

NAME**yaf** – Yet Another Flowmeter**SYNOPSIS**

```

yaf      [--in INPUT_SPECIFIER] [--out OUTPUT_SPECIFIER]
        [--live LIVE_TYPE] [--ipfix TRANSPORT_PROTOCOL]
        [--rotate ROTATE_DELAY ] [--lock] [--caplist]
        [--simulate] [--gre-decode] [--no-frag]
        [--max-frags FRAG_TABLE_MAX]
        [--ip4-only] [--ip6-only]
        [--idle-timeout IDLE_TIMEOUT]
        [--active-timeout ACTIVE_TIMEOUT]
        [--max-payload PAYLOAD_OCTETS]
        [--max-flows FLOW_TABLE_MAX]
        [--export-payload] [--silk]
        [--uniflow] [--mac] [--force-ip6-export]
        [--observation-domain DOMAIN_ID] [--entropy]
        [--applabel] [--applabel-rules RULES_FILE]
        [--ipfix-port PORT] [--tls] [--tls-ca CA_PEM_FILE]
        [--tls-cert CERT_PEM_FILE] [--tls-key KEY_PEM_FILE]
        [--become-user UNPRIVILEGED_USER]
        [--become-group UNPRIVILEGED_GROUP]
        [--log LOG_SPECIFIER] [--loglevel LOG_LEVEL]
        [--verbose] [--version]

```

DESCRIPTION

yaf is Yet Another Flowmeter and YAF is the suite of tools to do flow metering. *yaf* is used as a sensor to capture flow information on a network and export that information in IPFIX format. It reads packet data from *pcap* (3) dumpfiles as generated by *tcpdump* (1), from live capture from an interface using *pcap* (3), or from an Endace DAG capture device, aggregates these packets into flows, and exports flow records via IPFIX over SCTP, TCP or UDP, or into serialized IPFIX message streams (IPFIX files) on the local file system.

Since YAF is designed to be deployed on white-box sensors attached to local network segments or span ports at symmetric routing points, it supports bidirectional flow assembly natively. Biflow export is done via the export method specified in **RFC 5103** Bidirectional Flow Export using IPFIX. See the “OUTPUT” section below for information on this format.

yaf also supports experimental partial payload capture, specifically for banner-grabbing applications and protocol verification purposes.

The output of *yaf* is designed to be collected and manipulated by flow processing toolchains supporting IPFIX. The *yafscii* (1) tool, which is installed as part of YAF, can also be used to print *yaf* output in a human-readable format somewhat reminiscent of *tcpdump* (1). *yaf* output can also be analyzed using the SiLK suite, and the *nafalize* (1) tool, both available from the CERT NetSA group.

OPTIONS**Input Options**

These options control where *yaf* will take its input from. *yaf* can read packets from a pcap dumpfile (as generated by `tcpdump -w`) or live from an interface via *libpcap* or *libdag*. By default, if no input options are given, *yaf* reads a pcap dumpfile on standard input.

--in *INPUT_SPECIFIER*

INPUT_SPECIFIER is an input specifier. If **--live** is given, this is the name of an interface (e.g. `eth0`, `en0`, `dag0`) to capture packets from. Otherwise, it is a filename; the string `-` may be used to read from standard input (the default).

--caplist

If present, treat the filename in *INPUT_SPECIFIER* as an ordered newline-delimited list of pathnames to *pcap* (3) dumpfiles. Blank lines and lines beginning with the character '#' within this are ignored. All pathnames are evaluated with respect to the working directory **yaf** is run in. These dumpfiles are processed in order using the same flow table, so they must be listed in ascending time order. This option is intended to ease the use of **yaf** with rotated or otherwise split *tcpdump* (1) output.

--live LIVE_TYPE

If present, capture packets from an interface named in the *INPUT_SPECIFIER*. *LIVE_TYPE* is one of **pcap** for packet capture via libpcap, or **dag** for packet capture via an Endace DAG interface using libdag. **dag** is only available if **yaf** was built with Endace DAG support.

Output Options

These options control where **yaf** will send its output. **yaf** can write flows to an IPFIX file or export flows to an IPFIX collector over SCTP, TCP, or UDP. By default, if no output options are given, **yaf** writes an IPFIX file to standard output.

--out OUTPUT_SPECIFIER

OUTPUT_SPECIFIER is an output specifier. If **--ipfix** is present, the *OUTPUT_SPECIFIER* specifies the hostname or IP address of the collector to which the flows will be exported. Otherwise, if **--rotate** is present, *OUTPUT_SPECIFIER* is the prefix name of each output file to write to. Otherwise, *OUTPUT_SPECIFIER* is a filename in which the flows will be written; the string `-` may be used to write to standard output (the default).

--ipfix TRANSPORT_PROTOCOL

If present, causes **yaf** to operate as an IPFIX exporter, sending IPFIX Messages via the specified transport protocol to the collector (e.g., SiLK's *rwflowpack* or *flowcap* facilities) named in the *OUTPUT_SPECIFIER*. Valid *TRANSPORT_PROTOCOL* values are **tcp**, **udp**, and **sctp**; **sctp** is only available if **yaf** was built with SCTP support. Use the **--ipfix-port**, **--tls**, **--tls-ca**, **--tls-cert**, **--tls-key**, and **--tls-pass** options to further configure the connection to the IPFIX collector.

--rotate ROTATE_DELAY

If present, causes **yaf** to write output to multiple files, opening a new output file every *ROTATE_DELAY* seconds in the input data. Rotated files are named using the prefix given in the *OUTPUT_SPECIFIER*, followed by a suffix containing a timestamp in `YYYYMMDDhhmmss` format, a decimal serial number, and the file extension **.yaf**.

--lock

Use lockfiles for concurrent file access protection on output files. This is recommended for interoperating with the Airframe *filedaemon* facility.

Decoder Options

These options are used to modify the **yaf** packet decoder's behavior. None of these options are required; the default behavior for each option when not present is noted.

--simulate

If present and reading a *pcap* dumpfile, shift dumpfile timestamps to present time and simulate packet interarrival delay. Provided for use in testing, to make a dumpfile appear to be captured live. By default, *pcap* dumpfiles are processed as quickly as they can be read from disk.

--no-frag

If present, ignore all fragmented packets. By default, **yaf** will reassemble fragments with a 30 second fragment timeout.

--max-frags FRAG_TABLE_MAX

If present, limit the number of outstanding, not-yet reassembled fragments in the fragment table to *FRAG_TABLE_MAX* by prematurely expiring fragments from the table. This option is provided to limit **yaf** resource usage when operating on data from very large networks or networks with abnormal fragmentation. By default, there is no fragment table limit, and the fragment table can grow to resource

exhaustion.

--ip4-only

If present, ignore all IPv6 packets and export IPv4 flows only. The default is to process both IPv4 and IPv6 packets.

--ip6-only

If present, ignore all IPv4 packets and export IPv6 flows only. The default is to process both IPv4 and IPv6 packets.

--gre-decode

If present, attempt to decode GRE version 0 encapsulated packets. Flows will be created from packets within the GRE tunnels. Undecodeable GRE packets will be dropped. Without this option, GRE traffic is exported as IP protocol 47 flows. This option is presently experimental.

Flow Table Options

These options are used to modify the flow table behavior within `yaf`. None of these options are required; the default behavior for each option when not present is noted.

--idle-timeout *IDLE_TIMEOUT*

Set flow idle timeout in seconds. Flows are considered idle and flushed from the flow table if no packets are received for *IDLE_TIMEOUT* seconds. The default flow idle timeout is 300 seconds (5 minutes).

--active-timeout *ACTIVE_TIMEOUT*

Set flow active timeout in seconds. Any flow lasting longer than *ACTIVE_TIMEOUT* seconds will be flushed from the flow table. The default flow active timeout is 1800 seconds (30 minutes).

--max-payload *PAYLOAD_OCTETS*

If present, capture at most *PAYLOAD_OCTETS* octets from the start of each direction of each flow. Non-TCP flows will only capture payload from the first packet. If not present, `yaf` will not attempt to capture payload. Payload capture must be enabled for payload export (**--export-payload**), application labeling (**--applabel**), and entropy evaluation (**--entropy**). Note that payload capture is still an experimental feature.

--max-flows *FLOW_TABLE_MAX*

If present, limit the number of open flows in the flow table to *FLOW_TABLE_MAX* by prematurely expiring the flows with the least recently received packets; this is analogous to an adaptive idle timeout. This option is provided to limit `yaf` resource usage when operating on data from large networks. By default, there is no flow table limit, and the flow table can grow to resource exhaustion.

--silk

If present, export flows in “SiLK mode”. This introduces the following incompatibilities with standard IPFIX export:

- * `totalOctetCount` and `reverseTotalOctetCount` are clamped to 32 bits. Any packet that would cause either of these counters to overflow 32 bits will cause the flow to close with `flowEndReason 0x02` (active timeout), and will become the first packet of a new flow. This is analogous to forcing an active timeout when the octet counters overflow.
- * The high-order bit of the `flowEndReason` IE is set on any flow created on a counter overflow, as above.
- * The high-order bit of the `flowEndReason` IE is set on any flow created on an active timeout.

Since this changes the semantics of the exported `flowEndReason` IE, it should only be used when generating flows and exporting to `rwflowpack`, `flowcap`, or writing files for processing with `rwipfix2silk`.

Export Options

These options are used to modify the the data exported by `yaf`.

--export-payload

If present, export at most `PAYLOAD_OCTETS` (the argument to **--max-payload**) octets from the start of each direction of each flow. Non-TCP flows will only export payload from the first packet. By default, `yaf` will not export flow payload.

--uniflow

If present, export biflows using the Record Adjacency method in section 3 of RFC 5103. This is useful when exporting to IPFIX Collecting Processes that are not biflow-aware.

--mac

If present, export MAC-layer information; presently, this only exports VLAN IDs for 802.1q packets.

--force-ip6-export

If present, force IPv4 flows to be exported with IPv6-mapped IPv4 addresses in `::FFFF/96`. This will cause all flows to appear to be IPv6 flows.

--observation-domain *DOMAIN_ID*

Set the `observationDomainID` on each exported IPFIX message to the given integer value. If not present, the `observationDomainId` defaults to 1.

Application Labeler Options

If `yaf` is built with application labeler support enabled (using the `--enable-applabel` option to `./configure` when `yaf` is built), then `yaf` can examine packet payloads and determine the application protocol in use within a flow, and export a 16-bit application label with each flow.

The exported application label uses the common port number for the protocol. For example, HTTP traffic, independent of what port the traffic is detected on, will be labeled with a value of 80, the default HTTP port. Labels and rules are taken from a configuration file read at `yaf` startup time.

Application labeling requires payload capture to be enabled with the **--max-payload** option. A minimum payload capture length of 384 octets is recommended for best results.

Application labeling is presently experimental. There is no support in `yafscii` for printing application label data, though SiLK does support IPFIX import and translation of the application label via `rwflow-pack`, `flowcap`, and `rwipfix2silk`.

--applabel

If present, export application label data. Requires **--max-payload** to enable payload capture.

--applabel-rules *RULES_FILE*

Read application labeler rules from *RULES_FILE*. If not present, rules are read by default from `/usr/local/etc/yafApplabelRules.conf`.

Entropy Measurement

If `yaf` is built with entropy measurement enabled (using the `--enable-entropy` option to `./configure` when `yaf` is built,) then `yaf` can examine the packet payloads and determine a Shannon Entropy value for the payload. The entropy calculation does not include the network (IP) or transport (UDP/TCP) headers. The entropy is calculated in terms of bits per byte, (log base 2.) The calculation generates a real number value between 0.0 and 8.0. That number is then converted into an 8-bit integer value between 0 and 255. Roughly, numbers above 230 are generally compressed (or encrypted) and numbers centered around approximately 140 are English text. Lower numbers carry even less information content. Another useful piece of information is that SSL/TLS tends to zero pad its packets, which causes the entropy of those flows to drop quite low.

--entropy

If present, export the entropy values for both the forward and reverse payloads. Requires the **--max-payload** option to operate.

IPFIX Connection Options

These options are used to configure the connection to an IPFIX collector.

--ipfix-port *PORT*

If **--ipfix** is present, export flows to TCP, UDP, or SCTP port *PORT*. If not present, the default IPFIX port 4739 is used. If **--tls** is also present, the default secure IPFIX port 4740 is used.

--tls

If **--ipfix** is present, use TLS to secure the connection to the IPFIX collector. Requires the *TRANSPORT_PROTOCOL* to be **tcp**, as DTLS over UDP or SCTP is not yet supported. Requires the **--tls-ca**, **--tls-cert**, and **--tls-key** options to specify the X.509 certificate and TLS key information.

--tls-ca *CA_PEM_FILE*

Use the Certificate Authority or Authorities in *CA_PEM_FILE* to verify the remote IPFIX Collecting Process' X.509 certificate. The connection to the Collecting Process will fail if its certificate was not signed by this CA (or by a certificate signed by this CA, recursively); this prevents export to unauthorized Collecting Processes. Required if **--tls** is present.

--tls-cert *CERT_PEM_FILE*

Use the X.509 certificate in *CERT_PEM_FILE* to identify this IPFIX Exporting Process. This certificate should contain the public part of the private key in *KEY_PEM_FILE*. Required if **--tls** is present.

--tls-key *KEY_PEM_FILE*

Use the private key in *KEY_PEM_FILE* for this IPFIX Exporting Process. This key should contain the private part of the public key in *CERT_PEM_FILE*. Required if **--tls** is present. If the key is encrypted, the password must be present in the *YAF_TLS_PASS* environment variable.

Privilege Options

These options are used to cause *yaf* to drop privileges when running as root for live capture purposes.

--become-user *UNPRIVILEGED_USER*

After opening the live capture device in **--live** mode, drop privilege to the named user. Using **--become-user** requires *yaf* to be run as root or *setuid* root. This option will cause all files written by *yaf* to be owned by the user *UNPRIVILEGED_USER* and the user's primary group; use **--become-group** as well to change the group *yaf* runs as for output purposes.

If running as root for live capture purposes and **--become-user** is not present, *yaf* will warn that privilege is not being dropped. We highly recommend the use of this option, especially in production environments, for security purposes.

--become-group *UNPRIVILEGED_GROUP*

--become-group can be used to change the group from the default of the user given in **--become-user**. This option has no effect if given without the **--become-user** option as well.

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. 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

`yaf`'s output consists of an IPFIX message stream. `yaf` uses a variety of templates for IPFIX data records; the information elements that may appear in these templates are enumerated below. For further information about the IPFIX information model and IPFIX message stream, see **RFC 5102**, **RFC 5101**, and **RFC 5103**.

`yaf` assigns information element numbers to reverse flow elements in biflow capture based on the standard IPFIX PEN 29305. This applies only for information elements defined in the standard IPFIX Information Model (**RFC 5102**) that do not have a reverse information element already defined. For information elements defined under the CERT PEN, a standard method is used to calculate their reverse element identifier. The method is that bit fourteen is set on in the IE field, (e.g. 16384 + the forward IE number.)

flowStartMilliseconds IE 152, 8 octets

Flow start time in milliseconds since 1970-01-01 00:00:00 UTC. Always present.

flowEndMilliseconds IE 153, 8 octets

Flow end time in milliseconds since 1970-01-01 00:00:00 UTC. Always present.

octetTotalCount IE 85, 8 octets

Number of octets in packets in forward direction of flow. Always present. May be encoded in 4 octets using IPFIX reduced-length encoding.

reverseOctetTotalCount Reverse (PEN 29305) IE 85, 8 octets

Number of octets in packets in reverse direction of flow. Present if flow has a reverse direction. May be encoded in 4 octets using IPFIX reduced-length encoding.

packetTotalCount IE 86, 8 octets

Number of packets in forward direction of flow. Always present. May be encoded in 4 octets using IPFIX reduced-length encoding.

reversePacketTotalCount Reverse (PEN 29305) IE 86, 8 octets

Number of packets in reverse direction of flow. Present if flow has a reverse direction. May be encoded in 4 octets using IPFIX reduced-length encoding.

reverseFlowDeltaMilliseconds CERT (PEN 6871) IE 21, 4 octets

Difference in time in milliseconds between first packet in forward direction and first packet in reverse direction. Correlates with (but does not necessarily represent) round-trip time. Present if flow has a reverse direction.

sourceIPv4Address IE 8, 4 octets

IPv4 address of flow source or biflow initiator. Present for IPv4 flows without IPv6-mapped addresses only.

destinationIPv4Address IE 12, 4 octets

IPv4 address of flow source or biflow responder. Present for IPv4 flows without IPv6-mapped addresses only.

sourceIPv6Address IE 27, 16 octets

IPv6 address of flow source or biflow initiator. Present for IPv6 flows or IPv6-mapped IPv4 flows only.

destinationIPv6Address IE 28, 16 octets

IPv6 address of flow source or biflow responder. Present for IPv6 flows or IPv6-mapped IPv4 flows only.

sourceTransportPort IE 7, 2 octets

TCP or UDP port on the flow source or biflow initiator endpoint. Always present.

destinationTransportPort IE 11, 2 octets

TCP or UDP port on the flow destination or biflow responder endpoint. Always present. For ICMP flows, contains ICMP type * 256 + ICMP code. This is non-standard, and an open issue in YAF.

- protocolIdentifier** IE 4, 1 octet
IP protocol of the flow. Always present.
- flowEndReason** IE 136, 1 octet
Flow end reason code, as defined by the IPFIX Information Model. Always present. In **—silk** mode, the high-order bit is set if the flow was created by continuation.
- silkAppLabel** CERT (PEN 6871) IE 33, 2 octets
Application label, defined as the primary well-known port associated with a given application. Present if the application labeler is enabled, and was able to determine the application protocol used within the flow.
- tcpSequenceNumber** IE 184, 4 octets
Initial sequence number of the forward direction of the flow. Present if the flow's **protocolIdentifier** is 6 (TCP).
- reverseTcpSequenceNumber** Reverse (PEN 29305) IE 184, 4 octets
Initial sequence number of the reverse direction of the flow. Present if the flow's **protocolIdentifier** is 6 (TCP) and the flow has a reverse direction.
- initialTCPFlags** CERT (PEN 6871) IE 14, 1 octet
TCP flags of initial packet in the forward direction of the flow. Present if the flow's **protocolIdentifier** is 6 (TCP).
- unionTCPFlags** CERT (PEN 6871) IE 15, 1 octet
Union of TCP flags of all packets other than the initial packet in the forward direction of the flow. Present if the flow's **protocolIdentifier** is 6 (TCP).
- reverseInitialTCPFlags** CERT (PEN 6871) IE 16398, 1 octet
TCP flags of initial packet in the reverse direction of the flow. Present if the flow's **protocolIdentifier** is 6 (TCP) and the flow has a reverse direction.
- reverseUnionTCPFlags** CERT (PEN 6871) IE 16399, 1 octet
Union of TCP flags of all packets other than the initial packet in the reverse direction of the flow. Present if the flow's **protocolIdentifier** is 6 (TCP) and the flow has a reverse direction.
- vlanId** IE 58, 2 octets
802.1q VLAN tag of the first packet in the forward direction of the flow. Present if MAC layer export is enabled and a VLAN tag is present.
- reverseVlanId** Reverse (PEN 29305) IE 58, 2 octets
802.1q VLAN tag of the first packet in the reverse direction of the flow. Present if MAC layer export is enabled, the flow has a reverse direction, and a VLAN tag is present.
- payload** CERT (PEN 6871) IE 18, variable-length
Initial *n* bytes of forward direction of flow payload. Present if payload collection is enabled and payload is present in the forward direction of the flow.
- reversePayload** CERT (PEN 6871) IE 16402, variable-length
Initial *n* bytes of reverse direction of flow payload. Present if payload collection is enabled and payload is present in the reverse direction of the flow.
- payloadEntropy** CERT (PEN 6871) IE 35, 1 octet
Shannon Entropy calculation of the forward payload data.
- reversePayloadEntropy** CERT (PEN 6871) IE 16419, 1 octet
Shannon Entropy calculation of the reverse payload data.

SIGNALS

`yaf` responds to **SIGINT** or **SIGTERM** by terminating input processing, flushing any pending flows to the current output, and exiting. If **—verbose** is given, `yaf` responds to **SIGUSR1** by printing present flow and fragment table statistics to its log. All other signals are handled by the C runtimes in the default manner on the platform on which `yaf` is currently operating.

EXAMPLES

To generate flows from an pcap file into an IPFIX file:

```
yaf --in packets.pcap --out flows.yaf
```

To capture flows from a pcap interface and export them to files in the current directory rotated hourly:

```
yaf --live pcap --in enl --out enl_capture --rotate 3600
```

To capture flows from an Endace DAG card and export them via IPFIX over TCP:

```
yaf --live dag --in dag0 --ipfix tcp --out my-collector.example.com
```

To convert a pcap formatted packet capture and convert that into IPFIX:

```
yaf <packets.pcap >flows.yaf
```

CAVEATS

Input/output behavior has changed compared to YAF 0.6, and its command-line options have changed. Specifically:

- Daemon options are no longer supported. Use `filedaemon`, installed with Airframe, to have `yaf` watch a directory and process files therein; and `airdaemon`, installed with Airframe, to run `yaf` as a daemon with infinite retry (as when exporting to an IPFIX collecting process where connectivity may go down).
- The `--live` option now takes an argument, the type of live capture to run; available types are `pcap` and `dag`.

KNOWN ISSUES

YAF 0.7 does not interoperate with previous versions, because it no longer uses provisional information elements for the reverse direction of a biflow. YAF 0.7 must be used with an IPFIX Collecting Process that uses PEN 29305 for reverse information elements. For export to SiLK, this implies that the SiLK packer or `rwipfix2silk` utility must be built against `libfixbuf 0.7.0` or later.

Presently, the **destinationTransportPort** information element contains ICMP type and code information for ICMP or ICMP6 flows; this is nonstandard and may not be interoperable with other IPFIX implementations.

Bug reports and feature requests may be sent directly to the principal maintainer, Chris Inacio, via email at [<inacio@cert.org>](mailto:inacio@cert.org).

AUTHORS

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

SEE ALSO

yafscii (1), *tcpdump* (1), *pcap* (3), *nafalize* (1), and the following IETF Internet RFCs: Specification of the IPFIX Protocol for the Exchange of IP Traffic Flow Information **RFC 5101**, Information Model for IP Flow Information Export **RFC 5102**, Bidirectional Flow Export using IPFIX **RFC 5103**.