NAME

softflowd — Traffic flow monitoring

SYNOPSIS

softflowd [**-6dDhba**] [**-L** *hoplimit*] [**-T** *track_level*] [**-c** *ctl_sock*] [**-i** [*if_ndx*:]*interface*] [**-m** *max_flows*] [**-n** *host:port*] [**-p** *pidfile*] [**-r** *pcap_file*] [**-t** *timeout_name=seconds*] [**-v** *netflow_version*] [**-s** *sampling_rate*] [bpf_expression]

DESCRIPTION

softflowd is a software implementation of a flow-based network traffic monitor. **softflowd** reads network traffic and gathers information about active traffic flows. A "traffic flow" is communication between two IP addresses or (if the overlying protocol is TCP or UDP) address/port tuples.

The intended use of **softflowd** is as a software implementation of Cisco's NetFlow(tm) traffic account system. **softflowd** supports data export using versions 1, 5, 9 or 10 (a.k.a. IPFIX) of the NetFlow protocol. **softflowd** can also run in statistics-only mode, where it just collects summary information. However, too few statistics are collected to make this mode really useful for anything other than debugging.

Network traffic may be obtained by listening on a promiscuous network interface or by reading stored pcap(3) files, such as those written by tcpdump(8). Traffic may be filtered with an optional bpf(4) program, specified on the command-line as *bpf_expression*. **softflowd** is IPv6 capable and will track IPv6 flows if the NetFlow export protocol supports it (currently only NetFlow v.9 possesses an IPv6 export capability).

softflowd tries to track only active traffic flows. When the flow has been quiescent for a period of time it is expired automatically. Flows may also be expired early if they approach their traffic counts exceed 2 Gib or if the number of flows being tracked exceeds *max_flows* (default: 8192). In this last case, flows are expired oldest-first.

Upon expiry, the flow information is accumulated into statistics which may be viewed using softflowctl(8). If the **-n** option has been specified the flow information is formatted in a UDP datagram which is compatible with versions 1, 5 or 9 of Cisco's NetFlow(tm) accounting export format. These records are sent to the specified *host* and *port*. The host may represent a unicast host or a multicast group.

The command-line options are as follows:

-n *host:port*
: Specify the *host* and *port* that the accounting datagrams are to be sent to. The host may be specified using a hostname or using a numeric IPv4 or IPv6 address. Numeric IPv6 addresses should be enclosed in square brackets to avoid ambiguity between the address and the port. The destination port may be a portname listed in services(5) or a numeric port.

-i [*if_ndx*:]*interface*
: Specify a network interface on which to listen for traffic. Either the **-i** or the **-r** options must be specified.

-r *pcap_file*
: Specify that **softflowd** should read from a pcap(3) packet capture file (such as one created with the **-w** option of tcpdump(8)) file rather than a network interface. **softflowd** processes the whole capture file and only expires flows when *max_flows* is exceeded. In this mode, **softflowd** will not fork and will automatically print summary statistics before exiting.

-p *pidfile*
Specify an alternate location to store the process ID when in daemon mode. Default is `/var/run/softflowd.pid`

-c *ctlsock*
Specify an alternate location for the remote control socket in daemon mode. Default is `/var/run/softflowd.ctl`

-m *max_flows*
Specify the maximum number of flows to concurrently track. If this limit is exceeded, the flows which have least recently seen traffic are forcibly expired. In practice, the actual maximum may briefly exceed this limit by a small amount as expiry processing happens less frequently than traffic collection. The default is 8192 flows, which corresponds to slightly less than 800k of working data.

-t *timeout_name=time*
Set the timeout names *timeout_name* to *time*. Refer to the **Timeouts** section for the valid timeout names and their meanings. The *time* parameter may be specified using one of the formats explained in the **Time Formats** section below.

-d Specify that **softflowd** should not fork and daemonise itself.

-6 Force **softflowd** to track IPv6 flows even if the NetFlow export protocol does not support reporting them. This is useful for debugging and statistics gathering only.

-D Places **softflowd** in a debugging mode. This implies the **-d** and **-6** flags and turns on additional debugging output.

-b Bidirectional mode in IPFIX (-b work with -v 10)

-a Adjusting time for reading pcap file (-a work with -r)

-h Display command-line usage information.

-L *hoplimit*
Set the IPv4 TTL or the IPv6 hop limit to *hoplimit*. **softflowd** will use the default system TTL when exporting flows to a unicast host. When exporting to a multicast group, the default TTL will be 1 (i.e. link-local).

-T *track_level*
Specify which flow elements **softflowd** should be used to define a flow. *track_level* may be one of: "ether" (track everything including source and destination addresses, source and destination port, source and destination ethernet address, vlanid and protocol), "vlan" (track source and destination addresses, source and destination port, vlanid and protocol), "full" (track source and destination addresses, source and destination port and protocol in the flow, the default), "proto" (track source and destination addresses and protocol), or "ip" (only track source and destination addresses). Selecting either of the latter options will produce flows with less information in them (e.g. TCP/UDP ports will not be recorded). This will cause flows to be consolidated, reducing the quantity of output and CPU load that **softflowd** will place on the system at the cost of some detail being lost.

-v *netflow_version*
Specify which version of the NetFlow(tm) protocol **softflowd** should use for export of the flow data. Supported versions are 1, 5, 9 and 10(IPFIX). Default is version 5.

-s *sampling_rate*
Specify periodical sampling rate (denominator).

Any further command-line arguments will be concatenated together and applied as a bpf(4) packet filter. This filter will cause **softflowd** to ignore the specified traffic.

Timeouts

softflowd will expire quiescent flows after user-configurable periods. The exact timeout used depends on the nature of the flow. The various timeouts that may be set from the command-line (using the **-t** option) and their meanings are:

general
: This is the general timeout applied to all traffic unless overridden by one of the other timeouts.

tcp
: This is the general TCP timeout, applied to open TCP connections.

tcp.rst
: This timeout is applied to a TCP connection when a RST packet has been sent by one or both endpoints.

tcp.fin
: This timeout is applied to a TCP connection when a FIN packet has been sent by both endpoints.

udp
: This is the general UDP timeout, applied to all UDP connections.

maxlife
: This is the maximum lifetime that a flow may exist for. All flows are forcibly expired when they pass *maxlife* seconds. To disable this feature, specify a *maxlife* of 0.

expint
: Specify the interval between expiry checks. Increase this to group more flows into a NetFlow packet. To disable this feature, specify a *expint* of 0.

Flows may also be expired if there are not enough flow entries to hold them or if their traffic exceeds 2 Gib in either direction. softflowctl(8) may be used to print information on the average lifetimes of flows and the reasons for their expiry.

Time Formats

softflowd command-line arguments that specify time may be expressed using a sequence of the form: *time*[*qualifier*], where *time* is a positive integer value and *qualifier* is one of the following:

<none> seconds
s | **S** seconds
m | **M** minutes
h | **H** hours
d | **D** days
w | **W** weeks

Each member of the sequence is added together to calculate the total time value.

Time format examples:

600 600 seconds (10 minutes)
10m 10 minutes
1h30m 1 hour 30 minutes (90 minutes)

Run-time Control

A daemonised **softflowd** instance may be controlled using the softflowctl(8) command. This interface allows one to shut down the daemon, force expiry of all tracked flows and extract debugging and summary data. Also, receipt of a SIGTERM or SIGINT will cause **softflowd** to exit, after expiring all flows

(and thus sending flow export packets if **-n** was specified on the command-line). If you do not want to export flows upon shutdown, clear them first with `softflowctl`(8) or use `softflowctl`(8) 's "exit" command.

EXAMPLES

softflowd -i fxp0

This command-line will cause **softflowd** to listen on interface fxp0 and to run in statistics gathering mode only (i.e. no NetFlow data export).

softflowd -i fxp0 -n 10.1.0.2:4432

This command-line will cause **softflowd** to listen on interface fxp0 and to export NetFlow v.5 datagrams on flow expiry to a flow collector running on 10.1.0.2 port 4432.

softflowd -v 5 -i fxp0 -n 10.1.0.2:4432 -m 65536 -t udp=1m30s

This command-line increases the number of concurrent flows that **softflowd** will track to 65536 and increases the timeout for UDP flows to 90 seconds.

softflowd -v 9 -i fxp0 -n 224.0.1.20:4432 -L 64

This command-line will export NetFlow v.9 flows to the multicast group 224.0.1.20. The export datagrams will have their TTL set to 64, so multicast receivers can be many hops away.

softflowd -i fxp0 -p /var/run/sfd.pid.fxp0 -c /var/run/sfd.ctl.fxp0

This command-line specifies alternate locations for the control socket and pid file. Similar command-lines are useful when running multiple instances of **softflowd** on a single machine.

FILES

`/var/run/softflowd.pid`

This file stores the process ID when **softflowd** is in daemon mode. This location may be overridden using the **-p** command-line option.

`/var/run/softflowd.ctl`

This is the remote control socket. **softflowd** listens on this socket for commands from `softflowctl`(8). This location may be overridden using the **-c** command-line option.

BUGS

Currently **softflowd** does not handle maliciously fragmented packets properly, i.e. packets fragemented such that the UDP or TCP header does not fit into the first fragment. It will product correct traffic counts when presented with maliciously fragmented packets, but will not record TCP or UDP port information. Please report bugs in softflowd to https://github.com/irino/softflowd/issues

AUTHORS

Damien Miller <djm@mindrot.org>
Hitoshi Irino (current maintainer) <irino@sfc.wide.ad.jp>

SEE ALSO

`softflowctl`(8), `tcpdump`(8), `pcap`(3), `bpf`(4)

`http://www.ietf.org/rfc/rfc3954.txt`
`http://www.cisco.com/en/US/products/sw/netmgtsw/ps1964/products_implementation_des`