NAME

yafscii − YAF Flow Printer

SYNOPSIS

```
yafscii [--in INPUT_SPECIFIER] [--out OUTPUT_SPECIFIER]
        [--nextdir PROCESSED_INPUT_DIRECTORY]
        [--faildir FAILED_INPUT_DIRECTORY]
        [--poll POLLING_DELAY] [--lock]
        [--log LOG_SPECIFIER] [--loglevel LOG_LEVEL]
        [--verbose] [--version] [--daemon] [--foreground]
        [--tabular]
```

DESCRIPTION

yafscii takes IPFIX flow data files generated by *yaf*(1) and prints them in an ASCII format loosely analogous to that produced by *tcpdump*(1), with one flow per line. The text output format is detailed in the **OUTPUT** section, below. yafscii is generally intended to be used to print single files for verification or debugging purposes, or to operate in a pipe with *yaf*(1), but it can be used as a daemon as well.

OPTIONS

Input Options

The input specifier determines where yafscii will read its input from. If the input specifier is not given, yaf defaults to reading from standard input.

−−in *INPUT_SPECIFIER*

INPUT_SPECIFIER is an input specifier. This is a filename, a directory name, a file glob pattern (in which case it should be escaped or quoted to prevent the shell from expanding the glob pattern), or the string **−** to read from standard input.

Output Options

The output specifier determines where yaf will send its output. If reading standard input, output defaults to standard output. If reading from files on disk, output defaults to one file per input file, named as the input file in the same directory as the input file with a **.txt** extension.

−−out *OUTPUT_SPECIFIER*

OUTPUT_SPECIFIER is an output specifier. This should be a filename or a directory name, or the string **−** to write to standard output.

−−tabular

Use tabular output mode, which is designed for easy parsability over human readability. See the Tabular Output section below for details.

Daemon Options

These options are used to run yafscii in daemon mode for batch processing of pcap dumpfiles.

−−daemon

Run yafscii in daemon mode. Instead of processing its input then exiting, yafscii will continually look for new input matching its input specifier. This will cause yaf to fork into the background and exit. =item **−−foreground**

Instead of forking in **−−daemon** mode, stay in the foreground. Useful for debugging.

−−lock

Use lockfiles for concurrent file access protection. Highly recommended in **−−daemon** mode, especially if two daemons are interacting through a given directory.

−−poll *POLLING_DELAY*

POLLING_DELAY is the polling delay in seconds; how long yaf will wait for new input when none is available. The default is 60 seconds.

−−**nextdir** *PROCESSED_INPUT_DIRECTORY*
When reading from files, if this option is present, input files will be moved to *PROCESSED_INPUT_DIRECTORY* after they are successfully processed. The special string **delete** will cause successfully processed input to be removed instead. This option is required in daemon mode.

−−**faildir** *FAILED_INPUT_DIRECTORY*
When reading from files, if this option is present, input files will be moved to *FAILED_INPUT_DIRECTORY* if processing failed. The special string **delete** will cause failed input to be removed instead. This option is required in daemon mode.

Logging Options

These options are used to specify how log messages are routed. yaf can log to standard error, regular files, or the UNIX syslog facility.

−−**log** *LOG_SPECIFIER*
Specifies destination for log messages. *LOG_SPECIFIER* can be a *syslog*(3) facility name, the special value **stderr** for standard error, or the *absolute* path to a file for file logging. Standard error logging is only available in −−**daemon** mode if −−**foreground** is present. The default log specifier is **stderr** if available, **user** otherwise.

−−**loglevel** *LOG_LEVEL*
Specify minimum level for logged messages. In increasing levels of verbosity, the supported log levels are **quiet**, **error**, **critical**, **warning**, **message**, **info**, and **debug**. The default logging level is **warning**.

−−**verbose**
Equivalent to −−**loglevel debug**.

−−**version**
If present, print version and copyright information to standard error and exit.

OUTPUT

Human-Readable Output

yafscii's default output format, like that of tcpdump, is designed to be easily human−readable, at the possible expense of ease of automated parsing. Each flow is represented by a single output line representing the flow itself, followed by zero or more indented lines containing flow payload in hexdump format. This section details each flow format, where each field specifier is as follows:

start-time, *end-time*
Flow start or end time in ISO 8601 format, with milliseconds (*YYYY−MM−DD hh*:*mm*:*ss.ssss*). Start time is printed with a date; end time is not. End time is only present if the flow has a non-zero duration.

duration
Flow duration in fractional seconds. Only present if the flow has a non-zero duration.

proto
IP protocol identifier in decimal format.

sip, *dip*
Source or destination IPv4 address in dotted-quad format or IPv6 address in RFC 2373 format.

sp, *dp*
Source or destination transport port in decimal format.

type, *code*
ICMP type or code in decimal format.

isn, *risn*
Forward or reverse initial TCP sequence number in hexadecimal format.

iflags, riflags, uflags, ruflags
: Foward or reverse first-packet TCP flags; forward or reverse *n*th-packet TCP flags union; where each flags bit is represented by the first character in the flag's name: **F**IN, **S**YN, **R**ST, **P**SH, **A**CK, **U**RG, **E**CE, **C**WR. The character **0** means no flags are set (and will appear in the *n*th-packet field for single-packet TCP flows).

tag, rtag
: Forward or reverse first-packet 802.1q VLAN tag in hexadecimal format.

pkt, rpkt
: Forward or reverse packet count in decimal format.

oct, roct
: Forward or reverse octet count in decimal format.

rtt
: Round-trip time estimate in milliseconds in decimal format.

end-reason
: If not present, the flow ended normally (i.e., by TCP RST or FIN). Otherwise, the *end-reason* is one of the following strings:

  idle
  : Flow was expired by idle timeout. No packets were received for *IDLE_TIMEOUT* seconds (see *yaf*(1)) and the flow was presumed closed.

  active
  : Flow was expired by active timeout. The flow's duration was longer than *ACTIVE_TIMEOUT* seconds (see *yaf*(1)) and the flow was flushed from the flow table.

  eof
  : Flow was still active in the flow table at the end of the dumpfile or at *yaf*(1) shutdown time; it was flushed as the flow table was cleared.

  rsrc
  : Flow was prematurely flushed as **idle** because more than *FLOW_TABLE_MAX* flows (see *yaf*(1)) were active in the flow table.

applabel
: The application label, if *yaf*(1) was built with application labeling enabled and the application labeler was able to identify the payload in the flow.

entropy
: The Shannon Entropy for the forward then the reverse flow payload if the a payload existed and *yaf*(1) was built with entropy enabled.

Each flow line format is as follows:

Unidirectional IP flow
: *start-time* [**−** *end-time* **(***duration* **sec)**] **ip** *proto sip* **=>** *dip* [**vlan** *tag*] **(***pkt*/*oct* **−>)** [*end-reason*]

Unidirectional UDP flow
: *start-time* [**−** *end-time* **(***duration* **sec)**] **udp** *sip*:*sp* **=>** *dip*:*dp* [**vlan** *tag*] **(***pkt*/*oct* **−>)** [*end-reason*]

Unidirectional TCP flow
: *start-time* [**−** *end-time* **(***duration* **sec)**] **tcp** *sip*:*sp* **=>** *dip*:*dp isn iflags*/*uflags* [**vlan** *tag*] **(***pkt*/*oct* **−>)** [*end-reason*]

Unidirectional ICMP flow
: *start-time* [**−** *end-time* **(***duration* **sec)**] **icmp [***type*:*code***]** *sip* **=>** *dip* [**vlan** *tag*] **(***pkt*/*oct* **−>)** [*end-reason*]

Bidirectional IP flow
: *start-time* [**−** *end-time* **(***duration* **sec)**] **ip** *proto sip* **=>** *dip* [**vlan** *tag*:*rtag*] **(***pkt*/*oct* **<−>** *rpkt*/*roct***) rtt** *rtt* **ms** [*end-reason*]

Bidirectional UDP flow

start-time [− *end-time* **(***duration* **sec)**] **udp** *sip*:*sp* => *dip*:*dp* [**vlan** *tag*:*rtag*] **(***pkt*/*oct* <−> *rpkt*/*roct***) rtt** *rtt* **ms** [*end-reason*]

Bidirectional TCP flow

start-time [− *end-time* **(***duration* **sec)**] **tcp** *sip*:*sp* => *dip*:*dp isn*:*risn iflags*/*uflags*:*riflags*/*ruflags* [**vlan** *tag*:*rtag*] **(***pkt*/*oct* <−> *rpkt*/*roct***) rtt** *rtt* **ms** [*end-reason*]

If present, the payload follows each flow line. Forward direction payload lines are prefixed with the string −>, and reverse direction payload lines are prefixed with the string <−. Payload is only taken from the first packet for non-TCP flows (see *yaf*(1)).

Tabular Output

In −−**tabular** mode, yafscii prints its output as a table, without a header, with one flow per line and no payload information. Each column is separated by a pipe character. Columns have constant width and are filled with leading zeroes or spaces as appropriate. Every column appears in each row whether it is present in the flow data or not; non-present columns are represented with a **0**. All columns are formatted as they are in the human-readable output, except *end-time* which appears with a data and *rtt* which is expressed in fractional seconds instead of decimal milliseconds. For ICMP flows, ICMP type and code appear in the *dp* field, which has the value 256(*type*) + *code*. The order of columns is as follows:

start-time| *end-time*| *duration*| *rtt*| *proto*| *sip*| *sp*| *dip*| *dp*| *iflags*| *uflags*| *riflags*| *ruflags*| *isn*| *risn*| *tag*| *rtag*| *pkt*| *oct*| *rpkt*| *roct*| *applabel*| *entropy*| *rentropy*| *end-reason*

SIGNALS

yafscii responds to **SIGINT** or **SIGTERM** by terminating input processing and exiting.

BUGS

Known issues are listed in the **README** file in the YAF tools source distribution. Note that YAF should be considered alpha-quality software; not every concievable input and optionis exhaustively tested at each release, and specific features may be completely untested. Please be mindful of this before deploying YAF in production environments. YAF's output format may also change, as the development of YAF is intended to track progress in the IPFIX working group; the file output of YAF should not presently be used for archival storage of flow data. Bug reports and feature requests may be sent directly to the Network Situational Awareness team at <netsa−yaf@cert.org>.

AUTHORS

Brian Trammell, Chris Inacio <inacio@cert.org>, Michael Duggan <mwd@cert.org>, and the CERT Network Situational Awareness Group Engineering Team, http://www.cert.org/netsa.

SEE ALSO

yaf(1)