding
- **119:5** (http_inspect) obsolete event—should not appear
- **119:6** (http_inspect) UTF-8 encoding
- **119:7** (http_inspect) IIS unicode codepoint encoding
- **119:8** (http_inspect) multi_slash encoding
- **119:9** (http_inspect) IIS backslash evasion
- **119:10** (http_inspect) self directory traversal
- **119:11** (http_inspect) directory traversal
- **119:12** (http_inspect) apache whitespace (tab)
- **119:13** (http_inspect) non-RFC http delimiter
- **119:14** (http_inspect) non-RFC defined char
- **119:15** (http_inspect) oversize request-uri directory
- **119:16** (http_inspect) oversize chunk encoding
- **119:17** (http_inspect) unauthorized proxy use detected
- **119:18** (http_inspect) webroot directory traversal
- **119:19** (http_inspect) long header
- **119:20** (http_inspect) max header fields
- **119:21** (http_inspect) multiple content length
- **119:22** (http_inspect) chunk size mismatch detected
- **119:23** (http_inspect) invalid IP in true-client-IP/XFF header
- **119:24** (http_inspect) multiple host hdrs detected
- **119:25** (http_inspect) hostname exceeds 255 characters
- **119:26** (http_inspect) header parsing space saturation
- **119:27** (http_inspect) client consecutive small chunk sizes
- **119:28** (http_inspect) post w/o content-length or chunks
- **119:29** (http_inspect) multiple true ips in a session
- **119:30** (http_inspect) both true-client-IP and XFF hdrs present
- **119:31** (http_inspect) unknown method
- **119:32** (http_inspect) simple request
- **119:33** (http_inspect) unescaped space in HTTP URI
- **119:34** (http_inspect) too many pipelined requests
- **119:35** (http_inspect) anomalous http server on undefined HTTP port

- **119:36** (http_inspect) invalid status code in HTTP response
- **119:37** (http_inspect) no content-length or transfer-encoding in HTTP response
- **119:38** (http_inspect) HTTP response has UTF charset which failed to normalize
- **119:39** (http_inspect) HTTP response has UTF-7 charset
- **119:40** (http_inspect) HTTP response gzip decompression failed
- **119:41** (http_inspect) server consecutive small chunk sizes
- **119:42** (http_inspect) invalid content-length or chunk size
- **119:43** (http_inspect) javascript obfuscation levels exceeds 1
- **119:44** (http_inspect) javascript whitespaces exceeds max allowed
- **119:45** (http_inspect) multiple encodings within javascript obfuscated data
- **119:46** (http_inspect) SWF file zlib decompression failure
- **119:47** (http_inspect) SWF file LZMA decompression failure
- **119:48** (http_inspect) PDF file deflate decompression failure
- **119:49** (http_inspect) PDF file unsupported compression type
- **119:50** (http_inspect) PDF file cascaded compression
- **119:51** (http_inspect) PDF file parse failure
- **119:52** (http_inspect) not HTTP traffic
- **119:53** (http_inspect) chunk length has excessive leading zeros
- **119:54** (http_inspect) white space before or between messages
- **119:55** (http_inspect) request message without URI
- **119:56** (http_inspect) control character in reason phrase
- **119:57** (http_inspect) illegal extra whitespace in start line
- **119:58** (http_inspect) corrupted HTTP version
- **119:59** (http_inspect) unknown HTTP version
- **119:60** (http_inspect) format error in HTTP header
- **119:61** (http_inspect) chunk header options present
- **119:62** (http_inspect) URI badly formatted
- **119:63** (http_inspect) unrecognized type of percent encoding in URI
- **119:64** (http_inspect) HTTP chunk misformatted
- **119:65** (http_inspect) white space following chunk length
- **119:66** (http_inspect) white space within header name
- **119:67** (http_inspect) excessive gzip compression
- **119:68** (http_inspect) gzip decompression failed
- **119:69** (http_inspect) HTTP 0.9 requested followed by another request
- **119:70** (http_inspect) HTTP 0.9 request following a normal request

- **119:71** (http_inspect) message has both Content-Length and Transfer-Encoding
- **119:72** (http_inspect) status code implying no body combined with Transfer-Encoding or nonzero Content-Length
- **119:73** (http_inspect) Transfer-Encoding did not end with chunked
- **119:74** (http_inspect) Transfer-Encoding with chunked not at end
- **119:75** (http_inspect) misformatted HTTP traffic
- **119:76** (http_inspect) unsupported Transfer-Encoding or Content-Encoding used
- **119:77** (http_inspect) unknown Transfer-Encoding or Content-Encoding used
- **119:78** (http_inspect) multiple layers of compression encodings applied
- **122:1** (port_scan) TCP portscan
- **122:2** (port_scan) TCP decoy portscan
- **122:3** (port_scan) TCP portsweep
- **122:4** (port_scan) TCP distributed portscan
- **122:5** (port_scan) TCP filtered portscan
- **122:6** (port_scan) TCP filtered decoy portscan
- **122:7** (port_scan) TCP filtered portsweep
- **122:8** (port_scan) TCP filtered distributed portscan
- **122:9** (port_scan) IP protocol scan
- **122:10** (port_scan) IP decoy protocol scan
- **122:11** (port_scan) IP protocol sweep
- **122:12** (port_scan) IP distributed protocol scan
- **122:13** (port_scan) IP filtered protocol scan
- **122:14** (port_scan) IP filtered decoy protocol scan
- **122:15** (port_scan) IP filtered protocol sweep
- **122:16** (port_scan) IP filtered distributed protocol scan
- **122:17** (port_scan) UDP portscan
- **122:18** (port_scan) UDP decoy portscan
- **122:19** (port_scan) UDP portsweep
- **122:20** (port_scan) UDP distributed portscan
- **122:21** (port_scan) UDP filtered portscan
- **122:22** (port_scan) UDP filtered decoy portscan
- **122:23** (port_scan) UDP filtered portsweep
- **122:24** (port_scan) UDP filtered distributed portscan
- **122:25** (port_scan) ICMP sweep
- **122:26** (port_scan) ICMP filtered sweep
- **122:27** (port_scan) open port

- **123:1** (stream_ip) inconsistent IP options on fragmented packets
- **123:2** (stream_ip) teardrop attack
- **123:3** (stream_ip) short fragment, possible DOS attempt
- **123:4** (stream_ip) fragment packet ends after defragmented packet
- **123:5** (stream_ip) zero-byte fragment packet
- **123:6** (stream_ip) bad fragment size, packet size is negative
- **123:7** (stream_ip) bad fragment size, packet size is greater than 65536
- **123:8** (stream_ip) fragmentation overlap
- **123:11** (stream_ip) TTL value less than configured minimum, not using for reassembly
- **123:12** (stream_ip) excessive fragment overlap
- **123:13** (stream_ip) tiny fragment
- **124:1** (smtp) attempted command buffer overflow
- **124:2** (smtp) attempted data header buffer overflow
- **124:3** (smtp) attempted response buffer overflow
- **124:4** (smtp) attempted specific command buffer overflow
- **124:5** (smtp) unknown command
- **124:6** (smtp) illegal command
- **124:7** (smtp) attempted header name buffer overflow
- **124:8** (smtp) attempted X-Link2State command buffer overflow
- **124:10** (smtp) base64 decoding failed
- **124:11** (smtp) quoted-printable decoding failed
- **124:13** (smtp) Unix-to-Unix decoding failed
- **124:14** (smtp) Cyrus SASL authentication attack
- **124:15** (smtp) attempted authentication command buffer overflow
- **125:1** (ftp_server) TELNET cmd on FTP command channel
- **125:2** (ftp_server) invalid FTP command
- **125:3** (ftp_server) FTP command parameters were too long
- **125:4** (ftp_server) FTP command parameters were malformed
- **125:5** (ftp_server) FTP command parameters contained potential string format
- **125:6** (ftp_server) FTP response message was too long
- **125:7** (ftp_server) FTP traffic encrypted
- **125:8** (ftp_server) FTP bounce attempt
- **125:9** (ftp_server) evasive (incomplete) TELNET cmd on FTP command channel
- **126:1** (telnet) consecutive telnet AYT commands beyond threshold
- **126:2** (telnet) telnet traffic encrypted

- **126:3** (telnet) telnet subnegotiation begin command without subnegotiation end
- **128:1** (ssh) challenge-response overflow exploit
- **128:2** (ssh) SSH1 CRC32 exploit
- **128:3** (ssh) server version string overflow
- **128:5** (ssh) bad message direction
- **128:6** (ssh) payload size incorrect for the given payload
- **128:7** (ssh) failed to detect SSH version string
- **129:1** (stream_tcp) SYN on established session
- **129:2** (stream_tcp) data on SYN packet
- **129:3** (stream_tcp) data sent on stream not accepting data
- **129:4** (stream_tcp) TCP timestamp is outside of PAWS window
- **129:5** (stream_tcp) bad segment, adjusted size ⇐ 0 (deprecated)
- **129:6** (stream_tcp) window size (after scaling) larger than policy allows
- **129:7** (stream_tcp) limit on number of overlapping TCP packets reached
- **129:8** (stream_tcp) data sent on stream after TCP reset sent
- **129:9** (stream_tcp) TCP client possibly hijacked, different ethernet address
- **129:10** (stream_tcp) TCP server possibly hijacked, different ethernet address
- **129:11** (stream_tcp) TCP data with no TCP flags set
- **129:12** (stream_tcp) consecutive TCP small segments exceeding threshold
- **129:13** (stream_tcp) 4-way handshake detected
- **129:14** (stream_tcp) TCP timestamp is missing
- **129:15** (stream_tcp) reset outside window
- **129:16** (stream_tcp) FIN number is greater than prior FIN
- **129:17** (stream_tcp) ACK number is greater than prior FIN
- **129:18** (stream_tcp) data sent on stream after TCP reset received
- **129:19** (stream_tcp) TCP window closed before receiving data
- **129:20** (stream_tcp) TCP session without 3-way handshake
- **131:1** (dns) obsolete DNS RR types
- **131:2** (dns) experimental DNS RR types
- **131:3** (dns) DNS client rdata txt overflow
- **133:2** (dce_smb) SMB - bad NetBIOS session service session type
- **133:3** (dce_smb) SMB - bad SMB message type
- **133:4** (dce_smb) SMB - bad SMB Id (not \xffSMB for SMB1 or not \xfeSMB for SMB2)
- **133:5** (dce_smb) SMB - bad word count or structure size
- **133:6** (dce_smb) SMB - bad byte count

- **133:7** (dce_smb) SMB - bad format type
- **133:8** (dce_smb) SMB - bad offset
- **133:9** (dce_smb) SMB - zero total data count
- **133:10** (dce_smb) SMB - NetBIOS data length less than SMB header length
- **133:12** (dce_smb) SMB - remaining NetBIOS data length less than command byte count
- **133:13** (dce_smb) SMB - remaining NetBIOS data length less than command data size
- **133:14** (dce_smb) SMB - remaining total data count less than this command data size
- **133:15** (dce_smb) SMB - total data sent (STDu64) greater than command total data expected
- **133:16** (dce_smb) SMB - byte count less than command data size (STDu64)
- **133:17** (dce_smb) SMB - invalid command data size for byte count
- **133:18** (dce_smb) SMB - excessive tree connect requests with pending tree connect responses
- **133:19** (dce_smb) SMB - excessive read requests with pending read responses
- **133:20** (dce_smb) SMB - excessive command chaining
- **133:21** (dce_smb) SMB - multiple chained tree connect requests
- **133:22** (dce_smb) SMB - multiple chained tree connect requests
- **133:23** (dce_smb) SMB - chained/compounded login followed by logoff
- **133:24** (dce_smb) SMB - chained/compounded tree connect followed by tree disconnect
- **133:25** (dce_smb) SMB - chained/compounded open pipe followed by close pipe
- **133:26** (dce_smb) SMB - invalid share access
- **133:27** (dce_smb) connection oriented DCE/RPC - invalid major version
- **133:27** (dce_tcp) connection oriented DCE/RPC - invalid major version
- **133:28** (dce_smb) connection oriented DCE/RPC - invalid minor version
- **133:28** (dce_tcp) connection oriented DCE/RPC - invalid minor version
- **133:29** (dce_smb) connection-oriented DCE/RPC - invalid PDU type
- **133:29** (dce_tcp) connection-oriented DCE/RPC - invalid PDU type
- **133:30** (dce_smb) connection-oriented DCE/RPC - fragment length less than header size
- **133:30** (dce_tcp) connection-oriented DCE/RPC - fragment length less than header size
- **133:32** (dce_smb) connection-oriented DCE/RPC - no context items specified
- **133:32** (dce_tcp) connection-oriented DCE/RPC - no context items specified
- **133:33** (dce_smb) connection-oriented DCE/RPC -no transfer syntaxes specified
- **133:33** (dce_tcp) connection-oriented DCE/RPC -no transfer syntaxes specified
- **133:34** (dce_smb) connection-oriented DCE/RPC - fragment length on non-last fragment less than maximum negotiated fragment transmit size for client
- **133:34** (dce_tcp) connection-oriented DCE/RPC - fragment length on non-last fragment less than maximum negotiated fragment transmit size for client
- **133:35** (dce_smb) connection-oriented DCE/RPC - fragment length greater than maximum negotiated fragment transmit size

- **133:35** (dce_tcp) connection-oriented DCE/RPC - fragment length greater than maximum negotiated fragment transmit size
- **133:36** (dce_smb) connection-oriented DCE/RPC - alter context byte order different from bind
- **133:36** (dce_tcp) connection-oriented DCE/RPC - alter context byte order different from bind
- **133:37** (dce_smb) connection-oriented DCE/RPC - call id of non first/last fragment different from call id established for fragmented request
- **133:37** (dce_tcp) connection-oriented DCE/RPC - call id of non first/last fragment different from call id established for fragmented request
- **133:38** (dce_smb) connection-oriented DCE/RPC - opnum of non first/last fragment different from opnum established for fragmented request
- **133:38** (dce_tcp) connection-oriented DCE/RPC - opnum of non first/last fragment different from opnum established for fragmented request
- **133:39** (dce_smb) connection-oriented DCE/RPC - context id of non first/last fragment different from context id established for fragmented request
- **133:39** (dce_tcp) connection-oriented DCE/RPC - context id of non first/last fragment different from context id established for fragmented request
- **133:40** (dce_udp) connection-less DCE/RPC - invalid major version
- **133:41** (dce_udp) connection-less DCE/RPC - invalid PDU type
- **133:42** (dce_udp) connection-less DCE/RPC - data length less than header size
- **133:43** (dce_udp) connection-less DCE/RPC - bad sequence number
- **133:44** (dce_smb) SMB - invalid SMB version 1 seen
- **133:45** (dce_smb) SMB - invalid SMB version 2 seen
- **133:46** (dce_smb) SMB - invalid user, tree connect, file binding
- **133:47** (dce_smb) SMB - excessive command compounding
- **133:48** (dce_smb) SMB - zero data count
- **133:50** (dce_smb) SMB - maximum number of outstanding requests exceeded
- **133:51** (dce_smb) SMB - outstanding requests with same MID
- **133:52** (dce_smb) SMB - deprecated dialect negotiated
- **133:53** (dce_smb) SMB - deprecated command used
- **133:54** (dce_smb) SMB - unusual command used
- **133:55** (dce_smb) SMB - invalid setup count for command
- **133:56** (dce_smb) SMB - client attempted multiple dialect negotiations on session
- **133:57** (dce_smb) SMB - client attempted to create or set a file’s attributes to readonly/hidden/system
- **133:58** (dce_smb) SMB - file offset provided is greater than file size specified
- **133:59** (dce_smb) SMB - next command specified in SMB2 header is beyond payload boundary
- **134:1** (latency) rule tree suspended due to latency
- **134:2** (latency) rule tree re-enabled after suspend timeout
- **134:3** (latency) packet fastpathed due to latency

- **136:1** (reputation) packets blacklisted
- **136:2** (reputation) packets whitelisted
- **136:3** (reputation) packets monitored
- **137:1** (ssl) invalid client HELLO after server HELLO detected
- **137:2** (ssl) invalid server HELLO without client HELLO detected
- **137:3** (ssl) heartbeat read overrun attempt detected
- **137:4** (ssl) large heartbeat response detected
- **140:1** (sip) maximum sessions reached
- **140:2** (sip) empty request URI
- **140:3** (sip) URI is too long
- **140:4** (sip) empty call-Id
- **140:5** (sip) Call-Id is too long
- **140:6** (sip) CSeq number is too large or negative
- **140:7** (sip) request name in CSeq is too long
- **140:8** (sip) empty From header
- **140:9** (sip) From header is too long
- **140:10** (sip) empty To header
- **140:11** (sip) To header is too long
- **140:12** (sip) empty Via header
- **140:13** (sip) Via header is too long
- **140:14** (sip) empty Contact
- **140:15** (sip) contact is too long
- **140:16** (sip) content length is too large or negative
- **140:17** (sip) multiple SIP messages in a packet
- **140:18** (sip) content length mismatch
- **140:19** (sip) request name is invalid
- **140:20** (sip) Invite replay attack
- **140:21** (sip) illegal session information modification
- **140:22** (sip) response status code is not a 3 digit number
- **140:23** (sip) empty Content-type header
- **140:24** (sip) SIP version is invalid
- **140:25** (sip) mismatch in METHOD of request and the CSEQ header
- **140:26** (sip) method is unknown
- **140:27** (sip) maximum dialogs within a session reached
- **141:1** (imap) unknown IMAP3 command

- **141:2** (imap) unknown IMAP3 response
- **141:4** (imap) base64 decoding failed
- **141:5** (imap) quoted-printable decoding failed
- **141:7** (imap) Unix-to-Unix decoding failed
- **142:1** (pop) unknown POP3 command
- **142:2** (pop) unknown POP3 response
- **142:4** (pop) base64 decoding failed
- **142:5** (pop) quoted-printable decoding failed
- **142:7** (pop) Unix-to-Unix decoding failed
- **143:1** (gtp_inspect) message length is invalid
- **143:2** (gtp_inspect) information element length is invalid
- **143:3** (gtp_inspect) information elements are out of order
- **144:1** (modbus) length in Modbus MBAP header does not match the length needed for the given function
- **144:2** (modbus) Modbus protocol ID is non-zero
- **144:3** (modbus) reserved Modbus function code in use
- **145:1** (dnp3) DNP3 link-layer frame contains bad CRC
- **145:2** (dnp3) DNP3 link-layer frame was dropped
- **145:3** (dnp3) DNP3 transport-layer segment was dropped during reassembly
- **145:4** (dnp3) DNP3 reassembly buffer was cleared without reassembling a complete message
- **145:5** (dnp3) DNP3 link-layer frame uses a reserved address
- **145:6** (dnp3) DNP3 application-layer fragment uses a reserved function code

19.8 Command Set

- **packet_capture.disable**(): stop packet dump
- **packet_capture.enable**(filter): dump raw packets
- **snort.detach**(): exit shell w/o shutdown
- **snort.dump_stats**(): show summary statistics
- **snort.help**(): this output
- **snort.pause**(): suspend packet processing
- **snort.quit**(): shutdown and dump-stats
- **snort.reload_config**(filename): load new configuration
- **snort.reload_hosts**(filename): load a new hosts table
- **snort.resume**(): continue packet processing
- **snort.rotate_stats**(): roll perfmonitor log files
- **snort.show_plugins**(): show available plugins

19.9 Signals

Important
Signal numbers are for the system that generated this documentation and are not applicable elsewhere.

- **hosts**(23): reload hosts file
- **int**(2): shutdown normally
- **quit**(3): shutdown as if started with --dirty-pig
- **reload**(1): reload config file
- **rotate**(12): rotate stats files
- **stats**(10): dump stats to stdout
- **term**(15): shutdown normally

19.10 Configuration Changes

```
change ->  dynamicdetection ==> ’snort.--plugin_path=<path>’
change ->  dynamicengine ==> ’snort.--plugin_path=<path>’
change ->  dynamicpreprocessor ==> ’snort.--plugin_path=<path>’
change ->  dynamicsidechannel ==> ’snort.--plugin_path=<path>’
change -> alertfile: ’config alertfile:’ ==> ’alert_fast.file’
change -> alertfile: ’config alertfile:’ ==> ’alert_full.file’
change -> attribute_table: ’STREAM_POLICY’ ==> ’hosts: tcp_policy’
change -> attribute_table: ’filename <file_name>’ ==> ’hosts[]’
change -> config ’ addressspace_agnostic’  ==> ’ packets. address_space_agnostic’
change -> config ’ checksum_mode’  ==> ’ network. checksum_eval’
change -> config ’ daq’  ==> ’ daq. type’
change -> config ’ daq_dir’  ==> ’ daq. dir’
change -> config ’ daq_mode’  ==> ’ daq. mode’
change -> config ’ daq_var’  ==> ’ daq. var’
change -> config ’ detection_filter’  ==> ’ alerts. detection_filter_memcap’
change -> config ’ enable_deep_teredo_inspection’  ==> ’ udp. deep_teredo_inspection’
change -> config ’ event_filter’  ==> ’ alerts. event_filter_memcap’
change -> config ’ max_attribute_hosts’  ==> ’ attribute_table. max_hosts’
change -> config ’ max_attribute_services_per_host’  ==> ’ attribute_table.  ↩
    max_services_per_host’
change -> config ’ nopcre’  ==> ’ detection. pcre_enable’
change -> config ’ pkt_count’  ==> ’ packets. limit’
change -> config ’ rate_filter’  ==> ’ alerts. rate_filter_memcap’
change -> config ’ react’  ==> ’ react. page’
change -> config ’ threshold’  ==> ’ alerts. event_filter_memcap’
change -> csv: ’dgmlen’ ==> ’dgm_len’
change -> csv: ’dst’ ==> ’dst_addr’
change -> csv: ’dstport’ ==> ’dst_port’
change -> csv: ’ethdst’ ==> ’eth_dst’
change -> csv: ’ethlen’ ==> ’eth_len’
change -> csv: ’ethsrc’ ==> ’eth_src’
change -> csv: ’ethtype’ ==> ’eth_type’
change -> csv: ’icmpcode’ ==> ’icmp_code’
change -> csv: ’icmpid’ ==> ’icmp_id’
change -> csv: ’icmpseq’ ==> ’icmp_seq’
change -> csv: ’icmptype’ ==> ’icmp_type’
change -> csv: 'iplen' ==> 'ip_len'
change -> csv: 'sig_generator' ==> 'gid'
change -> csv: 'sig_id' ==> 'sid'
change -> csv: 'sig_rev' ==> 'rev'
change -> csv: 'src' ==> 'src_addr'
change -> csv: 'srcport' ==> 'src_port'
change -> csv: 'tcpack' ==> 'tcp_ack'
change -> csv: 'tcpflags' ==> 'tcp_flags'
change -> csv: 'tcplen' ==> 'tcp_len'
change -> csv: 'tcpseq' ==> 'tcp_seq'
change -> csv: 'tcpwindow' ==> 'tcp_win'
change -> csv: 'udplength' ==> 'udp_len'
change -> detection: 'ac' ==> 'ac_full_q'
change -> detection: 'ac-banded' ==> 'ac_banded'
change -> detection: 'ac-bnfa' ==> 'ac_bnfa_q'
change -> detection: 'ac-bnfa-nq' ==> 'ac_bnfa'
change -> detection: 'ac-bnfa-q' ==> 'ac_bnfa_q'
change -> detection: 'ac-nq' ==> 'ac_full'
change -> detection: 'ac-q' ==> 'ac_full_q'
change -> detection: 'ac-sparsebands' ==> 'ac_sparse_bands'
change -> detection: 'ac-split' ==> 'ac_full_q'
change -> detection: 'ac-split' ==> 'split_any_any'
change -> detection: 'ac-std' ==> 'ac_std'
change -> detection: 'acs' ==> 'ac_sparse'
change -> detection: 'bleedover-port-limit' ==> 'bleedover_port_limit'
change -> detection: 'intel-cpm' ==> 'intel_cpm'
change -> detection: 'lowmem' ==> 'lowmem_q'
change -> detection: 'lowmem-nq' ==> 'lowmem'
change -> detection: 'lowmem-q' ==> 'lowmem_q'
change -> detection: 'max-pattern-len' ==> 'max_pattern_len'
change -> detection: 'search-method' ==> 'search_method'
change -> detection: 'search-optimize' ==> 'search_optimize'
change -> detection: 'split-any-any' ==> 'split_any_any'
change -> dns: 'ports' ==> 'bindings'
change -> event_filter: 'gen_id' ==> 'gid'
change -> event_filter: 'sig_id' ==> 'sid'
change -> event_filter: 'threshold' ==> 'event_filter'
change -> file: 'config file: file_block_timeout' ==> 'block_timeout'
change -> file: 'config file: file_type_depth' ==> 'type_depth'
change -> file: 'config file: signature' ==> 'enable_signature'
change -> file: 'config file: type_id' ==> 'enable_type'
change -> frag3_engine: 'min_fragment_length' ==> 'min_frag_length'
change -> frag3_engine: 'overlap_limit' ==> 'max_overlaps'
change -> frag3_engine: 'policy bsd-right' ==> 'policy = bsd_right'
change -> frag3_engine: 'timeout' ==> 'session_timeout'
change -> ftp_telnet_protocol: 'alt_max_param_len' ==> 'cmd_validity'
change -> ftp_telnet_protocol: 'data_chan' ==> 'ignore_data_chan'
change -> ftp_telnet_protocol: 'ports' ==> 'bindings'
change -> gtp: 'ports' ==> 'gtp_ports'
change -> http_inspect: 'http_inspect' ==> 'http_global'
change -> http_inspect_server: 'apache_whitespace' ==> 'profile.apache_whitespace'
change -> http_inspect_server: 'ascii' ==> 'profile.ascii'
change -> http_inspect_server: 'bare_byte' ==> 'profile.bare_byte'
change -> http_inspect_server: 'chunk_length' ==> 'profile.chunk_length'
change -> http_inspect_server: 'client_flow_depth' ==> 'profile.client_flow_depth'
change -> http_inspect_server: 'directory' ==> 'profile.directory'
change -> http_inspect_server: 'double_decode' ==> 'profile.double_decode'
change -> http_inspect_server: 'enable_cookie' ==> 'enable_cookies'
change -> http_inspect_server: 'flow_depth' ==> 'server_flow_depth'
change -> http_inspect_server: 'http_inspect_server' ==> 'http_inspect'
change -> http_inspect_server: 'iis_backslash' ==> 'profile.iis_backslash'
change -> http_inspect_server: 'iis_delimiter' ==> 'profile.iis_delimiter'
change -> http_inspect_server: 'iis_unicode' ==> 'profile.iis_unicode'
change -> http_inspect_server: 'max_header_length' ==> 'profile.max_header_length'
change -> http_inspect_server: 'max_headers' ==> 'profile.max_headers'
change -> http_inspect_server: 'max_spaces' ==> 'profile.max_spaces'
change -> http_inspect_server: 'multi_slash' ==> 'profile.multi_slash'
change -> http_inspect_server: 'non_rfc_char' ==> 'non_rfc_chars'
change -> http_inspect_server: 'non_strict' ==> 'profile.non_strict'
change -> http_inspect_server: 'normalize_utf' ==> 'profile.normalize_utf'
change -> http_inspect_server: 'ports' ==> 'bindings'
change -> http_inspect_server: 'u_encode' ==> 'profile.u_encode'
change -> http_inspect_server: 'utf_8' ==> 'profile.utf_8'
change -> http_inspect_server: 'webroot' ==> 'profile.webroot'
change -> http_inspect_server: 'whitespace_chars' ==> 'profile.whitespace_chars'
change -> imap: 'ports' ==> 'bindings'
change -> paf_max: 'paf_max [0:63780]' ==> 'max_pdu [1460:63780]'
change -> perfmonitor: 'accumulate' ==> 'reset = false'
change -> perfmonitor: 'flow-file' ==> 'flow_file = true'
change -> perfmonitor: 'flow-ip' ==> 'flow_ip'
change -> perfmonitor: 'flow-ip-file' ==> 'flow_ip_file = true'
change -> perfmonitor: 'flow-ip-memcap' ==> 'flow_ip_memcap'
change -> perfmonitor: 'flow-ports' ==> 'flow_ports'
change -> perfmonitor: 'pktcnt' ==> 'packets'
change -> perfmonitor: 'snortfile' ==> 'file = true'
change -> perfmonitor: 'time' ==> 'seconds'
change -> policy_mode: 'inline_test' ==> 'inline-test'
change -> pop: 'ports' ==> 'bindings'
change -> ppm: 'max-pkt-time' ==> 'max_pkt_time'
change -> ppm: 'max-rule-time' ==> 'max_rule_time'
change -> ppm: 'pkt-log' ==> 'pkt_log'
change -> ppm: 'rule-log' ==> 'rule_log'
change -> ppm: 'suspend-timeout' ==> 'suspend_timeout'
change -> preprocessor 'normalize_ icmp4' ==> 'normalize. icmp4'
change -> preprocessor 'normalize_ icmp6' ==> 'normalize. icmp6'
change -> preprocessor 'normalize_ ip6' ==> 'normalize. ip6'
change -> profile: 'print' ==> 'count'
change -> rate_filter: 'gen_id' ==> 'gid'
change -> rate_filter: 'sig_id' ==> 'sid'
change -> rule_state: 'disabled' ==> 'enable'
change -> rule_state: 'enabled' ==> 'enable'
change -> sfportscan: 'proto' ==> 'protos'
change -> sfportscan: 'scan_type' ==> 'scan_types'
change -> sip: 'ports' ==> 'bindings'
change -> smtp: 'ports' ==> 'bindings'
change -> ssh: 'server_ports' ==> 'bindings'
change -> ssl: 'ports' ==> 'bindings'
change -> stream5_global: 'max_active_responses' ==> 'max_responses'
change -> stream5_global: 'max_icmp' ==> 'max_sessions'
change -> stream5_global: 'max_ip' ==> 'max_sessions'
change -> stream5_global: 'max_tcp' ==> 'max_sessions'
change -> stream5_global: 'max_udp' ==> 'max_sessions'
change -> stream5_global: 'min_response_seconds' ==> 'min_interval'
change -> stream5_global: 'prune_log_max' ==> 'histogram'
change -> stream5_global: 'tcp_cache_nominal_timeout' ==> 'pruning_timeout'
change -> stream5_global: 'tcp_cache_pruning_timeout' ==> 'idle_timeout'
change -> stream5_global: 'udp_cache_nominal_timeout' ==> 'idle_timeout'
change -> stream5_global: 'udp_cache_pruning_timeout' ==> 'pruning_timeout'
change -> stream5_ip: 'timeout' ==> 'session_timeout'
change -> stream5_tcp: 'bind_to' ==> 'bindings'
change -> stream5_tcp: 'dont_reassemble_async' ==> 'reassemble_async'
change -> stream5_tcp: 'max_queued_bytes' ==> 'queue_limit.max_bytes'
change -> stream5_tcp: 'max_queued_segs' ==> 'queue_limit.max_segments'
change -> stream5_tcp: 'policy hpux' ==> 'stream_tcp.policy = hpux11'
change -> stream5_tcp: ’timeout’ ==> ’session_timeout’
change -> stream5_tcp: ’use_static_footprint_sizes’ ==> ’footprint’
change -> stream5_udp: ’timeout’ ==> ’session_timeout’
change -> suppress: ’gen_id’ ==> ’gid’
change -> suppress: ’sig_id’ ==> ’sid’
change -> syslog: ’log_alert’ ==> ’level = alert’
change -> syslog: ’log_auth’ ==> ’facility = auth’
change -> syslog: ’log_authpriv’ ==> ’facility = authpriv’
change -> syslog: ’log_cons’ ==> ’options = cons’
change -> syslog: ’log_crit’ ==> ’level = crit’
change -> syslog: ’log_daemon’ ==> ’facility = daemon’
change -> syslog: ’log_debug’ ==> ’level = debug’
change -> syslog: ’log_emerg’ ==> ’level = emerg’
change -> syslog: ’log_err’ ==> ’level = err’
change -> syslog: ’log_info’ ==> ’level = info’
change -> syslog: ’log_local0’ ==> ’facility = local0’
change -> syslog: ’log_local1’ ==> ’facility = local1’
change -> syslog: ’log_local2’ ==> ’facility = local2’
change -> syslog: ’log_local3’ ==> ’facility = local3’
change -> syslog: ’log_local4’ ==> ’facility = local4’
change -> syslog: ’log_local5’ ==> ’facility = local5’
change -> syslog: ’log_local6’ ==> ’facility = local6’
change -> syslog: ’log_local7’ ==> ’facility = local7’
change -> syslog: ’log_ndelay’ ==> ’options = ndelay’
change -> syslog: ’log_notice’ ==> ’level = notice’
change -> syslog: ’log_perror’ ==> ’options = perror’
change -> syslog: ’log_pid’ ==> ’options = pid’
change -> syslog: ’log_user’ ==> ’facility = user’
change -> syslog: ’log_warning’ ==> ’level = warning’
change -> threshold: ’ips_option: threshold’ ==> ’event_filter’
change -> unified2: ’ alert_unified2’ ==> ’unified2’
change -> unified2: ’ log_unified2’ ==> ’unified2’
change -> unified2: ’ unified2’ ==> ’unified2’
deleted -> arpspoof: ’unicast’
deleted -> attribute_table: ’<FRAG_POLICY>hpux</FRAG_POLICY>’
deleted -> attribute_table: ’<FRAG_POLICY>irix</FRAG_POLICY>’
deleted -> attribute_table: ’<FRAG_POLICY>old-linux</FRAG_POLICY>’
deleted -> attribute_table: ’<FRAG_POLICY>unknown</FRAG_POLICY>’
deleted -> attribute_table: ’<STREAM_POLICY>noack</STREAM_POLICY>’
deleted -> attribute_table: ’<STREAM_POLICY>unknown</STREAM_POLICY>’
deleted -> config ’ cs_dir’
deleted -> config ’ disable_attribute_reload_thread’
deleted -> config ’ disable_decode_alerts’
deleted -> config ’ disable_decode_drops’
deleted -> config ’ disable_ipopt_alerts’
deleted -> config ’ disable_ipopt_drops’
deleted -> config ’ disable_tcpopt_alerts’
deleted -> config ’ disable_tcpopt_drops’
deleted -> config ’ disable_tcpopt_experimental_alerts’
deleted -> config ’ disable_tcpopt_experimental_drops’
deleted -> config ’ disable_tcpopt_obsolete_alerts’
deleted -> config ’ disable_tcpopt_obsolete_drops’
deleted -> config ’ disable_tcpopt_ttcp_alerts’
deleted -> config ’ disable_ttcp_alerts’
deleted -> config ’ disable_ttcp_drops’
deleted -> config ’ dump_dynamic_rules_path’
deleted -> config ’ enable_decode_drops’
deleted -> config ’ enable_decode_oversized_alerts’
deleted -> config ’ enable_decode_oversized_drops’
deleted -> config ’ enable_ipopt_drops’
deleted -> config ’ enable_tcpopt_drops’
deleted -> config ’ enable_tcpopt_experimental_drops’
deleted -> config ’ enable_tcpopt_obsolete_drops’
deleted -> config ’ enable_tcpopt_ttcp_drops’
deleted -> config ’ enable_ttcp_drops’
deleted -> config ’ flexresp2_attempts’
deleted -> config ’ flexresp2_interface’
deleted -> config ’ flexresp2_memcap’
deleted -> config ’ flexresp2_rows’
deleted -> config ’ flowbits_size’
deleted -> config ’ include_vlan_in_alerts’
deleted -> config ’ interface’
deleted -> config ’ layer2resets’
deleted -> config ’ policy_version’
deleted -> config ’ so_rule_memcap’
deleted -> csv: ’<filename> can no longer be specific’
deleted -> csv: ’default’
deleted -> csv: ’trheader’
deleted -> detection: ’mwm’
deleted -> dns: ’enable_experimental_types’
deleted -> dns: ’enable_obsolete_types’
deleted -> dns: ’enable_rdata_overflow’
deleted -> fast: ’<filename> can no longer be specific’
deleted -> frag3_engine: ’detect_anomalies’
deleted -> frag3_global: ’disabled’
deleted -> ftp_telnet_protocol: ’detect_anomalies’
deleted -> full: ’<filename> can no longer be specific’
deleted -> http_inspect: ’disabled’
deleted -> http_inspect_server: ’no_alerts’
deleted -> imap: ’disabled’
deleted -> imap: ’max_mime_mem’
deleted -> imap: ’memcap’
deleted -> perfmonitor: ’atexitonly’
deleted -> perfmonitor: ’atexitonly: base-stats’
deleted -> perfmonitor: ’atexitonly: events-stats’
deleted -> perfmonitor: ’atexitonly: flow-ip-stats’
deleted -> perfmonitor: ’atexitonly: flow-stats’
deleted -> pop: ’disabled’
deleted -> pop: ’max_mime_mem’
deleted -> pop: ’memcap’
deleted -> ppm: ’debug-pkts’
deleted -> react: ’block’
deleted -> react: ’warn’
deleted -> rpc_decode: ’alert_fragments’
deleted -> rpc_decode: ’no_alert_incomplete’
deleted -> rpc_decode: ’no_alert_large_fragments’
deleted -> rpc_decode: ’no_alert_multiple_requests’
deleted -> rule_state: ’action’
deleted -> sfportscan: ’detect_ack_scans’
deleted -> sfportscan: ’disabled’
deleted -> sfportscan: ’logfile’
deleted -> sip: ’disabled’
deleted -> smtp: ’alert_unknown_cmds’
deleted -> smtp: ’disabled’
deleted -> smtp: ’enable_mime_decoding’
deleted -> smtp: ’inspection_type’
deleted -> smtp: ’max_mime_depth’
deleted -> smtp: ’max_mime_mem’
deleted -> smtp: ’memcap’
deleted -> smtp: ’no_alerts’
deleted -> smtp: ’print_cmds’
deleted -> ssh: ’autodetect’
deleted -> ssh: ’enable_badmsgdir’
deleted -> ssh: ’enable_paysize’
deleted -> ssh: 'enable_protomismatch'
deleted -> ssh: 'enable_recognition'
deleted -> ssh: 'enable_respoverflow'
deleted -> ssh: 'enable_srvoverflow'
deleted -> ssh: 'enable_ssh1crc32'
deleted -> ssl: 'noinspect_encrypted'
deleted -> stream5_global: 'disabled'
deleted -> stream5_global: 'flush_on_alert'
deleted -> stream5_global: 'no_midstream_drop_alerts'
deleted -> stream5_tcp: 'check_session_hijacking'
deleted -> stream5_tcp: 'detect_anomalies'
deleted -> stream5_tcp: 'dont_store_large_packets'
deleted -> stream5_tcp: 'policy noack'
deleted -> stream5_tcp: 'policy unknown'
deleted -> tcpdump: '<filename> can no longer be specific'
deleted -> test: 'file'
deleted -> test: 'stdout'
deleted -> unified2: 'filename'
```

19.11 Module Listing

- **ack** (ips_option): rule option to match on TCP ack numbers
- **active** (basic): configure responses
- **alert_csv** (logger): output event in csv format
- **alert_fast** (logger): output event with brief text format
- **alert_full** (logger): output event with full packet dump
- **alert_sfsocket** (logger): output event over socket
- **alert_syslog** (logger): output event to syslog
- **alerts** (basic): configure alerts
- **appid** (inspector): application and service identification
- **appids** (ips_option): detection option for application ids
- **arp** (codec): support for address resolution protocol
- **arp_spoof** (inspector): detect ARP attacks and anomalies
- **asn1** (ips_option): rule option for asn1 detection
- **attribute_table** (basic): configure hosts loading
- **auth** (codec): support for IP authentication header
- **back_orifice** (inspector): back orifice detection
- **base64_decode** (ips_option): rule option to decode base64 data - must be used with base64_data option
- **binder** (inspector): configure processing based on CIDRs, ports, services, etc.
- **bufferlen** (ips_option): rule option to check length of current buffer
- **byte_extract** (ips_option): rule option to convert data to an integer variable
- **byte_jump** (ips_option): rule option to move the detection cursor
- **byte_test** (ips_option): rule option to convert data to integer and compare

- **ciscometadata** (codec): support for cisco metadata
- **classifications** (basic): define rule categories with priority
- **classtype** (ips_option): general rule option for rule classification
- **content** (ips_option): payload rule option for basic pattern matching
- **cvs** (ips_option): payload rule option for detecting specific attacks
- **daq** (basic): configure packet acquisition interface
- **dce_iface** (ips_option): detection option to check dcerpc interface
- **dce_opnum** (ips_option): detection option to check dcerpc operation number
- **dce_smb** (inspector): dce over smb inspection
- **dce_stub_data** (ips_option): sets the cursor to dcerpc stub data
- **dce_tcp** (inspector): dce over tcp inspection
- **dce_udp** (inspector): dce over udp inspection
- **decode** (basic): general decoder rules
- **detection** (basic): configure general IPS rule processing parameters
- **detection_filter** (ips_option): rule option to require multiple hits before a rule generates an event
- **dnp3** (inspector): dnp3 inspection
- **dnp3_data** (ips_option): sets the cursor to dnp3 data
- **dnp3_func** (ips_option): detection option to check dnp3 function code
- **dnp3_ind** (ips_option): detection option to check dnp3 indicator flags
- **dnp3_obj** (ips_option): detection option to check dnp3 object headers
- **dns** (inspector): dns inspection
- **dsize** (ips_option): rule option to test payload size
- **erspan2** (codec): support for encapsulated remote switched port analyzer - type 2
- **erspan3** (codec): support for encapsulated remote switched port analyzer - type 3
- **esp** (codec): support for encapsulating security payload
- **eth** (codec): support for ethernet protocol (DLT 1) (DLT 51)
- **event_filter** (basic): configure thresholding of events
- **event_queue** (basic): configure event queue parameters
- **fabricpath** (codec): support for fabricpath
- **file_connector** (connector): implement the file based connector
- **file_data** (ips_option): rule option to set detection cursor to file data
- **file_id** (basic): configure file identification
- **file_log** (inspector): log file event to file.log
- **file_type** (ips_option): rule option to check file type
- **flags** (ips_option): rule option to test TCP control flags

- **flow** (ips_option): rule option to check session properties
- **flowbits** (ips_option): rule option to set and test arbitrary boolean flags
- **fragbits** (ips_option): rule option to test IP frag flags
- **fragoffset** (ips_option): rule option to test IP frag offset
- **ftp_client** (inspector): FTP client configuration module for use with ftp_server
- **ftp_data** (inspector): FTP data channel handler
- **ftp_server** (inspector): main FTP module; ftp_client should also be configured
- **gid** (ips_option): rule option specifying rule generator
- **gre** (codec): support for generic routing encapsulation
- **gtp** (codec): support for general-packet-radio-service tunnelling protocol
- **gtp_info** (ips_option): rule option to check gtp info element
- **gtp_inspect** (inspector): gtp control channel inspection
- **gtp_type** (ips_option): rule option to check gtp types
- **gtp_version** (ips_option): rule option to check gtp version
- **high_availability** (basic): implement flow tracking high availability
- **host_cache** (basic): configure hosts
- **host_tracker** (basic): configure hosts
- **hosts** (basic): configure hosts
- **http_client_body** (ips_option): rule option to set the detection cursor to the request body
- **http_cookie** (ips_option): rule option to set the detection cursor to the HTTP cookie
- **http_header** (ips_option): rule option to set the detection cursor to the normalized headers
- **http_inspect** (inspector): HTTP inspector
- **http_method** (ips_option): rule option to set the detection cursor to the HTTP request method
- **http_raw_cookie** (ips_option): rule option to set the detection cursor to the unnormalized cookie
- **http_raw_header** (ips_option): rule option to set the detection cursor to the unnormalized headers
- **http_raw_request** (ips_option): rule option to set the detection cursor to the unnormalized request line
- **http_raw_status** (ips_option): rule option to set the detection cursor to the unnormalized status line
- **http_raw_trailer** (ips_option): rule option to set the detection cursor to the unnormalized trailers
- **http_raw_uri** (ips_option): rule option to set the detection cursor to the unnormalized URI
- **http_stat_code** (ips_option): rule option to set the detection cursor to the HTTP status code
- **http_stat_msg** (ips_option): rule option to set the detection cursor to the HTTP status message
- **http_trailer** (ips_option): rule option to set the detection cursor to the normalized trailers
- **http_uri** (ips_option): rule option to set the detection cursor to the normalized URI buffer
- **http_version** (ips_option): rule option to set the detection cursor to the version buffer
- **icmp4** (codec): support for Internet control message protocol v4

- **icmp6** (codec): support for Internet control message protocol v6
- **icmp_id** (ips_option): rule option to check ICMP ID
- **icmp_seq** (ips_option): rule option to check ICMP sequence number
- **icode** (ips_option): rule option to check ICMP code
- **id** (ips_option): rule option to check the IP ID field
- **igmp** (codec): support for Internet group management protocol
- **imap** (inspector): imap inspection
- **ip_proto** (ips_option): rule option to check the IP protocol number
- **ipopts** (ips_option): rule option to check for IP options
- **ips** (basic): configure IPS rule processing
- **ipv4** (codec): support for Internet protocol v4
- **ipv6** (codec): support for Internet protocol v6
- **isdataat** (ips_option): rule option to check for the presence of payload data
- **itype** (ips_option): rule option to check ICMP type
- **latency** (basic): packet and rule latency monitoring and control
- **log_codecs** (logger): log protocols in packet by layer
- **log_hext** (logger): output payload suitable for daq hext
- **log_pcap** (logger): log packet in pcap format
- **md5** (ips_option): payload rule option for hash matching
- **memory** (basic): memory management configuration
- **metadata** (ips_option): rule option for conveying arbitrary name, value data within the rule text
- **modbus** (inspector): modbus inspection
- **modbus_data** (ips_option): rule option to set cursor to modbus data
- **modbus_func** (ips_option): rule option to check modbus function code
- **modbus_unit** (ips_option): rule option to check modbus unit ID
- **mpls** (codec): support for multiprotocol label switching
- **msg** (ips_option): rule option summarizing rule purpose output with events
- **network** (basic): configure basic network parameters
- **normalizer** (inspector): packet scrubbing for inline mode
- **output** (basic): configure general output parameters
- **packet_capture** (inspector): raw packet dumping facility
- **packets** (basic): configure basic packet handling
- **pcre** (ips_option): rule option for matching payload data with pcre
- **perf_monitor** (inspector): performance monitoring and flow statistics collection
- **pgm** (codec): support for pragmatic general multicast

- **pkt_data** (ips_option): rule option to set the detection cursor to the normalized packet data
- **pop** (inspector): pop inspection
- **port_scan** (inspector): port scan inspector; also configure port_scan_global
- **port_scan_global** (inspector): shared settings for port_scan inspectors for use with port_scan
- **pppoe** (codec): support for point-to-point protocol over ethernet
- **priority** (ips_option): rule option for prioritizing events
- **process** (basic): configure basic process setup
- **profiler** (basic): configure profiling of rules and/or modules
- **rate_filter** (basic): configure rate filters (which change rule actions)
- **raw_data** (ips_option): rule option to set the detection cursor to the raw packet data
- **react** (ips_action): send response to client and terminate session
- **reference** (ips_option): rule option to indicate relevant attack identification system
- **references** (basic): define reference systems used in rules
- **regex** (ips_option): rule option for matching payload data with hyperscan regex
- **reject** (ips_action): terminate session with TCP reset or ICMP unreachable
- **rem** (ips_option): rule option to convey an arbitrary comment in the rule body
- **replace** (ips_option): rule option to overwrite payload data; use with rewrite action
- **reputation** (inspector): reputation inspection
- **rev** (ips_option): rule option to indicate current revision of signature
- **rewrite** (ips_action): overwrite packet contents
- **rpc** (ips_option): rule option to check SUNRPC CALL parameters
- **rpc_decode** (inspector): RPC inspector
- **rule_state** (basic): enable/disable specific IPS rules
- **sd_pattern** (ips_option): rule option for detecting sensitive data
- **search_engine** (basic): configure fast pattern matcher
- **seq** (ips_option): rule option to check TCP sequence number
- **session** (ips_option): rule option to check user data from TCP sessions
- **sha256** (ips_option): payload rule option for hash matching
- **sha512** (ips_option): payload rule option for hash matching
- **sid** (ips_option): rule option to indicate signature number
- **side_channel** (basic): implement the side-channel asynchronous messaging subsystem
- **sip** (inspector): sip inspection
- **sip_body** (ips_option): rule option to set the detection cursor to the request body
- **sip_header** (ips_option): rule option to set the detection cursor to the SIP header buffer
- **sip_method** (ips_option): detection option for sip stat code

- **sip_stat_code** (ips_option): detection option for sip stat code
- **smtp** (inspector): smtp inspection
- **snort** (basic): command line configuration and shell commands
- **so** (ips_option): rule option to call custom eval function
- **soid** (ips_option): rule option to specify a shared object rule ID
- **ssh** (inspector): ssh inspection
- **ssl** (inspector): ssl inspection
- **ssl_state** (ips_option): detection option for ssl state
- **ssl_version** (ips_option): detection option for ssl version
- **stream** (inspector): common flow tracking
- **stream_file** (inspector): stream inspector for file flow tracking and processing
- **stream_icmp** (inspector): stream inspector for ICMP flow tracking
- **stream_ip** (inspector): stream inspector for IP flow tracking and defragmentation
- **stream_reassemble** (ips_option): detection option for stream reassembly control
- **stream_size** (ips_option): detection option for stream size checking
- **stream_tcp** (inspector): stream inspector for TCP flow tracking and stream normalization and reassembly
- **stream_udp** (inspector): stream inspector for UDP flow tracking
- **stream_user** (inspector): stream inspector for user flow tracking and reassembly
- **suppress** (basic): configure event suppressions
- **tag** (ips_option): rule option to log additional packets
- **tcp** (codec): support for transmission control protocol
- **tcp_connector** (connector): implement the tcp stream connector
- **telnet** (inspector): telnet inspection and normalization
- **tos** (ips_option): rule option to check type of service field
- **ttl** (ips_option): rule option to check time to live field
- **udp** (codec): support for user datagram protocol
- **unified2** (logger): output event and packet in unified2 format file
- **vlan** (codec): support for local area network
- **window** (ips_option): rule option to check TCP window field
- **wizard** (inspector): inspector that implements port-independent protocol identification

19.12 Plugin Listing

- **codec::arp**: support for address resolution protocol
- **codec::auth**: support for IP authentication header
- **codec::ciscometadata**: support for cisco metadata
- **codec::erspan2**: support for encapsulated remote switched port analyzer - type 2
- **codec::erspan3**: support for encapsulated remote switched port analyzer - type 3
- **codec::esp**: support for encapsulating security payload
- **codec::eth**: support for ethernet protocol (DLT 1) (DLT 51)
- **codec::fabricpath**: support for fabricpath
- **codec::gre**: support for generic routing encapsulation
- **codec::gtp**: support for general-packet-radio-service tunnelling protocol
- **codec::icmp4**: support for Internet control message protocol v4
- **codec::icmp4_ip**: support for IP in ICMPv4
- **codec::icmp6**: support for Internet control message protocol v6
- **codec::icmp6_ip**: support for IP in ICMPv6
- **codec::igmp**: support for Internet group management protocol
- **codec::ipv4**: support for Internet protocol v4
- **codec::ipv6**: support for Internet protocol v6
- **codec::ipv6_dst_opts**: support for ipv6 destination options
- **codec::ipv6_frag**: support for IPv6 fragment decoding
- **codec::ipv6_hop_opts**: support for IPv6 hop options
- **codec::ipv6_mobility**: support for mobility
- **codec::ipv6_no_next**: sentinel codec
- **codec::ipv6_routing**: support for IPv6 routing extension
- **codec::llc**: support for logical link control
- **codec::mpls**: support for multiprotocol label switching
- **codec::pgm**: support for pragmatic general multicast
- **codec::ppp_encap**: support for point-to-point encapsulation
- **codec::pppoe_disc**: support for point-to-point discovery
- **codec::pppoe_sess**: support for point-to-point session
- **codec::sun_nd**: support for Sun ND
- **codec::swipe**: support for Swipe
- **codec::tcp**: support for transmission control protocol
- **codec::teredo**: support for teredo
- **codec::trans_bridge**: support for trans-bridging

- **codec::udp**: support for user datagram protocol
- **codec::user**: support for user sessions (DLT 230)
- **codec::vlan**: support for local area network
- **connector::file_connector**: implement the file based connector
- **connector::tcp_connector**: implement the tcp stream connector
- **inspector::appid**: application and service identification
- **inspector::arp_spoof**: detect ARP attacks and anomalies
- **inspector::back_orifice**: back orifice detection
- **inspector::binder**: configure processing based on CIDRs, ports, services, etc.
- **inspector::dce_smb**: dce over smb inspection
- **inspector::dce_tcp**: dce over tcp inspection
- **inspector::dce_udp**: dce over udp inspection
- **inspector::dnp3**: dnp3 inspection
- **inspector::dns**: dns inspection
- **inspector::file_log**: log file event to file.log
- **inspector::ftp_client**: FTP inspector client module
- **inspector::ftp_data**: FTP data channel handler
- **inspector::ftp_server**: FTP inspector server module
- **inspector::gtp_inspect**: gtp control channel inspection
- **inspector::http_inspect**: the new HTTP inspector!
- **inspector::imap**: imap inspection
- **inspector::modbus**: modbus inspection
- **inspector::normalizer**: packet scrubbing for inline mode
- **inspector::packet_capture**: raw packet dumping facility
- **inspector::perf_monitor**: performance monitoring and flow statistics collection
- **inspector::pop**: pop inspection
- **inspector::port_scan**: port scan inspector; also configure port_scan_global
- **inspector::port_scan_global**: shared settings for port_scan inspectors for use with port_scan
- **inspector::reputation**: reputation inspection
- **inspector::rpc_decode**: RPC inspector
- **inspector::sip**: sip inspection
- **inspector::smtp**: smtp inspection
- **inspector::ssh**: ssh inspection
- **inspector::ssl**: ssl inspection
- **inspector::stream**: common flow tracking

- **inspector::stream_file**: stream inspector for file flow tracking and processing
- **inspector::stream_icmp**: stream inspector for ICMP flow tracking
- **inspector::stream_ip**: stream inspector for IP flow tracking and defragmentation
- **inspector::stream_tcp**: stream inspector for TCP flow tracking and stream normalization and reassembly
- **inspector::stream_udp**: stream inspector for UDP flow tracking
- **inspector::stream_user**: stream inspector for user flow tracking and reassembly
- **inspector::telnet**: telnet inspection and normalization
- **inspector::wizard**: inspector that implements port-independent protocol identification
- **ips_action::react**: send response to client and terminate session
- **ips_action::reject**: terminate session with TCP reset or ICMP unreachable
- **ips_action::rewrite**: overwrite packet contents
- **ips_option::ack**: rule option to match on TCP ack numbers
- **ips_option::appids**: detection option for application ids
- **ips_option::asn1**: rule option for asn1 detection
- **ips_option::base64_data**: set detection cursor to decoded Base64 data
- **ips_option::base64_decode**: rule option to decode base64 data - must be used with base64_data option
- **ips_option::bufferlen**: rule option to check length of current buffer
- **ips_option::byte_extract**: rule option to convert data to an integer variable
- **ips_option::byte_jump**: rule option to move the detection cursor
- **ips_option::byte_test**: rule option to convert data to integer and compare
- **ips_option::classtype**: general rule option for rule classification
- **ips_option::content**: payload rule option for basic pattern matching
- **ips_option::cvs**: payload rule option for detecting specific attacks
- **ips_option::dce_iface**: detection option to check dcerpc interface
- **ips_option::dce_opnum**: detection option to check dcerpc operation number
- **ips_option::dce_stub_data**: sets the cursor to dcerpc stub data
- **ips_option::detection_filter**: rule option to require multiple hits before a rule generates an event
- **ips_option::dnp3_data**: sets the cursor to dnp3 data
- **ips_option::dnp3_func**: detection option to check dnp3 function code
- **ips_option::dnp3_ind**: detection option to check dnp3 indicator flags
- **ips_option::dnp3_obj**: detection option to check dnp3 object headers
- **ips_option::dsize**: rule option to test payload size
- **ips_option::file_data**: rule option to set detection cursor to file data
- **ips_option::file_type**: rule option to check file type
- **ips_option::flags**: rule option to test TCP control flags

- **ips_option::flow**: rule option to check session properties
- **ips_option::flowbits**: rule option to set and test arbitrary boolean flags
- **ips_option::fragbits**: rule option to test IP frag flags
- **ips_option::fragoffset**: rule option to test IP frag offset
- **ips_option::gid**: rule option specifying rule generator
- **ips_option::gtp_info**: rule option to check gtp info element
- **ips_option::gtp_type**: rule option to check gtp types
- **ips_option::gtp_version**: rule option to check gtp version
- **ips_option::http_client_body**: rule option to set the detection cursor to the request body
- **ips_option::http_cookie**: rule option to set the detection cursor to the HTTP cookie
- **ips_option::http_header**: rule option to set the detection cursor to the normalized headers
- **ips_option::http_method**: rule option to set the detection cursor to the HTTP request method
- **ips_option::http_raw_cookie**: rule option to set the detection cursor to the unnormalized cookie
- **ips_option::http_raw_header**: rule option to set the detection cursor to the unnormalized headers
- **ips_option::http_raw_request**: rule option to set the detection cursor to the unnormalized request line
- **ips_option::http_raw_status**: rule option to set the detection cursor to the unnormalized status line
- **ips_option::http_raw_trailer**: rule option to set the detection cursor to the unnormalized trailers
- **ips_option::http_raw_uri**: rule option to set the detection cursor to the unnormalized URI
- **ips_option::http_stat_code**: rule option to set the detection cursor to the HTTP status code
- **ips_option::http_stat_msg**: rule option to set the detection cursor to the HTTP status message
- **ips_option::http_trailer**: rule option to set the detection cursor to the normalized trailers
- **ips_option::http_uri**: rule option to set the detection cursor to the normalized URI buffer
- **ips_option::http_version**: rule option to set the detection cursor to the version buffer
- **ips_option::icmp_id**: rule option to check ICMP ID
- **ips_option::icmp_seq**: rule option to check ICMP sequence number
- **ips_option::icode**: rule option to check ICMP code
- **ips_option::id**: rule option to check the IP ID field
- **ips_option::ip_proto**: rule option to check the IP protocol number
- **ips_option::ipopts**: rule option to check for IP options
- **ips_option::isdataat**: rule option to check for the presence of payload data
- **ips_option::itype**: rule option to check ICMP type
- **ips_option::md5**: payload rule option for hash matching
- **ips_option::metadata**: rule option for conveying arbitrary name, value data within the rule text
- **ips_option::modbus_data**: rule option to set cursor to modbus data
- **ips_option::modbus_func**: rule option to check modbus function code

- **ips_option::modbus_unit**: rule option to check modbus unit ID
- **ips_option::msg**: rule option summarizing rule purpose output with events
- **ips_option::pcre**: rule option for matching payload data with pcre
- **ips_option::pkt_data**: rule option to set the detection cursor to the normalized packet data
- **ips_option::priority**: rule option for prioritizing events
- **ips_option::raw_data**: rule option to set the detection cursor to the raw packet data
- **ips_option::reference**: rule option to indicate relevant attack identification system
- **ips_option::regex**: rule option for matching payload data with hyperscan regex
- **ips_option::rem**: rule option to convey an arbitrary comment in the rule body
- **ips_option::replace**: rule option to overwrite payload data; use with rewrite action
- **ips_option::rev**: rule option to indicate current revision of signature
- **ips_option::rpc**: rule option to check SUNRPC CALL parameters
- **ips_option::sd_pattern**: rule option for detecting sensitive data
- **ips_option::seq**: rule option to check TCP sequence number
- **ips_option::session**: rule option to check user data from TCP sessions
- **ips_option::sha256**: payload rule option for hash matching
- **ips_option::sha512**: payload rule option for hash matching
- **ips_option::sid**: rule option to indicate signature number
- **ips_option::sip_body**: rule option to set the detection cursor to the request body
- **ips_option::sip_header**: rule option to set the detection cursor to the SIP header buffer
- **ips_option::sip_method**: detection option for sip stat code
- **ips_option::sip_stat_code**: detection option for sip stat code
- **ips_option::so**: rule option to call custom eval function
- **ips_option::soid**: rule option to specify a shared object rule ID
- **ips_option::ssl_state**: detection option for ssl state
- **ips_option::ssl_version**: detection option for ssl version
- **ips_option::stream_reassemble**: detection option for stream reassembly control
- **ips_option::stream_size**: detection option for stream size checking
- **ips_option::tag**: rule option to log additional packets
- **ips_option::tos**: rule option to check type of service field
- **ips_option::ttl**: rule option to check time to live field
- **ips_option::window**: rule option to check TCP window field
- **logger::alert_csv**: output event in csv format
- **logger::alert_fast**: output event with brief text format
- **logger::alert_full**: output event with full packet dump

- **logger::alert_sfsocket**: output event over socket
- **logger::alert_syslog**: output event to syslog
- **logger::log_codecs**: log protocols in packet by layer
- **logger::log_hext**: output payload suitable for daq hext
- **logger::log_pcap**: log packet in pcap format
- **logger::unified2**: output event and packet in unified2 format file
- **piglet::pp_codec**: Codec piglet
- **piglet::pp_inspector**: Inspector piglet
- **piglet::pp_ips_action**: Ips action piglet
- **piglet::pp_ips_option**: Ips option piglet
- **piglet::pp_logger**: Logger piglet
- **piglet::pp_search_engine**: Search engine piglet
- **piglet::pp_so_rule**: SO rule piglet
- **piglet::pp_test**: Test piglet
- **search_engine::ac_banded**: Aho-Corasick Banded (high memory, moderate performance)
- **search_engine::ac_bnfa**: Aho-Corasick Binary NFA (low memory, high performance) MPSE
- **search_engine::ac_full**: Aho-Corasick Full (high memory, best performance), implements search_all()
- **search_engine::ac_sparse**: Aho-Corasick Sparse (high memory, moderate performance) MPSE
- **search_engine::ac_sparse_bands**: Aho-Corasick Sparse-Banded (high memory, moderate performance) MPSE
- **search_engine::ac_std**: Aho-Corasick Full (high memory, best performance) MPSE
- **search_engine::hyperscan**: intel hyperscan-based mpse with regex support

19.13 Bugs

19.13.1 Build

- Enabling large pcap may erroneously affect the number of packets processed from pcaps.
- Enabling debug messages may erroneously affect the number of packets processed from pcaps.
- g++ 4.9.2 with -O3 reports:

  ```
  src/service_inspectors/back_orifice/back_orifice.cc:231:25: warning:
  iteration 930u invokes undefined behavior [-Waggressive-loop-optimizations]
  ```

- Building with clang and autotools on Linux will show the following warning many times. Please ignore.

  ```
  clang: warning: argument unused during compilation: '-pthread'
  ```

19.13.2 Config

- Parsing issue with IP lists. can't parse rules with $EXTERNAL_NET defined as below because of the space between ! and 10.

```
HOME_NET = [[ 10.0.17.0/24 10.0.14.0/24 10.247.0.0/16 10.246.0.0/16 ]]
EXTERNAL_NET = '! ' .. HOME_NET
```

- Multiple versions of luajit scripts are not handled correctly. The first loaded version will always be executed even though plugin manager saves the correct version.
- When using -c and -L together, the last on the command line wins (-c -L will dump; -L -c will analyze).
- Modules instantiated by command line only will not get default settings unless hard-coded. This notably applies to -A and -L options.
- --lua can only be used in addition to, not in place of, a -c config. Ideally, --lua could be used in lieu of -c.

19.13.3 Rules

- metdata:service foo; metadata:service foo; won't cause a duplicate service warning as does metadata:service foo, service foo;
- ip_proto doesn't work properly with reassembled packets so it can't be used to restrict the protocol of service rules.

19.13.4 snort2lua

- uricontent:"foo"; content:"bar"; → http_uri; content:"foo"; content:"bar"; (missing pkt_data)
- stream_tcp ports and protocols both go into a single binder.when; this is incorrect as the when fields are logically anded together (ie must all be true). Should create 2 separate bindings.
- There is a bug in pps_stream_tcp.cc.. when stream_tcp: is specified without any arguments, snort2lua doesn't convert it. Same for stream_udp.
- Loses the ip list delimiters [ ]; change to ( )

```
in snort.conf: var HOME_NET [A,B,C]
in snort.lua: HOME_NET = [[A B C]]
```

- Won't convert packet rules (alert tcp etc.) to service rules (alert http etc.).
- alert_fast and alert_full: output configuration includes "file = *foo.bar*", but file is a bool and you cannot specify an output file name in the configuration.
- preprocessor ports option: ports <number> not supported.

19.13.5 Runtime

- -B <mask> feature does not work. It does ordinary IP address obfuscation instead of using the mask.
- Obfuscation does not work for csv format.
- The hext DAQ will append a newline to text lines (starting with ").
- The hext DAQ does not support embedded quotes in text lines (use hex lines as a workaround).
- stream_tcp alert squash mechanism incorrectly squashes alerts for different TCP packets.