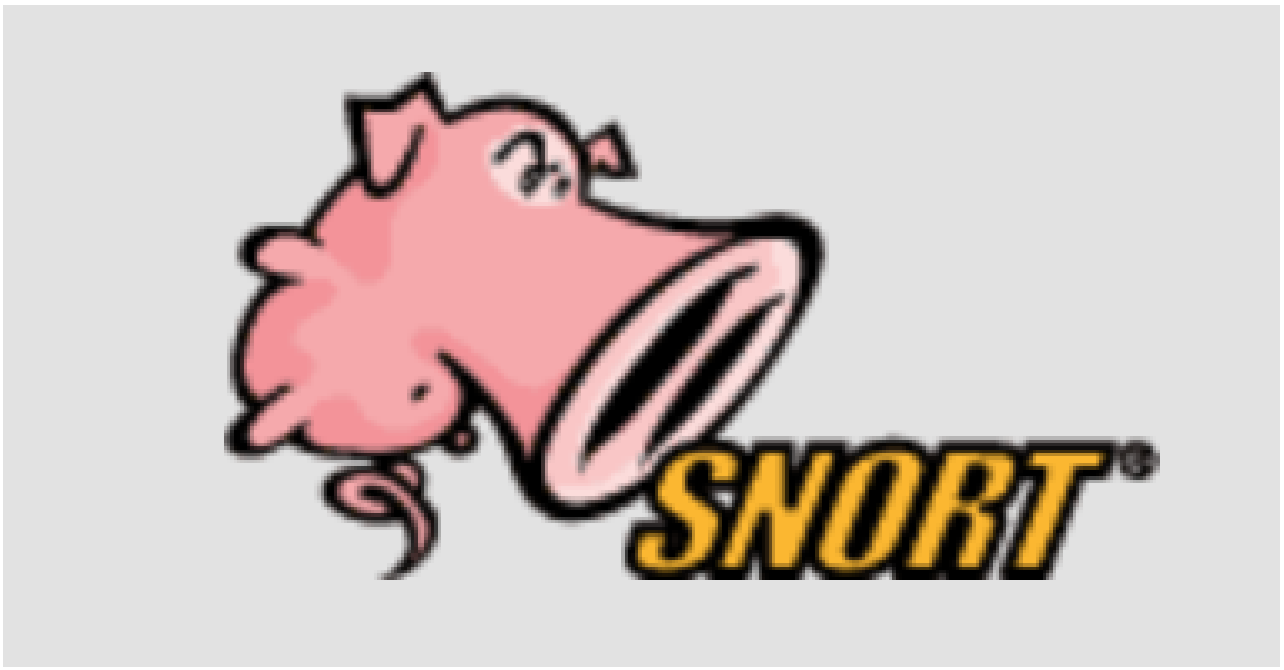
Snort++ User Manual

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents




```
 ,,_     -*> Snort++ <*-
o"  )~   Version 3.0.0-a1 (Build 131) from 2.9.6-9
 ''''    By Martin Roesch & The Snort Team
         http://snort.org/snort/snort-team
         Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
         Copyright (C) 1998-2013 Sourcefire, Inc., et al.
```

1 Overview

Snort++ is an updated version of the Snort IPS (intrusion prevention system). This document assumes you have some familiarity with Snort and are looking to see what Snort++ has to offer. Here are some of the basic goals for Snort++:

- Support multiple packet processing threads
- Use a shared configuration and attribute table
- Use a simple, scriptable configuration
- Make key components pluggable
- Autogenerate reference documentation
- Autodetect services for portless configuration
- Support sticky buffers in rules
- Provide better cross platform support

The above goals are met with this first alpha release. Additional, longer-term goals are:

- Use a shared network map
- Support pipelining of packet processing
- Support hardware offload and data plane integration

- Rewrite critical modules like TCP reassembly and HTTP inspection
- Support proxy mode
- Facilitate component testing
- Simplify memory management
- Provide all of Snort's functionality

This first alpha release is based on Snort 2.9.6-9 and excludes all but one of Snort's dynamic preprocessors. Work is underway to port that functionality and additions will be rolled out as they become available.

1.1 Configuration

Note that retaining backwards compatibility is not a goal. While Snort++ leverages some of the Snort code base, a lot has changed. The configuration of Snort++ is done with Lua, so your old conf won't work as is. Rules are still text based but nonetheless incompatible. However, Snort2Lua will help you convert your conf and rules to the new format.

The original Snort manual may be useful for some background information not yet documented for Snort++. The configuration differences are given in this manual.

1.2 Modules

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 }
```

1.3 Plugins

There are several plugin types:

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

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.

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.

1.4 Scripts

Some things just need to be tweaked or prototyped quickly. In addition to the Lua conf, which is a script that can contain functions to compute settings, etc., you can also script Loggers and IpsOptions.

1.5 New Http Inspector

The Http Inspector is being rewritten from scratch. The incomplete work-in-progress is included in this release as "new_http_inspect". The original Http Inspector remains available for regular use.

The new Http Inspector is based entirely on processing HTTP messages that it parses from the TCP data stream. This is in contrast to its predecessor which frequently works directly with raw packets. The new approach makes the inspector much simpler, enabling easy addition of new features. It also renders HTTP inspection independent of how the message is subdivided into packets, neutralizing many potential evasions.

The inspector will contain much more detailed information about HTTP features such as methods, header fields, and status codes, with easy addition of new items through tables. This information will be used to provide a more thorough inspection capability for incorrect or suspicious protocol usage. Detection will benefit from normalization of individual message header field values using algorithms tailored to the specific field. For example a field containing a date will be normalized into a standard date format.

Long-term the goal is to implement SPDY/HTTP 2.0 and other advanced web protocols.

1.6 Binder

The binder collects all the legacy Snort preprocessor network and port settings and config binding settings into one table within each policy. You can now bind by both VLANs and CIDRs instead of one or the other. The criteria are given by binder.when and include policy ID, interfaces, VLANs, networks, protocol, ports, role, and service.

If service inspectors and wizard are configured but no binder is configured, a default binder is created that binds based on the wizard.

1.7 Wizard

The wizard examines the initial payload of TCP and UDP sessions and attempts to determine which service inspector if any should examine the traffic. Use the wizard for portless configurations or to augment port-based configurations to catch otherwise unknown traffic, including command and control channels. The wizard is still under development and should be considered experimental. If you find you need to tweak the defaults please let us know.

2 Getting Started

The following pointers will help you get started:

2.1 Dependencies

Required:

- autotools or cmake to build from source
- g++ >= 4.8 or other recent C++11 compiler
- daq from http://www.snort.org for packet IO
- dnet from http://code.google.com/p/libdnet/ for network utility functions
- LuaJIT from http://luajit.org for configuration and scripting
- pcap from http://www.tcpdump.org for tcpdump style logging
- pcre from http://www.pcre.org for regular expression pattern matching
- zlib from http://www.zlib.net for decompression
- pkgconfig from http://www.freedesktop.org to build the example plugins

Optional:

- 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)
- check from http://check.sourceforge.net to build unit tests

2.2 Building

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

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

- 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 or use ccmake:

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

2.3 Run

First set up the environment:

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

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 -K text -d -e -q -r <pcap>
```

- Verify config, with or w/o rules:

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

2.4 Tips

- Legacy rules and includes must be put in ips.include = *filename* and/or ips.rules = *string* or [[multi line string]].
- You can also use snort2lua to convert your legacy Snort conf file into a Snort++ lua file.
- You can override or add to your lua config with the --lua command line option. You can even add rules to your config via the command line.
- The Lua config is a *live* script that is executed when loaded. You can add functions, grab environment variables, compute values, etc.
- By default, symbols unknown to Snort++ are silently ignored. You can generate warnings for with --warn-unknown. Adding --pedantic makes them fatal. To ignore such symbols, export them in the environment variable SNORT_IGNORE.
- You can also rename symbols that you want to disable. For example, changing normalizer to Xnormalizer will disable the normalizer. This can be easier than commenting in some cases.
- Load plugins from the command line with --plugin-path /path/to/install/lib.
- Snort++ tries hard not to error out too quickly. It will report multiple semantic errors. It always assumes the simplest mode of operation.
- You can process multiple sources at one time by using the -z or --max-threads option.
- Unit tests are configured with --enable-unit-tests (libcheck is required). They can then be run with snort --unit-test [<mode>] where mode is a libcheck print_mode (silent, minimal, normal, etc.).

2.5 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.
```

2.6 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"
```

2.7 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.

3 Basic Modules

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

3.1 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: }

3.2 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 memory for filters { 0: }

- int **alerts.event_filter_memcap** = 1048576: set available memory for filters { 0: }
- int **alerts.flowbits_size** = 1024: maximum number of allowed unique flowbits { 0:2048 }
- string **alerts.order** = pass drop alert log: change the order of rule action application
- int **alerts.rate_filter_memcap** = 1048576: set available memory for 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

3.3 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 }

3.4 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

3.5 daq

What: configure packet acquisition interface

Type: basic

Configuration:

- string **daq.dir**: directory where to search for DAQ plugins
- select **daq.mode**: set mode of operation { passive | inline | read-file }
- bool **daq.no_promisc** = false: whether to put DAQ device into promiscuous mode
- string **daq.type** = pcap: select type of DAQ
- string **daq.var**: list of name=value DAQ-specific parameters
- int **daq.snaplen** = deflt: set snap length (same as -P) { 0:65535 }
- bool **daq.decode_data_link** = false: display the second layer header info

Peg counts:

- **daq.pcaps**: total files 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.fail open**: packets passed during initialization
- **daq.idle**: attempts to acquire from DAQ without available packets

3.6 decode

What: general decoder rules

Type: basic

Rules:

- **116:450** (decode) BAD-TRAFFIC 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) bad traffic loopback IP
- **116:151** (decode) bad traffic same src/dst IP
- **116:467** (decode) too many protocols present

3.7 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.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

3.8 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

3.9 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

3.10 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
- bool **file_id.enable_type** = false: enable type ID
- bool **file_id.enable_signature** = false: enable signature calculation
- int **file_id.show_data_depth** = 100: print this many octets { 0: }

3.11 hosts

What: configure hosts

Type: basic

Configuration:

- addr **hosts[].ip** = 0.0.0.0/32: hosts address / cidr
- enum **hosts[].frag_policy** = linux: defragmentation policy { unknown | first | linux | bsd | bsd_right |last | windows | solaris }
- enum **hosts[].tcp_policy** = linux: tcp reassembly policy { unknown | first | last | bsd | linux | old-linux | windows | win-2003 | vista | solaris | hpux | hpux10 | irix | macos }
- string **hosts[].services[].name**: service identifier
- enum **hosts[].services[].proto** = tcp: ip protocol { tcp | udp }
- port **hosts[].services[].port**: port number

3.12 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** = tap: set policy mode { tap | inline | inline-test }
- string **ips.rules**: snort rules and includes

3.13 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 number of IP6 options Snort will process for a given IPv6 layer. If this limit is hit, rule 116:456 may fire. 0 = unlimited { 0:255 }
- int **network.max_ip_layers** = 0: The maximum number of IP layers Snort will process for a given packet If this limit is hit, rule 116:293 may fire. 0 = unlimited { 0:255 }

3.14 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

- string **output.event_trace.file**: where to write event trace logs
- 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.nolog** = false: turn off logging (alerts still work, same as -N)
- bool **output.obfuscate** = false: obfuscate the logged IP addresses (same as -O)
- 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)

3.15 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
- bool **packets.enable_inline_init_failopen** = true: whether to pass traffic during later stage of initialization to avoid drops
- 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

3.16 ppm

What: packet and rule latency monitoring and control (requires --enable-ppm)

Type: basic

Configuration:

- int **ppm.max_pkt_time** = 0: enable packet latency thresholding (usec), 0 = off { 0: }
- bool **ppm.fastpath_expensive_packets** = false: stop inspection if the max_pkt_time is exceeded
- enum **ppm.pkt_log** = none: log event if max_pkt_time is exceeded { none | log | alert | both }
- bool **ppm.debug_pkts** = false: enable packet debug
- int **ppm.max_rule_time** = 0: enable rule latency thresholding (usec), 0 = off { 0: }
- int **ppm.threshold** = 5: number of times to exceed limit before disabling rule { 1: }
- bool **ppm.suspend_expensive_rules** = false: temporarily disable rule if threshold is reached
- int **ppm.suspend_timeout** = 60: seconds to suspend rule, 0 = permanent { 0: }
- enum **ppm.rule_log** = none: enable event logging for suspended rules { none|log|alert|both }

- bool **ppm.debug_rules** = false: enable rule debug

Rules:

- **134:1** (ppm) rule options disabled by rule latency
- **134:2** (ppm) rule options re-enabled by rule latency
- **134:3** (ppm) packet aborted due to latency

3.17 process

What: configure basic process setup

Type: basic

Configuration:

- string **process.chroot**: set chroot directory (same as -t)
- int **process.threads[].cpu** = 0: pin the associated source/thread to this cpu { 0:127 }
- string **process.threads[].source**: set cpu affinity for this source (either pcap or <iface>
- 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

3.18 profile

What: configure profiling of rules and/or modules (requires --enable-perf-profiling)

Type: basic

Configuration:

- int **profile.rules.count** = -1: print results to given level (-1 = all, 0 = off?) { -1: }
- enum **profile.rules.sort** = avg_ticks: sort by given field { checks | avg_ticks | total_ticks | matches | no_matches | avg_ticks_per_match | avg_ticks_per_no_match }
- int **profile.modules.count** = -1: print results to given level (-1 = all, 0 = off?) { -1: }
- enum **profile.modules.sort** = avg_ticks: sort by given field { checks | avg_ticks | total_ticks }

3.19 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: }
- select **rate_filter[].new_action** = alert: take this action on future hits until timeout { alert | drop | log | pass | | reject | sdrop }
- int **rate_filter[].timeout** = 1: count interval { 0: }
- string **rate_filter[].apply_to**: restrict filter to these addresses according to track

3.20 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

3.21 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

3.22 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
- bool **search_engine.debug_print_fast_pattern** = false: print fast pattern info for each rule
- 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.no_stream_inserts** = false: don't inspect reassembled payload - good for performance, bad for detection
- string **search_engine.search_method** = ac_bnfa_q: set fast pattern algorithm - choose available search engine
- bool **search_engine.split_any_any** = false: evaluate any-any rules separately to save memory
- bool **search_engine.search_optimize** = false: tweak state machine construction for better performance

3.23 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_*
- implied **snort.-B**: <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**: enable daemon restart
- 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
- select **snort.-K** = none: <mode> logging mode { none|text|pcap }
- enum **snort.-k** = all: <mode> checksum mode (all,noip,notcp,noudp,noicmp,none) { all|noip|notcp|noudp|noicmp|none }

- 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: }
- implied **snort.-N**: ignored - for REG_TEST only
- 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**: <n=v> set rules file variable n equal to value v
- int **snort.-s**: <snap> (same as --snaplen) { 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.-w**: dump 802.11 management and control frames
- 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**: <count> maximum number of packet threads (same as --max-packet-threads) { 1: }
- 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
- 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-mode**: <mode> select the DAQ operating mode
- 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) }
- string **snort.--dump-version**: output the version, the whole version, and only the version { (optional) }
- 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**: <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-reset**: ignored - for REG_TEST only

- 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> where to find 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**: <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
- implied **snort.--version**: show version number (same as -V)
- implied **snort.--warn-all**: enable all warnings
- implied **snort.--warn-flowbits**: warn about flowbits that are checked but not set and vice-versa
- implied **snort.--warn-unknown**: warn about unknown symbols in your config
- int **snort.--x2c**: output ASCII char for given hex

Commands:

- **snort.show_plugins**(): show available plugins
- **snort.dump_stats**(): show summary statistics
- **snort.rotate_stats**(): roll perfmonitor log files
- **snort.reload_config**(): load new configuration
- **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

3.24 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

4 Codec Modules

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

4.1 arp

What: support for address resolution protocol

Type: codec

Rules:

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

4.2 auth

What: support for IP authentication header

Type: codec

Rules:

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

4.3 eapol

What: support for extensible authentication protocol over LAN

Type: codec

Rules:

- **116:110** (eapol) truncated EAP header
- **116:111** (eapol) EAP key truncated
- **116:112** (eapol) EAP header truncated

4.4 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 < ERSpan type2 header length

4.5 erspan3

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

Type: codec

Rules:

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

4.6 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

4.7 eth

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

Type: codec

Rules:

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

4.8 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

4.9 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

4.10 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) BAD-TRAFFIC Linux ICMP header DOS attempt
- **116:426** (icmp4) truncated ICMP4 header

Peg counts:

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

4.11 icmp6

What: support for Internet control message protocol v6

Type: codec

Rules:

- **116:427** (icmp6) truncated ICMP6 header
- **116:431** (icmp6) ICMP6 type not decoded
- **116:432** (icmp6) ICMP6 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

4.12 igmp

What: support for Internet group management protocol

Type: codec

Rules:

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

4.13 ipv4

What: support for Internet protocol v4

Type: codec

Rules:

- **116:1** (ipv4) Not IPv4 datagram
- **116:2** (ipv4) hlen < minimum
- **116:3** (ipv4) IP dgm len < IP Hdr len
- **116:4** (ipv4) Ipv4 Options found with bad lengths
- **116:5** (ipv4) Truncated Ipv4 Options

- **116:6** (ipv4) IP dgm len > captured len
- **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) BAD-TRAFFIC IP reserved bit set
- **116:449** (ipv4) BAD-TRAFFIC unassigned/reserved IP protocol
- **116:444** (ipv4) MISC IP option set
- **116:425** (ipv4) truncated IP4 header

Peg counts:

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

4.14 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) IP dgm len < IP Hdr len
- **116:275** (ipv6) IP dgm len > captured len
- **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) BAD-TRAFFIC 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 IP6 extension headers

4.15 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

4.16 pgm

What: support for pragmatic general multicast

Type: codec

Rules:

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

4.17 pppoe

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

Type: codec

Rules:

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

4.18 tcp

What: support for transmission control protocol

Type: codec

Rules:

- **116:45** (tcp) TCP packet len 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) BAD-TRAFFIC TCP port 0 traffic

- **116:402** (tcp) DOS NAPTHA vulnerability detected
- **116:403** (tcp) bad traffic SYN to multicast address

Peg counts:

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

4.19 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) misc large UDP Packet
- **116:447** (udp) BAD-TRAFFIC UDP port 0 traffic

Peg counts:

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

4.20 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

4.21 wlan

What: support for wireless local area network protocol (DLT 105)

Type: codec

Rules:

- **116:133** (wlan) bad 802.11 LLC header
- **116:134** (wlan) bad 802.11 extra LLC info

5 Data Modules

Data modules are adjunct configurations for use with certain inspectors.

5.1 ftp_client

What: FTP client configuration module for use with ftp_server

Type: data

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

5.2 http_global

What: http inspector global configuration and client rules for use with http_server

Type: data

Configuration:

- int **http_global.compress_depth** = 65535: maximum amount of packet payload to decompress { 1:65535 }
- int **http_global.decode.b64_decode_depth** = 0: single packet decode depth { -1:65535 }
- int **http_global.decode.bitenc_decode_depth** = 0: single packet decode depth { -1:65535 }
- int **http_global.decode.max_mime_mem** = 838860: single packet decode depth { 3276: }
- int **http_global.decode.qp_decode_depth** = 0: single packet decode depth { -1:65535 }
- int **http_global.decode.uu_decode_depth** = 0: single packet decode depth { -1:65535 }
- int **http_global.decompress_depth** = 65535: maximum amount of decompressed data to process { 1:65535 }
- bool **http_global.detect_anomalous_servers** = false: inspect non-configured ports for HTTP - bad idea

- int **http_global.max_gzip_mem** = 838860: total memory used for decompression across all active sessions { 3276: }
- int **http_global.memcap** = 150994944: limit of memory used for logging extra data { 2304: }
- bool **http_global.proxy_alert** = false: alert on proxy usage for servers without allow_proxy_use
- int **http_global.unicode_map.code_page** = 1252: select code page in map file { 0: }
- string **http_global.unicode_map.map_file**: unicode map file

Rules:

- **119:1** (http_global) ascii encoding
- **119:2** (http_global) double decoding attack
- **119:3** (http_global) u encoding
- **119:4** (http_global) bare byte unicode encoding
- **119:5** (http_global) base36 encoding
- **119:6** (http_global) UTF-8 encoding
- **119:7** (http_global) IIS unicode codepoint encoding
- **119:8** (http_global) multi_slash encoding
- **119:9** (http_global) IIS backslash evasion
- **119:10** (http_global) self directory traversal
- **119:11** (http_global) directory traversal
- **119:12** (http_global) apache whitespace (tab)
- **119:13** (http_global) non-RFC http delimiter
- **119:14** (http_global) non-RFC defined char
- **119:15** (http_global) oversize request-URI directory
- **119:16** (http_global) oversize chunk encoding
- **119:17** (http_global) unauthorized proxy use detected
- **119:18** (http_global) webroot directory traversal
- **119:19** (http_global) long header
- **119:20** (http_global) max header fields
- **119:21** (http_global) multiple content length
- **119:22** (http_global) chunk size mismatch detected
- **119:23** (http_global) invalid ip in true-client-IP/XFF header
- **119:24** (http_global) multiple host hdrs detected
- **119:25** (http_global) hostname exceeds 255 characters
- **119:26** (http_global) header parsing space saturation
- **119:27** (http_global) client consecutive small chunk sizes
- **119:28** (http_global) post w/o content-length or chunks

- **119:29** (http_global) multiple true IPs in a session
- **119:30** (http_global) both true-client-IP and XFF hdrs present
- **119:31** (http_global) unknown method
- **119:32** (http_global) simple request
- **119:33** (http_global) unescaped space in http URI
- **119:34** (http_global) too many pipelined requests

Peg counts:

- **http_global.packets**: total packets processed
- **http_global.gets**: GET requests
- **http_global.posts**: POST requests
- **http_global.request headers**: total requests
- **http_global.response headers**: total responses
- **http_global.request cookies**: requests with Cookie
- **http_global.response cookies**: responses with Set-Cookie
- **http_global.post params**: POST parameters extracted
- **http_global.unicode**: unicode normalizations
- **http_global.double unicode**: double unicode normalizations
- **http_global.non-ascii**: non-ascii normalizations
- **http_global.paths with ../**: directory traversal normalizations
- **http_global.paths with //**: double slash normalizations
- **http_global.paths with ./**: relative directory normalizations
- **http_global.gzip packets**: packets with gzip compression
- **http_global.compressed bytes**: total comparessed bytes processed
- **http_global.decompressed bytes**: total bytes decompressed

5.3 port_scan_global

What: shared settings for port_scan inspectors for use with port_scan

Type: data

Configuration:

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

Peg counts:

- **port_scan_global.packets**: total packets

6 Inspector Modules

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

6.1 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

6.2 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

6.3 binder

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

Type: inspector

Configuration:

- int **binder[].when.policy_id**: 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 }
- 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 { 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** = defaults to type: symbol name

Peg counts:

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

6.4 ftp_data

What: FTP data channel handler

Type: inspector

Peg counts:

- **ftp_data.packets**: total packets

6.5 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_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

6.6 http_inspect

What: http inspection and server rules; also configure http_inspect

Type: inspector

Configuration:

- bool **http_inspect.allow_proxy_use** = false: don't alert on proxy use for this server
- bool **http_inspect.apache_whitespace** = false: don't alert if tab is used in lieu of space characters
- bool **http_inspect.ascii** = false: enable decoding ASCII like %2f to /
- bool **http_inspect.bare_byte** = false: decode non-standard, non-ASCII character encodings
- int **http_inspect.chunk_length** = 500000: alert on chunk lengths greater than specified { 1: }
- int **http_inspect.client_flow_depth** = 0: raw request payload to inspect { -1:1460 }
- bool **http_inspect.directory** = false: normalize . and .. sequences out of URI
- bool **http_inspect.double_decode** = false: iis specific extra decoding
- bool **http_inspect.enable_cookies** = true: extract cookies
- bool **http_inspect.enable_xff** = false: log True-Client-IP and X-Forwarded-For headers with unified2 alerts as extra data
- bool **http_inspect.extended_ascii_uri** = false: allow extended ASCII codes in the request URI
- bool **http_inspect.extended_response_inspection** = true: extract response headers
- string **http_inspect.http_methods** = GET POST PUT SEARCH MKCOL COPY MOVE LOCK UNLOCK NOTIFY POLL BCOPY BDELETE BMOVE LINK UNLINK OPTIONS HEAD DELETE TRACE TRACK CONNECT SOURCE SUBSCRIBE UNSUBSCRIBE PROPFIND PROPPATCH BPROPFIND BPROPPATCH RPC_CONNECT PROXY_SUCCESS BITS_POST CCM_POST SMS_POST RPC_IN_DATA RPC_OUT_DATA RPC_ECHO_DATA: request methods allowed in addition to GET and POST
- bool **http_inspect.iis_backslash** = false: normalize directory slashes
- bool **http_inspect.iis_delimiter** = false: allow use of non-standard delimiter
- bool **http_inspect.iis_unicode** = false: enable unicode code point mapping using unicode_map settings
- int **http_inspect.iis_unicode_map.code_page** = 1252: select code page in map file { 0: }
- string **http_inspect.iis_unicode_map.map_file**: unicode map file
- bool **http_inspect.inspect_gzip** = true: enable gzip decompression of compressed bodies
- bool **http_inspect.inspect_uri_only** = false: disable all detection except for uricontent
- bool **http_inspect.log_hostname** = false: enable logging of Hostname with unified2 alerts as extra data
- bool **http_inspect.log_uri** = false: enable logging of URI with unified2 alerts as extra data
- int **http_inspect.max_header_length** = 750: maximum allowed client request header field { 0:65535 }
- int **http_inspect.max_headers** = 100: maximum allowed client request headers { 0:1024 }
- int **http_inspect.max_spaces** = 200: maximum allowed whitespaces when folding { 0:65535 }
- bool **http_inspect.multi_slash** = false: normalize out consecutive slashes in URI
- bool **http_inspect.no_pipeline_req** = false: don't inspect pipelined requests after first (still does general detection)

- bit_list **http_inspect.non_rfc_chars** = 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07: alert on given non-RFC chars being present in the URI { 255 }
- bool **http_inspect.non_strict** = true: allows HTTP 0.9 processing
- bool **http_inspect.normalize_cookies** = false: normalize cookies similar to URI
- bool **http_inspect.normalize_headers** = false: normalize headers other than cookie similar to URI
- bool **http_inspect.normalize_javascript** = true: normalize javascript between <script> tags
- int **http_inspect.max_javascript_whitespaces** = 200: maximum number of consecutive whitespaces { 0: }
- bool **http_inspect.normalize_utf** = true: normalize response bodies with UTF content-types
- int **http_inspect.oversize_dir_length** = 500: alert if a URL has a directory longer than this limit { 0: }
- int **http_inspect.post_depth** = 65495: amount of POST data to inspect { -1:65535 }
- enum **http_inspect.profile** = none: set defaults appropriate for selected server { none | all | apache | iis | iis_40 | iis_50 }
- int **http_inspect.server_flow_depth** = 0: response payload to inspect; includes headers with extended_response_inspection { -1:65535 }
- int **http_inspect.small_chunk_count** = 5: alert if more than this limit of consecutive chunks are below small_chunk_length { 0:255 }
- int **http_inspect.small_chunk_length** = 10: alert if more than small_chunk_count consecutive chunks below this limit { 0:255 }
- bool **http_inspect.tab_uri_delimiter** = false: whether a tab not preceded by a space is considered a delimiter or part of URI
- bool **http_inspect.u_encode** = true: decode %uXXXX character sequences
- bool **http_inspect.unlimited_decompress** = true: decompress across multiple packets
- bool **http_inspect.utf_8** = false: decode UTF-8 unicode sequences in URI
- bool **http_inspect.webroot** = false: alert on directory traversals past the top level (web server root)
- bit_list **http_inspect.whitespace_chars**: allowed white space characters { 255 }

Rules:

- **120:1** (http_inspect) anomalous http server on undefined HTTP port
- **120:2** (http_inspect) invalid status code in HTTP response
- **120:3** (http_inspect) no content-length or transfer-encoding in HTTP response
- **120:4** (http_inspect) HTTP response has UTF charset which failed to normalize
- **120:5** (http_inspect) HTTP response has UTF-7 charset
- **120:6** (http_inspect) HTTP response gzip decompression failed
- **120:7** (http_inspect) server consecutive small chunk sizes
- **120:8** (http_inspect) invalid content-length or chunk size
- **120:9** (http_inspect) javascript obfuscation levels exceeds 1
- **120:10** (http_inspect) javascript whitespaces exceeds max allowed
- **120:11** (http_inspect) multiple encodings within javascript obfuscated data

6.7 new_http_inspect

What: new HTTP inspector

Type: inspector

Configuration:

- bool **new_http_inspect.test_input** = false: read HTTP messages from text file
- bool **new_http_inspect.test_output** = false: print out HTTP section data

Rules:

- **219:1** (new_http_inspect) ascii encoding
- **219:2** (new_http_inspect) double decoding attack
- **219:3** (new_http_inspect) u encoding
- **219:4** (new_http_inspect) bare byte unicode encoding
- **219:5** (new_http_inspect) obsolete event—should not appear
- **219:6** (new_http_inspect) UTF-8 encoding
- **219:7** (new_http_inspect) IIS unicode codepoint encoding
- **219:8** (new_http_inspect) multi_slash encoding
- **219:9** (new_http_inspect) IIS backslash evasion
- **219:10** (new_http_inspect) self directory traversal
- **219:11** (new_http_inspect) directory traversal
- **219:12** (new_http_inspect) apache whitespace (tab)
- **219:13** (new_http_inspect) non-RFC http delimiter
- **219:14** (new_http_inspect) non-RFC defined char
- **219:15** (new_http_inspect) oversize request-uri directory
- **219:16** (new_http_inspect) oversize chunk encoding
- **219:17** (new_http_inspect) unauthorized proxy use detected
- **219:18** (new_http_inspect) webroot directory traversal
- **219:19** (new_http_inspect) long header
- **219:20** (new_http_inspect) max header fields
- **219:21** (new_http_inspect) multiple content length
- **219:22** (new_http_inspect) chunk size mismatch detected
- **219:23** (new_http_inspect) invalid IP in true-client-IP/XFF header
- **219:24** (new_http_inspect) multiple host hdrs detected
- **219:25** (new_http_inspect) hostname exceeds 255 characters
- **219:26** (new_http_inspect) header parsing space saturation
- **219:27** (new_http_inspect) client consecutive small chunk sizes

- **219:28** (new_http_inspect) post w/o content-length or chunks
- **219:29** (new_http_inspect) multiple true ips in a session
- **219:30** (new_http_inspect) both true-client-IP and XFF hdrs present
- **219:31** (new_http_inspect) unknown method
- **219:32** (new_http_inspect) simple request
- **219:33** (new_http_inspect) unescaped space in HTTP URI
- **219:34** (new_http_inspect) too many pipelined requests
- **219:35** (new_http_inspect) anomalous http server on undefined HTTP port
- **219:36** (new_http_inspect) invalid status code in HTTP response
- **219:37** (new_http_inspect) no content-length or transfer-encoding in HTTP response
- **219:38** (new_http_inspect) HTTP response has UTF charset which failed to normalize
- **219:39** (new_http_inspect) HTTP response has UTF-7 charset
- **219:40** (new_http_inspect) HTTP response gzip decompression failed
- **219:41** (new_http_inspect) server consecutive small chunk sizes
- **219:42** (new_http_inspect) invalid content-length or chunk size
- **219:43** (new_http_inspect) javascript obfuscation levels exceeds 1
- **219:44** (new_http_inspect) javascript whitespaces exceeds max allowed
- **219:45** (new_http_inspect) multiple encodings within javascript obfuscated data
- **219:46** (new_http_inspect) SWF file zlib decompression failure
- **219:47** (new_http_inspect) SWF file LZMA decompression failure
- **219:48** (new_http_inspect) PDF file deflate decompression failure
- **219:49** (new_http_inspect) PDF file unsupported compression type
- **219:50** (new_http_inspect) PDF file cascaded compression
- **219:51** (new_http_inspect) PDF file parse failure

6.8 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.ips** = false: ensure consistency in retransmitted data
- bool **normalizer.tcp.urp** = true: adjust urgent pointer if beyond segment length
- select **normalizer.tcp.ecn** = off: clear ecn for all packets | sessions w/o ecn setup { off | packet | stream }
- bool **normalizer.tcp.trim** = false: trim data on syn and reset and any outside window or past mss,
- bool **normalizer.tcp.opts** = false: clear all options except mss, wscale, timestamp, and any explicitly allowed
- 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.ip4 tos**: type of service normalizations
- **normalizer.ip4 df**: don't frag bit normalizations
- **normalizer.ip4 rf**: reserved flag bit clears
- **normalizer.ip4 ttl**: time-to-live normalizations
- **normalizer.ip4 opts**: ip4 options cleared
- **normalizer.icmp4 echo**: icmp4 ping normalizations
- **normalizer.ip6 hops**: ip6 hop limit normalizations
- **normalizer.ip6 options**: ip6 options cleared
- **normalizer.icmp6 echo**: icmp6 echo normalizations
- **normalizer.tcp syn options**: SYN only options cleared from non-SYN packets
- **normalizer.tcp ts ecr**: timestamp cleared on non-ACKs
- **normalizer.tcp options**: packets with options cleared
- **normalizer.tcp paddding**: packets with padding cleared
- **normalizer.tcp reserved**: packets with reserved bits cleared
- **normalizer.tcp ecn pkt**: packets with ECN bits cleared
- **normalizer.tcp nonce**: packets with nonce bit cleared
- **normalizer.tcp urgent flag**: packets without urgent flag with urgent pointer cleared
- **normalizer.tcp urgent ptr**: packets without data with urgent poniter cleared
- **normalizer.tcp trim**: tcp segments trimmed to correct size
- **normalizer.tcp ecn session**: ECN bits cleared
- **normalizer.tcp ts nop**: timestamp options cleared
- **normalizer.tcp ips data**: normalized segments
- **normalizer.tcp block**: blocked segments

6.9 perf_monitor

What: performance monitoring and flow statistics collection

Type: inspector

Configuration:

- int **perf_monitor.packets** = 10000: minim packets to report { 0: }
- int **perf_monitor.seconds** = 60: report interval; 0 means report at exit only { 0: }
- int **perf_monitor.flow_ip_memcap** = 52428800: maximum memory for flow tracking { 0: }
- int **perf_monitor.max_file_size** = 4096: files will be rolled over if they exceed this size { 4096: }
- int **perf_monitor.flow_ports** = 1023: maximum ports to track { 0: }
- bool **perf_monitor.reset** = true: reset (clear) statistics after each reporting interval
- bool **perf_monitor.max** = false: calculate theoretical maximum performance
- bool **perf_monitor.console** = false: output to console
- bool **perf_monitor.events** = false: report on qualified vs non-qualified events
- bool **perf_monitor.file** = false: output base stats to perf_monitor.csv instead of stdout
- bool **perf_monitor.flow** = false: enable traffic statistics
- bool **perf_monitor.flow_file** = false: output traffic statistics to a perf_monitor_flow.csv instead of stdout
- bool **perf_monitor.flow_ip** = false: enable statistics on host pairs
- bool **perf_monitor.flow_ip_file** = false: output host pair statistics to perf_monitor_flow_ip.csv instead of stdout

Peg counts:

- **perf_monitor.packets**: total packets

6.10 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

6.11 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

6.12 stream

What: common flow tracking

Type: inspector

Configuration:

- int **stream.icmp_cache.memcap**: maximum cache memory { 0: }
- int **stream.icmp_cache.idle_timeout** = 60: maximum inactive time before retiring session tracker { 1: }
- int **stream.icmp_cache.pruning_timeout** = 30: minimum inactive time before being eligible for pruning { 1: }
- int **stream.icmp_cache.max_sessions** = 262144: maximum simultaneous tcp sessions tracked before pruning { 0: }
- int **stream.ip_cache.memcap**: maximum cache memory { 0: }
- int **stream.ip_cache.idle_timeout** = 60: maximum inactive time before retiring session tracker { 1: }
- int **stream.ip_cache.pruning_timeout** = 30: minimum inactive time before being eligible for pruning { 1: }
- int **stream.ip_cache.max_sessions** = 262144: maximum simultaneous tcp sessions tracked before pruning { 0: }
- int **stream.tcp_cache.memcap**: maximum cache memory { 0: }
- int **stream.tcp_cache.idle_timeout** = 60: maximum inactive time before retiring session tracker { 1: }
- int **stream.tcp_cache.pruning_timeout** = 30: minimum inactive time before being eligible for pruning { 1: }
- int **stream.tcp_cache.max_sessions** = 262144: maximum simultaneous tcp sessions tracked before pruning { 0: }
- int **stream.udp_cache.memcap**: maximum cache memory { 0: }
- int **stream.udp_cache.idle_timeout** = 60: maximum inactive time before retiring session tracker { 1: }
- int **stream.udp_cache.pruning_timeout** = 30: minimum inactive time before being eligible for pruning { 1: }
- int **stream.udp_cache.max_sessions** = 262144: maximum simultaneous tcp sessions tracked before pruning { 0: }

Peg counts:

- **stream.tcp flows**: total tcp sessions
- **stream.tcp prunes**: tcp sessions pruned
- **stream.udp flows**: total udp sessions
- **stream.udp prunes**: udp sessions pruned
- **stream.icmp flows**: total icmp sessions
- **stream.icmp prunes**: icmp sessions pruned
- **stream.ip flows**: total ip sessions
- **stream.ip prunes**: ip sessions pruned

6.13 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.created**: icmp session trackers created
- **stream_icmp.released**: icmp session trackers released

6.14 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 }

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.fragments**: total fragments
- **stream_ip.reassembled**: reassembled datagrams
- **stream_ip.discards**: fragments discarded
- **stream_ip.memory faults**: memory faults
- **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.nodes inserted**: fragments added to tracker
- **stream_ip.nodes deleted**: fragments deleted from tracker

6.15 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 { 1460:63780 }
- enum **stream_tcp.policy** = linux: determines operating system characteristics like reassembly { first | last | linux | old-linux | bsd | macos | solaris | irix | hpux | hpux10 | windows | win-2003 | vista }
- 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 }
- bit_list **stream_tcp.small_segments.ignore_ports**: limit number of small segments queued { 65535 }
- 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 $\Leftarrow$ 0
- **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 sessions
- **stream_tcp.timeouts**: sessions timed out

- **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.trackers created**: tcp session trackers created
- **stream_tcp.trackers released**: tcp session trackers released
- **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.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

6.16 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.created**: udp session trackers created
- **stream_udp.released**: udp session trackers released
- **stream_udp.timeouts**: udp session timeouts

6.17 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

6.18 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 (*)

Peg counts:

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

7 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.

7.1 react

What: send response to client and terminate session

Type: ips_action

Configuration:

- implied **react.msg**: use rule message in response page
- string **react.page**: file containing HTTP response (headers and body)

7.2 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 }

7.3 rewrite

What: overwrite packet contents

Type: ips_action

8 IPS Option Modules

IPS options are the building blocks of IPS rules.

8.1 ack

What: rule option to match on TCP ack numbers

Type: ips_option

Configuration:

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

8.2 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**: <>max | <max | >min
- 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.

8.3 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**: Bytes past start of buffer to start decoding. { 0: }
- implied **base64_decode.relative**: Apply offset to cursor instead of start of buffer.

8.4 bufferlen

What: rule option to check length of current buffer

Type: ips_option

Configuration:

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

8.5 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**: scale extracted value by given amount { 1:65535 }

- int **byte_extract.align**: 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

8.6 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**: scale extracted value by given amount { 1:65535 }
- int **byte_jump.align**: round the number of converted bytes up to the next 2- or 4-byte boundary { 0:4 }
- int **byte_jump.post_offset**: 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

8.7 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

8.8 classtype

What: general rule option for rule classification

Type: ips_option

Configuration:

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

8.9 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**: number of leading characters of this content the fast pattern matcher should exclude
- int **content.fast_pattern_length**: maximum number of characters from this content the fast pattern matcher should use
- 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

8.10 cvs

What: payload rule option for detecting specific attacks

Type: ips_option

Configuration:

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

8.11 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: }

8.12 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*

8.13 file_data

What: rule option to set detection cursor to file data

Type: ips_option

8.14 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

8.15 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

8.16 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

8.17 fragbits

What: rule option to test IP frag flags

Type: ips_option

Configuration:

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

8.18 fragoffset

What: rule option to test IP frag offset

Type: ips_option

Configuration:

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

8.19 gid

What: rule option specifying rule generator

Type: ips_option

Configuration:

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

8.20 http_client_body

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

Type: ips_option

8.21 http_cookie

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

Type: ips_option

8.22 http_header

What: rule option to set the detection cursor to the normalized header(s)

Type: ips_option

Configuration:

- string **http_header.~name**: restrict to given header

8.23 http_method

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

Type: ips_option

8.24 http_raw_cookie

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

Type: ips_option

8.25 http_raw_header

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

Type: ips_option

8.26 http_raw_uri

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

Type: ips_option

8.27 http_stat_code

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

Type: ips_option

8.28 http_stat_msg

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

Type: ips_option

8.29 http_uri

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

Type: ips_option

8.30 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*

8.31 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*

8.32 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*

8.33 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*

8.34 ip_proto

What: rule option to check the IP protocol number

Type: ips_option

Configuration:

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

8.35 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 }

8.36 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

8.37 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*

8.38 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

8.39 msg

What: rule option summarizing rule purpose output with events

Type: ips_option

Configuration:

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

8.40 pcre

What: rule option for matching payload data with regex

Type: ips_option

Configuration:

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

8.41 pkt_data

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

Type: ips_option

8.42 priority

What: rule option for prioritizing events

Type: ips_option

Configuration:

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

8.43 raw_data

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

Type: ips_option

8.44 reference

What: rule option to indicate relevant attack identification system

Type: ips_option

Configuration:

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

8.45 rem

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

Type: ips_option

Configuration:

- string **rem.~**: comment

8.46 replace

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

Type: ips_option

Configuration:

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

8.47 rev

What: rule option to indicate current revision of signature

Type: ips_option

Configuration:

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

8.48 rpc

What: rule option to check SUNRPC CALL parameters

Type: ips_option

Configuration:

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

8.49 seq

What: rule option to check TCP sequence number

Type: ips_option

Configuration:

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

8.50 session

What: rule option to check user data from TCP sessions

Type: ips_option

Configuration:

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

8.51 sid

What: rule option to indicate signature number

Type: ips_option

Configuration:

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

8.52 so

What: rule option to call custom eval function

Type: ips_option

Configuration:

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

8.53 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

8.54 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

8.55 stream_size

What: detection option for stream size checking

Type: ips_option

Configuration:

- enum **stream_size.direction**: compare applies to the given direction(s) { either|client|server|both }
- enum **stream_size.operator**: how to compare { = | != | < | > | ⇐ | >= }
- int **stream_size.size**: size for comparison

8.56 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: }

8.57 tos

What: rule option to check type of service field

Type: ips_option

Configuration:

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

8.58 ttl

What: rule option to check time to live field

Type: ips_option

Configuration:

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

8.59 window

What: rule option to check TCP window field

Type: ips_option

Configuration:

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

9 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.

10 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.

11 Logger Modules

All output of events and packets is done by Loggers.

11.1 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.csv** = timestamp gid sid rev src_addr src_port dst_addr dst_port: selected fields will be output in given order left to right { timestamp | gid | sid | rev | msg | proto | src_addr | dst_addr | src_port | dst_port | eth_src | eth_dst | eth_type | eth_len | ttl | tos | id | ip_len | dgm_len | icmp_type | icmp_code | icmp_id | icmp_seq | tcp_flags | tcp_seq | tcp_ack | tcp_len | tcp_win | udp_len }
- int **alert_csv.limit** = 0: set limit (0 is unlimited) { 0: }
- enum **alert_csv.units** = B: bytes | KB | MB | GB { B | K | M | G }

11.2 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 }

11.3 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 }

11.4 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 }

11.5 alert_test

What: output event in custom tsv format

Type: logger

Configuration:

- bool **alert_test.file** = false: output to alert_test.txt instead of stdout
- bool **alert_test.rebuilt** = false: include type:count where type is S for stream and F for frag
- bool **alert_test.session** = false: include src-dst each of form -addr:port
- bool **alert_test.msg** = false: include alert msg

11.6 alert_unixsock

What: output event over unix socket

Type: logger

11.7 log_codecs

What: log protocols in packet by layer

Type: logger

Configuration:

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

11.8 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 }

11.9 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

12 Snort++ vs Snort

Snort++ differs from Snort 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 config options are grouped into Snort++ modules

12.1 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

12.2 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 -K text; -K text/pcap is old dump/log mode
- 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

12.3 Conf File

- Snort++ has a default unicode.map
- Snort++ will not enforce an upper bound on memcaps and the like within 64 bits
- Snort++ will supply a default *_global config if not specified (Snort 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

12.4 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

12.5 Output

- alert_fast includes packet data by default
- all text mode outputs default to stdout
- changed default logging mode to -K none
- deleted layer2resets and flexresp2_*
- deleted log_ascii
- general output guideline: don't print zero counts
- Snort++ 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 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

13 Snort2Lua

Snort2Lua reads your legacy Snort conf file(s) and generates Snort++ Lua and rules files. When running this program, the only mandatory option is to provide Snort2Lua with a Snort 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 Snort++ 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++ does not support those preprocessors. Additionally, any rule options associated with those preprocessors are also not supported. Finally, Snort2Lua expects a valid Snort configuration. Therefore, if the configuration is invalid or has questionable syntax, Snort2Lua may find those errors and fail to parse the 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++ 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.

13.1 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 and Snort++ 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++ configuration. Instead, you can see the exact options from the input configuration that have changed.

13.1.1 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** print all data, including errors and Snort - Snort++ configuration differences.
- **-c <snort_conf>** The Snort <snort_conf> file to convert
- **-d** print the differences, and only the differences, between the Snort and Snort++ configurations
- **-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
- **-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
- **--output-file=<out_file>** Same as *-o*. output the new Snort++ lua configuration to <out_file>
- **--print-all** Same as *-a*. print all data, including errors, Snort - Snort++ configuration differences, and developer warnings.
- **--print-differences** Same as *-d*. print the differences, and only the differences, between the Snort and Snort++ configurations
- **--quiet** Same as *-q*. quiet mode. Only output valid confiration information
- **--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.

13.2 Known Problems

- Any Snort '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.

14 Reference

14.1 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 legacy Snort 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.

14.2 Optional Features

Listed below are the features that must be explicitly enabled so they are built into the Snort binary. For a full list of build features, run ./configure --help.

- **--enable-ppm**: enable packet and rule performance monitoring and coarse latency enforcement.
- **--enable-perf-profiling**: enable module and rule performance profiling.
- **--enable-shell**: enable local and remote command line shell support.

14.3 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.

14.4 Command Line Options

- **--alert-before-pass** process alert, drop, sdrop, or reject before pass; default is pass before alert, drop,...
- **--bpf** <filter options> are standard BPF options, as seen in TCPDump
- **--c2x** output hex for given char
- **--create-pidfile** create PID file, even when not in Daemon mode
- **--daq** <type> select packet acquisition module (default is pcap)
- **--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-mode** <mode> select the DAQ operating mode
- **--daq-var** <name=value> specify extra DAQ configuration variable
- **--dirty-pig** don't flush packets on shutdown
- **--dump-builtin-rules** [<module prefix>] output stub rules for selected modules
- **--dump-defaults** [<module prefix>] output module defaults in Lua format
- **--dump-dynamic-rules** output stub rules for all loaded rules libraries
- **--dump-version** output the version, the whole version, and only the version
- **--enable-inline-test** enable Inline-Test Mode Operation
- **--help** list command line options
- **--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-options** <option prefix> output matching command line option quick help (same as -?)
- **--help-plugins** list all available plugins with brief help
- **--help-signals** dump available control signals
- **--id-subdir** create/use instance subdirectories in logdir instead of instance filename prefix
- **--id-zero** use id prefix / subdirectory even with one packet thread
- **--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 of given type
- **--list-plugins** list all known plugins
- **--logid** <0xid> log Identifier to uniquely id events for multiple snorts (same as -G)
- **--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)
- **--nolock-pidfile** do not try to lock Snort PID file
- **--nostamps** don't include timestamps in log file names
- **--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
- **--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-reset** ignored - for REG_TEST only
- **--pcap-show** print a line saying what pcap is currently being read
- **--pedantic** warnings are fatal
- **--plugin-path** <path> where to find plugins
- **--process-all-events** process all action groups
- **--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> where to find luajit scripts
- **--shell** enable the interactive command line
- **--show-plugins** list module and plugin versions
- **--skip** <n> skip 1st n packets
- **--snaplen** <snap> set snaplen of packet (same as -s)
- **--stdin-rules** read rules from stdin until EOF or a line starting with END is read
- **--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
- **--version** show version number (same as -V)
- **--warn-all** enable all warnings
- **--warn-flowbits** warn about flowbits that are checked but not set and vice-versa
- **--warn-unknown** warn about unknown symbols in your config
- **--x2c** output ASCII char for given hex
- **-?** <option prefix> output matching command line option quick help (same as --help-options)
- **-A** <mode> set alert mode: none, cmg, or alert_*

- **-B** <mask> obfuscated IP addresses in alerts and packet dumps using CIDR mask
- **-C** print out payloads with character data only (no hex)
- **-D** run Snort in background (daemon) mode
- **-E** enable daemon restart
- **-G** <0xid> (same as --logid)
- **-H** make hash tables deterministic
- **-K** <mode> logging mode
- **-M** log messages to syslog (not alerts)
- **-N** ignored - for REG_TEST only
- **-O** obfuscate the logged IP addresses
- **-Q** enable inline mode operation
- **-R** <rules> include this rules file in the default policy
- **-S** <n=v> set rules file variable n equal to value v
- **-T** test and report on the current Snort configuration
- **-U** use UTC for timestamps
- **-V** (same as --version)
- **-W** lists available interfaces
- **-X** dump the raw packet data starting at the link layer
- **-c** <conf> use this configuration
- **-d** dump the Application Layer
- **-e** display the second layer header info
- **-f** turn off fflush() calls after binary log writes
- **-g** <gname> run snort gid as <gname> group (or gid) after initialization
- **-i** <iface>… list of interfaces
- **-j** <port> to listen for telnet connections
- **-k** <mode> checksum mode (all,noip,notcp,noudp,noicmp,none)
- **-l** <logdir> log to this directory instead of current directory
- **-m** <umask> set umask = <umask>
- **-n** <count> stop after count packets
- **-q** quiet mode - Don’t show banner and status report
- **-r** <pcap>… (same as --pcap-list)
- **-s** <snap> (same as --snaplen)
- **-t** <dir> chroots process to <dir> after initialization
- **-u** <uname> run snort as <uname> or <uid> after initialization
- **-v** be verbose

- **-w** dump 802.11 management and control frames
- **-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)

14.5 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
- **enum**: a string selected from the given range
- **implied**: a 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

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.

14.6 Configuration

- string **ack.~range**: check if packet payload size is *size* | *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.csv** = timestamp gid sid rev src_addr src_port dst_addr dst_port: selected fields will be output in given order left to right { timestamp | gid | sid | rev | msg | proto | src_addr | dst_addr | src_port | dst_port | eth_src | eth_dst | eth_type | eth_len | ttl | tos | id | ip_len | dgm_len | icmp_type | icmp_code | icmp_id | icmp_seq | tcp_flags | tcp_seq | tcp_ack | tcp_len | tcp_win | 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: }
- 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 }
- 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 }
- bool **alert_test.file** = false: output to alert_test.txt instead of stdout
- bool **alert_test.msg** = false: include alert msg
- bool **alert_test.rebuilt** = false: include type:count where type is S for stream and F for frag
- bool **alert_test.session** = false: include src-dst each of form -addr:port
- 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 memory for filters { 0: }
- int **alerts.event_filter_memcap** = 1048576: set available memory for filters { 0: }
- int **alerts.flowbits_size** = 1024: maximum number of allowed unique flowbits { 0:2048 }
- string **alerts.order** = pass drop alert log: change the order of rule action application

- int **alerts.rate_filter_memcap** = 1048576: set available memory for 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
- 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**: <>max | <max | >min
- 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**: 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 { block | allow | inspect }
- string **binder[].use.file**: use configuration in given file
- string **binder[].use.name** = defaults to type: symbol name
- 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**: 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 }
- 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**: 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.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**: scale extracted value by given amount { 1:65535 }
- implied **byte_extract.oct**: convert from octal string
- implied **byte_extract.relative**: offset from cursor instead of start of buffer
- implied **byte_extract.string**: convert from string
- int **byte_extract.~count**: number of bytes to pick up from the buffer { 1:10 }
- string **byte_extract.~name**: name of the variable that will be used in other rule options
- int **byte_extract.~offset**: number of bytes into the buffer to start processing { -65535:65535 }
- int **byte_jump.align**: round the number of converted bytes up to the next 2- or 4-byte boundary { 0:4 }
- implied **byte_jump.big**: big endian
- 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**: scale extracted value by given amount { 1:65535 }
- implied **byte_jump.oct**: convert from octal string
- int **byte_jump.post_offset**: 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
- 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_test.big**: big endian
- 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
- implied **byte_test.relative**: offset from cursor instead of start of buffer
- implied **byte_test.string**: convert from string
- 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 }

- 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
- 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.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
- implied **content.fast_pattern**: use this content in the fast pattern matcher instead of the content selected by default
- int **content.fast_pattern_length**: maximum number of characters from this content the fast pattern matcher should use
- int **content.fast_pattern_offset**: number of leading characters of this content the fast pattern matcher should exclude
- 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
- string **content.~data**: data to match
- implied **cvs.invalid-entry**: looks for an invalid Entry string
- bool **daq.decode_data_link** = false: display the second layer header info
- string **daq.dir**: directory where to search for DAQ plugins
- select **daq.mode**: set mode of operation { passive | inline | read-file }
- bool **daq.no_promisc** = false: whether to put DAQ device into promiscuous mode
- int **daq.snaplen** = deflt: set snap length (same as -P) { 0:65535 }
- string **daq.type** = pcap: select type of DAQ
- string **daq.var**: list of name=value DAQ-specific parameters
- 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 }
- 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 }
- 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
- 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
- bool **file_id.enable_signature** = false: enable signature calculation
- bool **file_id.enable_type** = false: enable type ID
- int **file_id.lookup_timeout** = 2: give up on lookup after this many seconds { 0: }
- int **file_id.show_data_depth** = 100: print this many octets { 0: }
- int **file_id.signature_depth** = 10485760: stop signature at this point { 0: }
- int **file_id.type_depth** = 1460: stop type ID at this point { 0: }
- string **flags.~mask_flags**: these flags are don't cares
- string **flags.~test_flags**: these flags are tested
- 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 **flowbits.~arg1**: bits or group
- string **flowbits.~arg2**: group if arg1 is bits
- string **flowbits.~command**: set|reset|isset|etc.
- string **fragbits.~flags**: these flags are tested
- string **fragoffset.~range**: check if packet payload size is *size* | *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_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: }
- enum **hosts[].frag_policy** = linux: defragmentation policy { unknown | 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** = linux: tcp reassembly policy { unknown | first | last | bsd | linux | old-linux | windows | win-2003 | vista | solaris | hpux | hpux10 | irix | macos }
- int **http_global.compress_depth** = 65535: maximum amount of packet payload to decompress { 1:65535 }

- int **http_global.decode.b64_decode_depth** = 0: single packet decode depth { -1:65535 }
- int **http_global.decode.bitenc_decode_depth** = 0: single packet decode depth { -1:65535 }
- int **http_global.decode.max_mime_mem** = 838860: single packet decode depth { 3276: }
- int **http_global.decode.qp_decode_depth** = 0: single packet decode depth { -1:65535 }
- int **http_global.decode.uu_decode_depth** = 0: single packet decode depth { -1:65535 }
- int **http_global.decompress_depth** = 65535: maximum amount of decompressed data to process { 1:65535 }
- bool **http_global.detect_anomalous_servers** = false: inspect non-configured ports for HTTP - bad idea
- int **http_global.max_gzip_mem** = 838860: total memory used for decompression across all active sessions { 3276: }
- int **http_global.memcap** = 150994944: limit of memory used for logging extra data { 2304: }
- bool **http_global.proxy_alert** = false: alert on proxy usage for servers without allow_proxy_use
- int **http_global.unicode_map.code_page** = 1252: select code page in map file { 0: }
- string **http_global.unicode_map.map_file**: unicode map file
- string **http_header.~name**: restrict to given header
- bool **http_inspect.allow_proxy_use** = false: don't alert on proxy use for this server
- bool **http_inspect.apache_whitespace** = false: don't alert if tab is used in lieu of space characters
- bool **http_inspect.ascii** = false: enable decoding ASCII like %2f to /
- bool **http_inspect.bare_byte** = false: decode non-standard, non-ASCII character encodings
- int **http_inspect.chunk_length** = 500000: alert on chunk lengths greater than specified { 1: }
- int **http_inspect.client_flow_depth** = 0: raw request payload to inspect { -1:1460 }
- bool **http_inspect.directory** = false: normalize . and .. sequences out of URI
- bool **http_inspect.double_decode** = false: iis specific extra decoding
- bool **http_inspect.enable_cookies** = true: extract cookies
- bool **http_inspect.enable_xff** = false: log True-Client-IP and X-Forwarded-For headers with unified2 alerts as extra data
- bool **http_inspect.extended_ascii_uri** = false: allow extended ASCII codes in the request URI
- bool **http_inspect.extended_response_inspection** = true: extract response headers
- string **http_inspect.http_methods** = GET POST PUT SEARCH MKCOL COPY MOVE LOCK UNLOCK NOTIFY POLL BCOPY BDELETE BMOVE LINK UNLINK OPTIONS HEAD DELETE TRACE TRACK CONNECT SOURCE SUBSCRIBE UNSUBSCRIBE PROPFIND PROPPATCH BPROPFIND BPROPPATCH RPC_CONNECT PROXY_SUCCESS BITS_POST CCM_POST SMS_POST RPC_IN_DATA RPC_OUT_DATA RPC_ECHO_DATA: request methods allowed in addition to GET and POST
- bool **http_inspect.iis_backslash** = false: normalize directory slashes
- bool **http_inspect.iis_delimiter** = false: allow use of non-standard delimiter
- bool **http_inspect.iis_unicode** = false: enable unicode code point mapping using unicode_map settings
- int **http_inspect.iis_unicode_map.code_page** = 1252: select code page in map file { 0: }
- string **http_inspect.iis_unicode_map.map_file**: unicode map file
- bool **http_inspect.inspect_gzip** = true: enable gzip decompression of compressed bodies

- bool **http_inspect.inspect_uri_only** = false: disable all detection except for uricontent
- bool **http_inspect.log_hostname** = false: enable logging of Hostname with unified2 alerts as extra data
- bool **http_inspect.log_uri** = false: enable logging of URI with unified2 alerts as extra data
- int **http_inspect.max_header_length** = 750: maximum allowed client request header field { 0:65535 }
- int **http_inspect.max_headers** = 100: maximum allowed client request headers { 0:1024 }
- int **http_inspect.max_javascript_whitespaces** = 200: maximum number of consecutive whitespaces { 0: }
- int **http_inspect.max_spaces** = 200: maximum allowed whitespaces when folding { 0:65535 }
- bool **http_inspect.multi_slash** = false: normalize out consecutive slashes in URI
- bool **http_inspect.no_pipeline_req** = false: don't inspect pipelined requests after first (still does general detection)
- bit_list **http_inspect.non_rfc_chars** = 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07: alert on given non-RFC chars being present in the URI { 255 }
- bool **http_inspect.non_strict** = true: allows HTTP 0.9 processing
- bool **http_inspect.normalize_cookies** = false: normalize cookies similar to URI
- bool **http_inspect.normalize_headers** = false: normalize headers other than cookie similar to URI
- bool **http_inspect.normalize_javascript** = true: normalize javascript between <script> tags
- bool **http_inspect.normalize_utf** = true: normalize response bodies with UTF content-types
- int **http_inspect.oversize_dir_length** = 500: alert if a URL has a directory longer than this limit { 0: }
- int **http_inspect.post_depth** = 65495: amount of POST data to inspect { -1:65535 }
- enum **http_inspect.profile** = none: set defaults appropriate for selected server { none | all | apache | iis | iis_40 | iis_50 }
- int **http_inspect.server_flow_depth** = 0: response payload to inspect; includes headers with extended_response_inspection { -1:65535 }
- int **http_inspect.small_chunk_count** = 5: alert if more than this limit of consecutive chunks are below small_chunk_length { 0:255 }
- int **http_inspect.small_chunk_length** = 10: alert if more than small_chunk_count consecutive chunks below this limit { 0:255 }
- bool **http_inspect.tab_uri_delimiter** = false: whether a tab not preceded by a space is considered a delimiter or part of URI
- bool **http_inspect.u_encode** = true: decode %uXXXX character sequences
- bool **http_inspect.unlimited_decompress** = true: decompress across multiple packets
- bool **http_inspect.utf_8** = false: decode UTF-8 unicode sequences in URI
- bool **http_inspect.webroot** = false: alert on directory traversals past the top level (web server root)
- bit_list **http_inspect.whitespace_chars**: allowed white space characters { 255 }
- 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*
- string **ip_proto.~proto**: [!|>|<] name or number

- select **ipopts.~opt**: output format { rr|eol|nop|ts|sec|esec|lsrr|lsrre|ssrr|satid|any }
- 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** = tap: set policy mode { tap | inline | inline-test }
- string **ips.rules**: snort rules and includes
- implied **isdataat.relative**: offset from cursor instead of start of buffer
- string **isdataat.~length**: num | !num
- string **itype.~range**: check if icmp type is *type* | *min*<>*max* | <*max* | >*min*
- bool **log_codecs.file** = stdout: output to log_codecs.txt instead of stdout
- bool **log_codecs.msg** = false: include alert msg
- 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 **metadata.***: additional parameters not used by snort
- string **metadata.service**: service name
- 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 number of IP6 options Snort will process for a given IPv6 layer. If this limit is hit, rule 116:456 may fire. 0 = unlimited { 0:255 }
- int **network.max_ip_layers** = 0: The maximum number of IP layers Snort will process for a given packet If this limit is hit, rule 116:293 may fire. 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 **new_http_inspect.test_input** = false: read HTTP messages from text file
- bool **new_http_inspect.test_output** = false: print out HTTP section data
- 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
- 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** = false: clear all options except mss, wscale, timestamp, and any explicitly allowed
- bool **normalizer.tcp.trim** = false: trim data on syn and reset and any outside window or past mss,
- 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)
- string **output.event_trace.file**: where to write event trace logs
- int **output.event_trace.max_data** = 0: maximum amount of packet data to capture { 0:65535 }
- bool **output.log_ipv6_extra_data** = false: log IPv6 source and destination addresses as unified2 extra data records
- string **output.logdir** = .: where to put log files (same as -l)
- bool **output.nolog** = false: turn off logging (alerts still work, same as -N)
- bool **output.obfuscate** = false: obfuscate the logged IP addresses (same as -O)
- 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 **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
- bool **packets.enable_inline_init_failopen** = true: whether to pass traffic during later stage of initialization to avoid drops
- 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.~regex**: Snort regular expression
- bool **perf_monitor.console** = false: output to console
- bool **perf_monitor.events** = false: report on qualified vs non-qualified events
- bool **perf_monitor.file** = false: output base stats to perf_monitor.csv instead of stdout
- bool **perf_monitor.flow** = false: enable traffic statistics
- bool **perf_monitor.flow_file** = false: output traffic statistics to a perf_monitor_flow.csv instead of stdout
- bool **perf_monitor.flow_ip** = false: enable statistics on host pairs
- bool **perf_monitor.flow_ip_file** = false: output host pair statistics to perf_monitor_flow_ip.csv instead of stdout
- int **perf_monitor.flow_ip_memcap** = 52428800: maximum memory for flow tracking { 0: }
- int **perf_monitor.flow_ports** = 1023: maximum ports to track { 0: }
- bool **perf_monitor.max** = false: calculate theoretical maximum performance
- int **perf_monitor.max_file_size** = 4096: files will be rolled over if they exceed this size { 4096: }
- int **perf_monitor.packets** = 10000: minim packets to report { 0: }
- bool **perf_monitor.reset** = true: reset (clear) statistics after each reporting interval
- int **perf_monitor.seconds** = 60: report interval; 0 means report at exit only { 0: }
- 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 **port_scan_global.memcap** = 1048576: maximum tracker memory { 1: }
- bool **ppm.debug_pkts** = false: enable packet debug
- bool **ppm.debug_rules** = false: enable rule debug
- bool **ppm.fastpath_expensive_packets** = false: stop inspection if the max_pkt_time is exceeded
- int **ppm.max_pkt_time** = 0: enable packet latency thresholding (usec), 0 = off { 0: }
- int **ppm.max_rule_time** = 0: enable rule latency thresholding (usec), 0 = off { 0: }
- enum **ppm.pkt_log** = none: log event if max_pkt_time is exceeded { none | log | alert | both }
- enum **ppm.rule_log** = none: enable event logging for suspended rules { none|log|alert|both }
- bool **ppm.suspend_expensive_rules** = false: temporarily disable rule if threshold is reached
- int **ppm.suspend_timeout** = 60: seconds to suspend rule, 0 = permanent { 0: }
- int **ppm.threshold** = 5: number of times to exceed limit before disabling rule { 1: }

- 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)
- int **process.threads[].cpu** = 0: pin the associated source/thread to this cpu { 0:127 }
- string **process.threads[].source**: set cpu affinity for this source (either pcap or <iface>
- 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 **profile.modules.count** = -1: print results to given level (-1 = all, 0 = off?) { -1: }
- enum **profile.modules.sort** = avg_ticks: sort by given field { checks | avg_ticks | total_ticks }
- int **profile.rules.count** = -1: print results to given level (-1 = all, 0 = off?) { -1: }
- enum **profile.rules.sort** = avg_ticks: sort by given field { checks | avg_ticks | total_ticks | matches | no_matches | avg_ticks_per_match | avg_ticks_per_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: }
- select **rate_filter[].new_action** = alert: take this action on future hits until timeout { alert | drop | log | pass | | reject | sdrop }
- 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 }
- implied **react.msg**: use rule message in response page
- 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
- 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
- int **rev.~**: revision { 1: }

- string **rpc.~app**: application number
- string **rpc.~proc**: procedure number or * for any
- string **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: }
- 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_fast_pattern** = false: print fast pattern info for each rule
- 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
- 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.no_stream_inserts** = false: don't inspect reassembled payload - good for performance, bad for detection
- string **search_engine.search_method** = ac_bnfa_q: set fast pattern algorithm - choose available search engine
- bool **search_engine.search_optimize** = false: tweak state machine construction for better performance
- bool **search_engine.split_any_any** = false: evaluate any-any rules separately to save memory
- string **seq.~range**: check if packet payload size is *size* | *min*<>*max* | <*max* | >*min*
- enum **session.~mode**: output format { printable|binary|all }
- int **sid.~**: signature id { 1: }
- 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
- 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-mode**: <mode> select the DAQ operating mode
- 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

- 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
- string **snort.--dump-version**: output the version, the whole version, and only the version { (optional) }
- 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
- 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**: <count> configure maximum number of packet threads (same as -z) { 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.--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-reset**: ignored - for REG_TEST only

- 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> where to find 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**: <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
- implied **snort.--version**: show version number (same as -V)
- implied **snort.--warn-all**: enable all warnings
- implied **snort.--warn-flowbits**: warn about flowbits that are checked but not set and vice-versa
- implied **snort.--warn-unknown**: warn about unknown symbols in your config
- int **snort.--x2c**: output ASCII char for given hex
- 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_*
- implied **snort.-B**: <mask> obfuscated IP addresses in alerts and packet dumps using CIDR mask
- implied **snort.-C**: print out payloads with character data only (no hex)
- implied **snort.-D**: run Snort in background (daemon) mode
- implied **snort.-E**: enable daemon restart
- int **snort.-G**: <0xid> (same as --logid) { 0:65535 }
- implied **snort.-H**: make hash tables deterministic
- select **snort.-K** = none: <mode> logging mode { none|text|pcap }
- implied **snort.-M**: log messages to syslog (not alerts)
- implied **snort.-N**: ignored - for REG_TEST only
- implied **snort.-O**: obfuscate the logged IP addresses
- implied **snort.-Q**: enable inline mode operation
- string **snort.-R**: <rules> include this rules file in the default policy

- string **snort.-S**: <n=v> set rules file variable n equal to value v
- implied **snort.-T**: test and report on the current Snort configuration
- implied **snort.-U**: use UTC for timestamps
- implied **snort.-V**: (same as --version)
- implied **snort.-W**: lists available interfaces
- implied **snort.-X**: dump the raw packet data starting at the link layer
- string **snort.-c**: <conf> use this configuration
- 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
- string **snort.-g**: <gname> run snort gid as <gname> group (or gid) after initialization
- string **snort.-i**: <iface>... list of interfaces
- port **snort.-j**: <port> to listen for telnet connections
- enum **snort.-k** = all: <mode> checksum mode (all,noip,notcp,noudp,noicmp,none) { all|noip|notcp|noudp|noicmp|none }
- string **snort.-l**: <logdir> log to this directory instead of current directory
- int **snort.-m**: <umask> set umask = <umask> { 0: }
- int **snort.-n**: <count> stop after count packets { 0: }
- implied **snort.-q**: quiet mode - Don't show banner and status report
- string **snort.-r**: <pcap>... (same as --pcap-list)
- int **snort.-s**: <snap> (same as --snaplen) { 68:65535 }
- string **snort.-t**: <dir> chroots process to <dir> after initialization
- string **snort.-u**: <uname> run snort as <uname> or <uid> after initialization
- implied **snort.-v**: be verbose
- implied **snort.-w**: dump 802.11 management and control frames
- implied **snort.-x**: same as --pedantic
- implied **snort.-y**: include year in timestamp in the alert and log files
- int **snort.-z**: <count> maximum number of packet threads (same as --max-packet-threads) { 1: }
- string **so.~func**: name of eval function
- string **soid.~**: SO rule ID has <gid>|<sid> format, like 3|12345
- int **stream.icmp_cache.idle_timeout** = 60: maximum inactive time before retiring session tracker { 1: }
- int **stream.icmp_cache.max_sessions** = 262144: maximum simultaneous tcp sessions tracked before pruning { 0: }
- int **stream.icmp_cache.memcap**: maximum cache memory { 0: }
- int **stream.icmp_cache.pruning_timeout** = 30: minimum inactive time before being eligible for pruning { 1: }
- int **stream.ip_cache.idle_timeout** = 60: maximum inactive time before retiring session tracker { 1: }
- int **stream.ip_cache.max_sessions** = 262144: maximum simultaneous tcp sessions tracked before pruning { 0: }

- int **stream.ip_cache.memcap**: maximum cache memory { 0: }
- int **stream.ip_cache.pruning_timeout** = 30: minimum inactive time before being eligible for pruning { 1: }
- int **stream.tcp_cache.idle_timeout** = 60: maximum inactive time before retiring session tracker { 1: }
- int **stream.tcp_cache.max_sessions** = 262144: maximum simultaneous tcp sessions tracked before pruning { 0: }
- int **stream.tcp_cache.memcap**: maximum cache memory { 0: }
- int **stream.tcp_cache.pruning_timeout** = 30: minimum inactive time before being eligible for pruning { 1: }
- int **stream.udp_cache.idle_timeout** = 60: maximum inactive time before retiring session tracker { 1: }
- int **stream.udp_cache.max_sessions** = 262144: maximum simultaneous tcp sessions tracked before pruning { 0: }
- int **stream.udp_cache.memcap**: maximum cache memory { 0: }
- int **stream.udp_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.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 }
- 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|client|server|both }
- enum **stream_size.operator**: how to compare { = | != | < | > | <= | >= }
- int **stream_size.size**: size for comparison
- 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 { 1460:63780 }
- 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** = linux: determines operating system characteristics like reassembly { first | last | linux | old-linux | bsd | macos | solaris | irix | hpux | hpux10 | windows | win-2003 | vista }
- 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 }
- bit_list **stream_tcp.small_segments.ignore_ports**: limit number of small segments queued { 65535 }
- int **stream_tcp.small_segments.maximum_size** = 0: limit number of small segments queued { 0:2048 }
- 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 **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: }
- int **tag.packets**: tag this many packets { 1: }
- int **tag.seconds**: tag for this many seconds { 1: }
- enum **tag.~**: log all packets in session or all packets to or from host { session|host_src|host_dst }
- 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 packet payload size is *size* | *min*<>*max* | <*max* | >*min*
- string **ttl.~range**: check if packet payload size is *size* | *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 packet payload size is *size* | *min*<>*max* | <*max* | >*min*
- 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 (*)

14.7 Counts

- **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
- **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.fail open**: packets passed during initialization
- **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 processed
- **daq.received**: total packets received from DAQ
- **daq.replace**: total replace verdicts
- **daq.skipped**: packets skipped at startup
- **daq.whitelist**: total whitelist verdicts
- **detection.alert limit**: events previously triggered on same PDU
- **detection.alerts**: alerts not including IP reputation

- **detection.analyzed**: packets sent to detection
- **detection.event limit**: events filtered
- **detection.log limit**: events queued but not logged
- **detection.logged**: logged packets
- **detection.match limit**: fast pattern matches not processed
- **detection.passed**: passed packets
- **detection.queue limit**: events not queued because queue full
- **detection.total alerts**: alerts including IP reputation
- **ftp_data.packets**: total packets
- **ftp_server.packets**: total packets
- **http_global.compressed bytes**: total comparessed bytes processed
- **http_global.decompressed bytes**: total bytes decompressed
- **http_global.double unicode**: double unicode normalizations
- **http_global.gets**: GET requests
- **http_global.gzip packets**: packets with gzip compression
- **http_global.non-ascii**: non-ascii normalizations
- **http_global.packets**: total packets processed
- **http_global.paths with ../**: directory traversal normalizations
- **http_global.paths with ./**: relative directory normalizations
- **http_global.paths with //**: double slash normalizations
- **http_global.post params**: POST parameters extracted
- **http_global.posts**: POST requests
- **http_global.request cookies**: requests with Cookie
- **http_global.request headers**: total requests
- **http_global.response cookies**: responses with Set-Cookie
- **http_global.response headers**: total responses
- **http_global.unicode**: unicode normalizations
- **icmp4.bad checksum**: non-zero icmp checksums
- **icmp6.bad checksum (ip4)**: nonzero ipcm4 checksums
- **icmp6.bad checksum (ip6)**: nonzero ipcm6 checksums
- **ipv4.bad checksum**: nonzero ip checksums
- **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 reserved**: packets with reserved bits cleared
- **normalizer.tcp syn options**: SYN only options cleared from non-SYN packets
- **normalizer.tcp trim**: tcp segments trimmed to correct size
- **normalizer.tcp ts ecr**: timestamp cleared on non-ACKs
- **normalizer.tcp ts nop**: timestamp options cleared
- **normalizer.tcp urgent flag**: packets without urgent flag with urgent pointer cleared
- **normalizer.tcp urgent ptr**: packets without data with urgent poniter cleared
- **perf_monitor.packets**: total packets
- **port_scan_global.packets**: total packets
- **rpc_decode.packets**: total packets
- **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
- **stream.icmp flows**: total icmp sessions
- **stream.icmp prunes**: icmp sessions pruned
- **stream.ip flows**: total ip sessions
- **stream.ip prunes**: ip sessions pruned
- **stream.tcp flows**: total tcp sessions
- **stream.tcp prunes**: tcp sessions pruned

- **stream.udp flows**: total udp sessions
- **stream.udp prunes**: udp sessions pruned
- **stream_icmp.created**: icmp session trackers created
- **stream_icmp.released**: icmp session trackers released
- **stream_ip.alerts**: alerts generated
- **stream_ip.anomalies**: anomalies detected
- **stream_ip.discards**: fragments discarded
- **stream_ip.drops**: fragments dropped
- **stream_ip.frag timeouts**: datagrams abandoned
- **stream_ip.fragments**: total fragments
- **stream_ip.memory faults**: memory faults
- **stream_ip.nodes deleted**: fragments deleted from tracker
- **stream_ip.nodes inserted**: fragments added to tracker
- **stream_ip.overlaps**: overlapping fragments
- **stream_ip.reassembled**: reassembled datagrams
- **stream_ip.trackers added**: datagram trackers created
- **stream_ip.trackers freed**: datagram trackers released
- **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.data trackers**: tcp session tracking started on data
- **stream_tcp.discards**: tcp packets discarded
- **stream_tcp.events**: events generated
- **stream_tcp.gaps**: missing data between PDUs
- **stream_tcp.ignored**: tcp packets ignored
- **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 segs**: number of times the maximum queued segment limit was reached
- **stream_tcp.overlaps**: overlapping segments queued
- **stream_tcp.rebuilt buffers**: rebuilt PDU sections
- **stream_tcp.rebuilt packets**: total reassembled PDUs
- **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 sessions
- **stream_tcp.syn trackers**: tcp session tracking started on syn
- **stream_tcp.syn-ack trackers**: tcp session tracking started on syn-ack
- **stream_tcp.timeouts**: sessions timed out
- **stream_tcp.trackers created**: tcp session trackers created
- **stream_tcp.trackers released**: tcp session trackers released
- **stream_tcp.untracked**: tcp packets not tracked
- **stream_udp.created**: udp session trackers created
- **stream_udp.released**: udp session trackers released
- **stream_udp.sessions**: total udp sessions
- **stream_udp.timeouts**: udp session timeouts
- **tcp.bad checksum (ip4)**: nonzero tcp over ip checksums
- **tcp.bad checksum (ip6)**: nonzero tcp over ipv6 checksums
- **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

14.8 Generators

- **105**: back_orifice
- **106**: rpc_decode
- **112**: arp_spoof
- **116**: arp
- **116**: auth
- **116**: decode
- **116**: eapol
- **116**: erspan2
- **116**: erspan3
- **116**: esp
- **116**: eth
- **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
- **116**: wlan
- **119**: http_global
- **120**: http_inspect
- **122**: port_scan
- **123**: stream_ip
- **125**: ftp_server
- **126**: telnet
- **129**: stream_tcp
- **134**: ppm
- **219**: new_http_inspect

14.9 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) hlen < minimum
- **116:3** (ipv4) IP dgm len < IP Hdr len
- **116:4** (ipv4) Ipv4 Options found with bad lengths
- **116:5** (ipv4) Truncated Ipv4 Options
- **116:6** (ipv4) IP dgm len > captured len
- **116:45** (tcp) TCP packet len 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
- **116:110** (eapol) truncated EAP header
- **116:111** (eapol) EAP key truncated
- **116:112** (eapol) EAP header truncated
- **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
- **116:133** (wlan) bad 802.11 LLC header
- **116:134** (wlan) bad 802.11 extra LLC info
- **116:150** (decode) bad traffic loopback IP

- **116:151** (decode) bad traffic 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) IP dgm len < IP Hdr len
- **116:275** (ipv6) IP dgm len > captured len
- **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) bad traffic 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 eth header
- **116:425** (ipv4) truncated IP4 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) ICMP6 type not decoded
- **116:432** (icmp6) ICMP6 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) MISC IP option set
- **116:445** (udp) misc large UDP Packet
- **116:446** (tcp) BAD-TRAFFIC TCP port 0 traffic
- **116:447** (udp) BAD-TRAFFIC UDP port 0 traffic
- **116:448** (ipv4) BAD-TRAFFIC IP reserved bit set
- **116:449** (ipv4) BAD-TRAFFIC unassigned/reserved IP protocol
- **116:450** (decode) BAD-TRAFFIC bad IP protocol
- **116:451** (icmp4) ICMP path MTU denial of service attempt
- **116:452** (icmp4) BAD-TRAFFIC Linux ICMP header DOS attempt
- **116:453** (ipv6) BAD-TRAFFIC ISATAP-addressed IPv6 traffic spoofing attempt
- **116:454** (pgm) BAD-TRAFFIC PGM nak list overflow attempt
- **116:455** (igmp) DOS IGMP IP options validation attempt
- **116:456** (ipv6) too many IP6 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 < 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** (decode) too many protocols present
- **119:1** (http_global) ascii encoding
- **119:2** (http_global) double decoding attack
- **119:3** (http_global) u encoding
- **119:4** (http_global) bare byte unicode encoding
- **119:5** (http_global) base36 encoding
- **119:6** (http_global) UTF-8 encoding
- **119:7** (http_global) IIS unicode codepoint encoding
- **119:8** (http_global) multi_slash encoding
- **119:9** (http_global) IIS backslash evasion
- **119:10** (http_global) self directory traversal
- **119:11** (http_global) directory traversal
- **119:12** (http_global) apache whitespace (tab)
- **119:13** (http_global) non-RFC http delimiter
- **119:14** (http_global) non-RFC defined char
- **119:15** (http_global) oversize request-URI directory
- **119:16** (http_global) oversize chunk encoding
- **119:17** (http_global) unauthorized proxy use detected
- **119:18** (http_global) webroot directory traversal
- **119:19** (http_global) long header
- **119:20** (http_global) max header fields
- **119:21** (http_global) multiple content length
- **119:22** (http_global) chunk size mismatch detected
- **119:23** (http_global) invalid ip in true-client-IP/XFF header
- **119:24** (http_global) multiple host hdrs detected
- **119:25** (http_global) hostname exceeds 255 characters

- **119:26** (http_global) header parsing space saturation
- **119:27** (http_global) client consecutive small chunk sizes
- **119:28** (http_global) post w/o content-length or chunks
- **119:29** (http_global) multiple true IPs in a session
- **119:30** (http_global) both true-client-IP and XFF hdrs present
- **119:31** (http_global) unknown method
- **119:32** (http_global) simple request
- **119:33** (http_global) unescaped space in http URI
- **119:34** (http_global) too many pipelined requests
- **120:1** (http_inspect) anomalous http server on undefined HTTP port
- **120:2** (http_inspect) invalid status code in HTTP response
- **120:3** (http_inspect) no content-length or transfer-encoding in HTTP response
- **120:4** (http_inspect) HTTP response has UTF charset which failed to normalize
- **120:5** (http_inspect) HTTP response has UTF-7 charset
- **120:6** (http_inspect) HTTP response gzip decompression failed
- **120:7** (http_inspect) server consecutive small chunk sizes
- **120:8** (http_inspect) invalid content-length or chunk size
- **120:9** (http_inspect) javascript obfuscation levels exceeds 1
- **120:10** (http_inspect) javascript whitespaces exceeds max allowed
- **120:11** (http_inspect) multiple encodings within javascript obfuscated data
- **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
- **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

- **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 $\Leftarrow$ 0
- **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
- **134:1** (ppm) rule options disabled by rule latency
- **134:2** (ppm) rule options re-enabled by rule latency
- **134:3** (ppm) packet aborted due to latency
- **219:1** (new_http_inspect) ascii encoding
- **219:2** (new_http_inspect) double decoding attack
- **219:3** (new_http_inspect) u encoding
- **219:4** (new_http_inspect) bare byte unicode encoding
- **219:5** (new_http_inspect) obsolete event—should not appear
- **219:6** (new_http_inspect) UTF-8 encoding
- **219:7** (new_http_inspect) IIS unicode codepoint encoding
- **219:8** (new_http_inspect) multi_slash encoding
- **219:9** (new_http_inspect) IIS backslash evasion
- **219:10** (new_http_inspect) self directory traversal
- **219:11** (new_http_inspect) directory traversal
- **219:12** (new_http_inspect) apache whitespace (tab)

- **219:13** (new_http_inspect) non-RFC http delimiter
- **219:14** (new_http_inspect) non-RFC defined char
- **219:15** (new_http_inspect) oversize request-uri directory
- **219:16** (new_http_inspect) oversize chunk encoding
- **219:17** (new_http_inspect) unauthorized proxy use detected
- **219:18** (new_http_inspect) webroot directory traversal
- **219:19** (new_http_inspect) long header
- **219:20** (new_http_inspect) max header fields
- **219:21** (new_http_inspect) multiple content length
- **219:22** (new_http_inspect) chunk size mismatch detected
- **219:23** (new_http_inspect) invalid IP in true-client-IP/XFF header
- **219:24** (new_http_inspect) multiple host hdrs detected
- **219:25** (new_http_inspect) hostname exceeds 255 characters
- **219:26** (new_http_inspect) header parsing space saturation
- **219:27** (new_http_inspect) client consecutive small chunk sizes
- **219:28** (new_http_inspect) post w/o content-length or chunks
- **219:29** (new_http_inspect) multiple true ips in a session
- **219:30** (new_http_inspect) both true-client-IP and XFF hdrs present
- **219:31** (new_http_inspect) unknown method
- **219:32** (new_http_inspect) simple request
- **219:33** (new_http_inspect) unescaped space in HTTP URI
- **219:34** (new_http_inspect) too many pipelined requests
- **219:35** (new_http_inspect) anomalous http server on undefined HTTP port
- **219:36** (new_http_inspect) invalid status code in HTTP response
- **219:37** (new_http_inspect) no content-length or transfer-encoding in HTTP response
- **219:38** (new_http_inspect) HTTP response has UTF charset which failed to normalize
- **219:39** (new_http_inspect) HTTP response has UTF-7 charset
- **219:40** (new_http_inspect) HTTP response gzip decompression failed
- **219:41** (new_http_inspect) server consecutive small chunk sizes
- **219:42** (new_http_inspect) invalid content-length or chunk size
- **219:43** (new_http_inspect) javascript obfuscation levels exceeds 1
- **219:44** (new_http_inspect) javascript whitespaces exceeds max allowed
- **219:45** (new_http_inspect) multiple encodings within javascript obfuscated data
- **219:46** (new_http_inspect) SWF file zlib decompression failure
- **219:47** (new_http_inspect) SWF file LZMA decompression failure

- **219:48** (new_http_inspect) PDF file deflate decompression failure
- **219:49** (new_http_inspect) PDF file unsupported compression type
- **219:50** (new_http_inspect) PDF file cascaded compression
- **219:51** (new_http_inspect) PDF file parse failure

14.10 Command Set

- **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**(): load new configuration
- **snort.resume**(): continue packet processing
- **snort.rotate_stats**(): roll perfmonitor log files
- **snort.show_plugins**(): show available plugins

14.11 Signals

Important
Signal numbers are for the system that generated this documentation and are not applicable elsewhere.

- **term**(15): shutdown normally
- **int**(2): shutdown normally
- **quit**(3): shutdown as if started with --dirty-pig
- **stats**(30): dump stats to stdout
- **rotate**(31): rotate stats files
- **reload**(1): reload config file
- **hosts**(16): reload hosts file

14.12 Configuration Changes

```
change -> alertfile: 'config alertfile:' ==> 'alert_fast.file'
change -> alertfile: 'config alertfile:' ==> 'alert_full.file'
change -> attribute_table: '<FRAG_POLICY>bsd-right</FRAG_POLICY>' ==> 'hosts.frag_policy =  ↩
    bsd_right'
change -> attribute_table: '<STREAM_POLICY>grannysmith</STREAM_POLICY>' ==> 'hosts. ↩
    tcp_policy = macos'
change -> attribute_table: '<STREAM_POLICY>hpux11</STREAM_POLICY>' ==> 'hosts.tcp_policy =  ↩
    hpux'
```

```
change -> attribute_table: '<STREAM_POLICY>win2003</STREAM_POLICY>' ==> 'hosts.tcp_policy = win-2003'
change -> attribute_table: '<STREAM_POLICY>win2k3</STREAM_POLICY>' ==> 'hosts.tcp_policy = win-2003'
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 'autogenerate_preprocessor_decoder_rules'  ==> 'ips.enable_builtin_rules'
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 'disable_inline_init_failopen'  ==> 'packets.enable_inline_init_failopen'
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 'policy_mode'  ==> 'ips.mode'
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: 'bleedover-warnings-enabled' ==> 'bleedover_warnings_enabled'
change -> detection: 'debug-print-fast-pattern' ==> 'debug_print_fast_pattern'
change -> detection: 'debug-print-nocontent-rule-tests' ==> ' ↩
    debug_print_nocontent_rule_tests'
change -> detection: 'debug-print-rule-group-build-details' ==> ' ↩
    debug_print_rule_group_build_details'
change -> detection: 'debug-print-rule-groups-compiled' ==> ' ↩
    debug_print_rule_groups_compiled'
change -> detection: 'debug-print-rule-groups-uncompiled' ==> ' ↩
    debug_print_rule_groups_uncompiled'
change -> detection: 'enable-single-rule-group' ==> 'enable_single_rule_group'
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 -> dynamicdetection ==> 'snort.--plugin_path=<path>'
change -> dynamicengine ==> 'snort.--plugin_path=<path>'
change -> dynamicpreprocessor ==> 'snort.--plugin_path=<path>'
change -> dynamicsidechannel ==> 'snort.--plugin_path=<path>'
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_lookup_timeout' ==> 'lookup_timeout'
change -> file: 'config file: file_signature_depth' ==> 'signature_depth'
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: 'preprocessor frag3_engine: timeout 0' ==> 'session_timeout 256'
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: '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: 'non_rfc_char' ==> 'non_rfc_chars'
change -> http_inspect_server: 'ports' ==> 'bindings'
change -> http_inspect_server: 'post_depth [-1:65495]' ==> 'post_depth [-1:65535]'
change -> mpls_payload_type: 'config mpls_payload_type: ethernet' ==> 'mpls_payload_type =  ↩
    eth'
change -> mpls_payload_type: 'config mpls_payload_type: ipv4' ==> 'mpls_payload_type = ip4'
change -> mpls_payload_type: 'config mpls_payload_type: ipv6' ==> 'mpls_payload_type = ip6'
change -> normalizers: 'block' ==> 'base'
change -> normalizers: 'pad' ==> 'base'
change -> normalizers: 'req_pay' ==> 'base'
change -> normalizers: 'req_urg' ==> 'base'
change -> normalizers: 'req_urp' ==> 'base'
change -> normalizers: 'rsv' ==> 'base'
change -> normalizers: 'trim_mss' ==> 'trim'
change -> normalizers: 'trim_rst' ==> 'trim'
change -> normalizers: 'trim_syn' ==> 'trim'
change -> normalizers: 'trim_win' ==> 'trim'
```

```
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 -> ppm: 'debug-pkts' ==> 'debug_pkts'
change -> ppm: 'fastpath-expensive-packets' ==> 'fastpath_expensive_packets'
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-expensive-rules' ==> 'suspend_expensive_rules'
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 -> profile: 'sort avg_ticks_per_nomatch' ==> 'sort = avg_ticks_per_no_match'
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 -> 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: 'both ports' ==> 'binder.when.ports; binder.when.role = any'
change -> stream5_tcp: 'both protocol' ==> 'binder.when.proto; binder.when.role = any'
change -> stream5_tcp: 'client ports' ==> 'binder.when.ports; binder.when.role = client'
change -> stream5_tcp: 'client protocol' ==> 'binder.when.proto; binder.when.role = client'
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 grannysmith' ==> 'stream_tcp.policy = macos'
change -> stream5_tcp: 'policy hpux11' ==> 'stream_tcp.policy = hpux'
change -> stream5_tcp: 'policy win2003' ==> 'stream_tcp.policy = win-2003'
change -> stream5_tcp: 'policy win2k3' ==> 'stream_tcp.policy = win-2003'
change -> stream5_tcp: 'server ports' ==> 'binder.when.ports; binder.when.role = server'
change -> stream5_tcp: 'server protocol' ==> 'binder.when.proto; binder.when.role = server'
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 '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 -> 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 -> perfmonitor: 'atexitonly'
deleted -> perfmonitor: 'atexitonly: base-stats'
deleted -> perfmonitor: 'atexitonly: events-stats'
deleted -> perfmonitor: 'atexitonly: flow-ip-stats'
deleted -> perfmonitor: 'atexitonly: flow-stats'
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 -> 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'
```

14.13 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_syslog** (logger): output event to syslog
- **alert_test** (logger): output event in custom tsv format
- **alert_unixsock** (logger): output event over unix socket

- **alerts** (basic): configure alerts
- **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
- **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
- **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
- **dsize** (ips_option): rule option to test payload size
- **eapol** (codec): support for extensible authentication protocol over LAN
- **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
- **file_data** (ips_option): rule option to set detection cursor to file data
- **file_id** (basic): configure file identification
- **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** (data): 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
- **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_global** (data): http inspector global configuration and client rules for use with http_server
- **http_header** (ips_option): rule option to set the detection cursor to the normalized header(s)
- **http_inspect** (inspector): http inspection and server rules; also configure http_inspect
- **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_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_uri** (ips_option): rule option to set the detection cursor to the normalized URI 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
- **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
- **log_codecs** (logger): log protocols in packet by layer

- **log_pcap** (logger): log packet in pcap format
- **metadata** (ips_option): rule option for conveying arbitrary name, value data within the rule text
- **mpls** (codec): support for multiprotocol label switching
- **msg** (ips_option): rule option summarizing rule purpose output with events
- **network** (basic): configure basic network parameters
- **new_http_inspect** (inspector): new HTTP inspector
- **normalizer** (inspector): packet scrubbing for inline mode
- **output** (basic): configure general output parameters
- **packets** (basic): configure basic packet handling
- **pcre** (ips_option): rule option for matching payload data with regex
- **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
- **port_scan** (inspector): port scan inspector; also configure port_scan_global
- **port_scan_global** (data): shared settings for port_scan inspectors for use with port_scan
- **ppm** (basic): packet and rule latency monitoring and control (requires --enable-ppm)
- **pppoe** (codec): support for point-to-point protocol over ethernet
- **priority** (ips_option): rule option for prioritizing events
- **process** (basic): configure basic process setup
- **profile** (basic): configure profiling of rules and/or modules (requires --enable-perf-profiling)
- **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
- **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
- **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
- **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
- **sid** (ips_option): rule option to indicate signature number
- **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
- **stream** (inspector): common flow tracking
- **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
- **suppress** (basic): configure event suppressions
- **tag** (ips_option): rule option to log additional packets
- **tcp** (codec): support for transmission control protocol
- **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
- **wlan** (codec): support for wireless local area network protocol (DLT 105) :leveloffset: 0

14.13.1 Plugin Listing

- **codec::arp**: support for address resolution protocol
- **codec::auth**: support for IP authentication header
- **codec::eapol**: support for extensible authentication protocol over LAN
- **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::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::linux_sll**: support for Linux SLL (DLT 113)
- **codec::llc**: support for logical link control
- **codec::mpls**: support for multiprotocol label switching
- **codec::null**: support for null encapsulation (DLT 0)
- **codec::pgm**: support for pragmatic general multicast
- **codec::ppp**: support for point-to-point encapsulation (DLT 9)
- **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::raw4**: support for unencapsulated IPv4 (DLT 12) (DLT 228)
- **codec::raw6**: support for unencapsulated IPv6 (DLT 229)
- **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::vlan**: support for local area network
- **codec::wlan**: support for wireless local area network protocol (DLT 105)
- **data::ftp_client**: FTP inspector client module
- **data::http_global**: shared HTTP inspector settings
- **data::port_scan_global**: shared settings for port_scan inspectors for use with port_scan

- **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::ftp_data**: FTP data channel handler
- **inspector::ftp_server**: FTP inspector server module
- **inspector::http_inspect**: main HTTP inspector module
- **inspector::new_http_inspect**: the new HTTP inspector!
- **inspector::normalizer**: packet scrubbing for inline mode
- **inspector::perf_monitor**: performance monitoring and flow statistics collection
- **inspector::port_scan**: port scan inspector; also configure port_scan_global
- **inspector::rpc_decode**: RPC inspector
- **inspector::stream**: common flow tracking
- **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::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::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::detection_filter**: rule option to require multiple hits before a rule generates an event
- **ips_option::dsize**: rule option to test payload size
- **ips_option::file_data**: rule option to set detection cursor to file data

- **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::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 header(s)
- **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_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_uri**: rule option to set the detection cursor to the normalized URI 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::metadata**: rule option for conveying arbitrary name, value data within the rule text
- **ips_option::msg**: rule option summarizing rule purpose output with events
- **ips_option::pcre**: rule option for matching payload data with regex
- **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::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::seq**: rule option to check TCP sequence number
- **ips_option::session**: rule option to check user data from TCP sessions
- **ips_option::sid**: rule option to indicate signature number
- **ips_option::so**: rule option to call custom eval function
- **ips_option::soid**: rule option to specify a shared object rule ID
- **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_syslog**: output event to syslog
- **logger::alert_test**: output event in custom tsv format
- **logger::alert_unixsock**: output event over unix socket
- **logger::log_codecs**: log protocols in packet by layer
- **logger::log_null**: support for null encapsulation
- **logger::log_pcap**: log packet in pcap format
- **logger::unified2**: output event and packet in unified2 format file
- **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_bnfa_q**: Aho-Corasick Binary NFA (low memory, high performance) with queued events
- **search_engine::ac_full**: Aho-Corasick Full (high memory, best performance)
- **search_engine::ac_full_q**: Aho-Corasick Full (high memory, best performance) with queued events
- **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

14.14 Extending Snort++

14.14.1 Plugins

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

- Codec - to decode and encode packets
- Inspector - like the prior 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

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.

The Module is pervasive in Snort.`It is how everything, including plugins, are configured.It also provides access to builtin rules.And as the glue that binds functionality to Snort`, the capabilities of a Module are expected to grow to include statistics support, etc.

Module configuration is handled by a list of Parameters. Most parameters can be validated by the framework, which means for example that conversion from string to number is done in exactly one place. Providing the builtin rules allows the documentation to include them automatically and also allows for autogenerating the rules at startup.

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.)

14.14.2 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.

General

- Generally try to follow http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml, but there are a few differences.

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.
- Use lower case filenames with underscores.

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.
- 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 = P | H | M | L, indicating perf, high, med, or low priority. For now, H, M, or L can indicate alpha 1, 2, or 3. Perf changes fall between alpha 1 and 2.

Logging

- Messages intended for the user should not look like debug messages. In, other words, the function name should not be included.
- Most debug messages should just be deleted.
- Don't bang your messages (no !). The user feels bad enough about the problem already w/o you shouting at him.

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.
- 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.

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.

Formatting

- 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();
```

Headers

- Don't hesitate to create a new header if it is needed. Don't lump unrelated stuff into a 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"
```

- 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.
- Include required headers, all required headers, and nothing but required headers. Don't just clone a bunch of headers because it is convenient.
- Any file depending of #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.

Warnings

- With g++, use at least these compiler flags:

```
-Wall -Wextra -pedantic -Wformat -Wformat-security
-Wunused-but-set-variable -Wno-deprecated-declarations
```

- Then Fix All Warnings. None Allowed.

Other

- Prefer *and* over && and *or* over || for new source files.