Table of Contents

Chapter 5. NetFlow.. 1
5.1. Overview of NetFlow and Flow-Tools.. 1
5.2. What NetFlow Can Help You Do.. 1
5.3. How NetFlow Works... 2
5.4. Installing Flow-Tools.. 4
5.5. Configuring NetFlow on the Router.. 4
5.6. Using Flow-Tools.. 5
5.7. References and Further Study.. 20

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

Chapter 5. NetFlow

5.1. Overview of NetFlow and Flow-Tools

The traffic analysis tools described so far all produce quantitative information about network traffic: the amount of bandwidth used or the number of transmitted packets per second, for example. However, it is often necessary to have a more qualitative view of traffic. If your network is attacked by a flood of packets, you would like to know something about the data in the packets. You need to know at least enough to block the traffic from your network.

NetFlow is a feature available on some routers that will allow you to view this information. It includes data such as the source and destination IP addresses, source and destination protocol port numbers, number of packets transmitted, number of bytes transmitted, and much more. Once NetFlow is enabled, the information can be viewed on the router itself or it can be sent to another host on the network for more detailed collection and analysis. NetFlow was originally implemented by Cisco and therefore is available on Cisco routers as well as the Cisco Catalyst 5000 switch if it is installed with a special board. Juniper routers now also have the ability to export packet data in the same NetFlow format as Cisco.

An excellent set of open source tools for collecting and processing NetFlow data is the Flow-Tools package, written by Mark Fullmer and available from Ohio State University. It includes utilities for collecting flows on a server, storing the results, and printing and manipulating flows as well as tools for producing reports based on the data.

5.2. What NetFlow Can Help You Do

Imagine your network is hit by a denial of service attack. The first thing you notice is that your network has degraded connectivity to the rest of the Internet. The interface counters on your border router indicate a very high rate of traffic, and when you examine the MRTG graph, you see a sudden, dramatic rise in traffic levels. This all points to a possible denial of service attack. Turning to NetFlow, you examine the traffic in real time and notice a large number of connections from a single host, all to sequentially increasing IP addresses inside your network:

```
srcIP          dstIP        prot srcPort dstPort octets packets
10.194.158.201 10.36.1.21   6    80      32966   466    1
10.54.59.138   10.209.0.60  17   32781   22      165    3
10.54.59.138   10.209.0.61  17   32781   22      165    3
10.54.59.138   10.209.0.62  17   32781   22      165    3
10.89.67.212   10.225.0.86  6    1751    1214    652    6
10.54.59.138   10.209.0.63  17   32781   22      165    3
10.54.59.138   10.209.0.64  17   32781   22      165    3
10.54.59.138   10.209.0.65  17   32781   22      165    3
10.54.59.138   10.209.0.66  17   32781   22      165    3
10.54.59.138   10.209.0.67  17   32781   22      165    3
10.54.59.138   10.209.0.68  17   32781   22      165    3
10.226.244.82  10.215.0.200 17   6257    6257    309    4
```

Realizing this is someone scanning your network, you can now block the traffic at the border router and notify the network administrators at the remote site.

5.3. How NetFlow Works

The following sections explore NetFlow in detail.

5.3.1. Flows

NetFlow is based on the idea of a **flow** of network traffic. What is a flow? It is what you would naturally think of as one full network conversation. The Transmission Control Protocol (TCP) connection a workstation opens to retrieve a Web page is a flow. The start of the TCP connection is the beginning of the flow, and when the connection is closed, the flow is complete. Another example of a flow would be the series of ICMP packets that make up a ping test. The flow begins with the first ICMP packet and ends with the last.

Every flow must have a unique set of the following:

- Source IP address
- Destination IP address
- Source port
- Destination port
- IP protocol number
- Type of service field
- Input interface

These values must remain the same for the entire flow. Retrieving multiple Web pages over a single TCP connection, for example, is all one flow. But if multiple TCP connections are made to the same Web server, different port numbers will be used for each connection, so each connection will be a different flow. Similarly, an attacker who is scanning destination IP addresses on your network will generate many flows: one for each source/destination pair.

Also note that a flow is unidirectional. That is, one of the addresses is only the source of traffic and the other is only the destination. This means that a flow represents just the traffic moving in one direction. In communications with a Web server there are really two flows present: one is the traffic from the workstation to the server, and the other is the traffic from the server to the workstation. The router will always report flows for traffic that enters an interface, not traffic that leaves an interface. So if NetFlow is enabled on the interface where your workstation resides, your workstation will be the source address in any flow data. If NetFlow is also enabled on interfaces where another machine is sending data to your workstation, the router will then report your workstation as the destination address for those flows.

Of course, it is not always easy to determine what constitutes one session's worth of traffic and therefore when a particular flow ends. For a TCP connection, it is relatively easy; the session begins with a TCP SYN and ends with a TCP FIN. For UDP or Internet Control Message Protocol (ICMP), it is more difficult, and the router must use heuristics to decide when the flow is complete.

Because the router has only a limited amount of memory available for holding flow data, it must occasionally remove flows from the cache in order to make room for new ones. For this reason, it is important to realize that when a router reports on a flow, it may not really be the entire conversation as expected. For example, one flow may represent the first part of a TCP connection, another two flows might be the middle, and yet another flow, the end. Or it is just as possible for an entire TCP connection to be reported in a single flow. It depends on whether the router needs to free up resources. By default, a Cisco router will expire a flow when one of the following occurs:

- The end of a TCP connection is found.
- No traffic has been present in the flow for 15 seconds.
- The flow has been running for over 30 minutes.
- The table of flows in the router is filled.

5.3.2. NetFlow and Switching Paths

NetFlow is sometimes called NetFlow Accounting or NetFlow Switching. Both are acceptable terms, but one very prevalent myth should be dispelled about the latter. NetFlow is *not* a switching path. That is, it is not a system the router uses to figure out which packets are sent to which interface. NetFlow always relies on one of the other switching paths, such as Cisco Express Forwarding (CEF) or optimum switching, to make these decisions. NetFlow does interact with the switching path, however. Decisions on whether a packet matches an access-list filter are optimized so that the decision is made only once per flow, instead of for every packet in the flow. This practice also has an impact on access-list counters, which will no longer represent the number of packets matched when used in conjunction with NetFlow.

5.3.3. Exporting NetFlow Data

Routers can export NetFlow data so that it can be received by a server and then processed in any way the network administrator wishes. The host or software that receives the data is called a **flow collector**. The NetFlow data is exported in UDP packets to an IP address and UDP port number that are configured on the router. Because routers send the exported data using UDP, it is possible for packets of NetFlow data to be lost. As described below, version 5 of NetFlow introduces sequence numbers so that if data is lost, the flow collector will at least be aware of the loss. There is, however, no way to retrieve the lost data.

On a Cisco router, only one IP address can be specified as the recipient of exported flows, so if you need to send NetFlow data to more than one collector, it will have to be forwarded from one collector to the others. Note that one piece of data that is *not* included in a flow is the router from which the flow was sent. Typically, a collector knows which router the flow came from based on the source IP address of the exported packets. However, if a flow is forwarded from one collector to another, this data may be lost unless the forwarder either specially includes the information via some other mechanism or forges the source address of the forwarded packets to be that of the original router.

5.3.4. NetFlow Versions

There are several different version of NetFlow available. Version 1 was the first, and each flow contained the following information:

- Source IP address
- Destination IP address
- Source port
- Destination port
- Next hop router address
- Input interface number
- Output interface number
- Number of packets sent
- Number of bytes sent
- sysUpTime when flow began
- sysUpTime when flow ended
- IP protocol number
- IP type of service
- Logical OR of all TCP flags seen

There are currently four other versions of NetFlow:

- **Version 5.** Includes Autonomous System (AS) numbers from Border Gateway Protocol (BGP) if available and also includes sequence numbers, which allows checking for lost packets.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

- **Version 7.** Used only on the Cisco Catalyst 5000 product series, which also requires a special NetFlow Feature Card in order to perform NetFlow accounting.
- **Version 8.** Introduces Router-Based NetFlow Aggregation, a feature that can greatly reduce the amount of data sent when flows are exported from the router. By choosing an appropriate aggregation scheme, you can instruct the router to group flows with similar data into a single aggregate flow that is reported on behalf of all the others.
- **Version 9.** Introduces a template-based scheme for reporting flows, which allows a client receiving exported flow data to process the data without necessarily having prior knowledge of what kinds of data will be present.

5.4. Installing Flow-Tools

The OSU Flow-Tools package is available from http://www.net.ohiostate.edu/software/. Download the latest stable version (or an even later version if it suits you), then unpackage and build the software:

```
Linux% gunzip -c flow-tools-0.59.tar.gz | tar xvf -
Linux% cd flow-tools-0.59
Linux% ./configure
Linux% make
```

It should build without difficulty. Once it is built, you can install it on your system from a root account with:

```
Linux# make install
```

This will install everything in `/usr/local/netflow/`. The programs will be in `/usr/local/netflow/bin/`, which you can add to your path or you can run the programs with a full path name. For the examples in this chapter, it is assumed that the programs are in your path.

5.5. Configuring NetFlow on the Router

Before you can use the Flow-Tools programs to view flow data, you must configure the router to use NetFlow on at least one interface, and you must also configure the router to export flows to a host that will be running the Flow-Tools software. On a Cisco router, you can enable NetFlow on an interface with the interface configuration command `ip route-cache flow`. For example:

```
router#config term
Enter configuration commands, one per line. End with CNTL/Z.
router(config)#int Ethernet1/2
router(config-if)#ip route-cache flow
router(config-if)#end
```

Remember that NetFlow will report on packets that enter an interface and not on packets that leave an interface. You can enable NetFlow on as many interfaces as you like, though be aware that if you export flow data, more interfaces means more reporting traffic that will be sent to the collector.

To instruct the router to export flows, use the `ip flow-export destination` configure command:

```
router(config)#ip flow-export destination 192.0.2.5 9995
router(config)#ip flow-export source Loopback0
```

This will send all flow data to the IP address 192.0.2.5 on UDP port 9995. In this case, we also specified an interface to be used as the source of the packets. This is optional; it is configured here to use a Loopback interface so that the source address of the flows will be consistent even if the other interface addresses on the router change.

Finally, you can specify which version of NetFlow the router should export. If you do not explicitly specify a version, it will default to version 1. To export version 5 instead, type:

```
router(config)#ip flow-export version 5
```

Remember to issue a `write mem` to save configuration changes. Now the router should be sending NetFlow packets to the address specified in the `ip flow-export destination` command, and you can use the Flow-Tools programs to receive them.

5.6. Using Flow-Tools

A number of programs make up the Flow-Tools software package. Figure 5.1 lists each program and its function. All major tools are described here, categorized into groups of tools that capture flows, tools that view flow data, and tools that manipulate flows.

Figure 5.1. Programs Included with Flow-Tools.

Program	**Function**
flow-receive	Receive flows, send to stdout
flow-capture	Receive flows, store to disk
flow-print	Print flow data to the screen
flow-report	Produce flow reports for other programs
flow-stat	Print flow statistics to the screen
flow-dscan	Detect suspicious network traffic
flow-cat	Concatenate flow data files
flow-merge	Sort and concatenate flow data files
flow-expire	Remove old flow data files
flow-split	Split data files into smaller files
flow-header	Print meta data on a capture session
flow-fanout	Redistribute UDP exports to other addresses

Program	Function
flow-send	Send flow data in NetFlow format
flow-tag	Add user-defined tags to flows
flow-filter	Filter flows
flow-gen	Generate test flow data
flow-import	Import data from other NetFlow tools
flow-export	Export data to other NetFlow tools
flow-xlate	Translate flow data

5.6.1. Capturing Flows

There are two programs in the Flow-Tools distribution that receive flow data. One is `flow-capture`, which not only receives the flows, but also stores them to files on disk, rotates the files, and ensures that they do not grow larger than a limit you specify on the command line. The other program is `flow-receive` which sends the flow data to the standard output instead of storing flows on disk.

Note that the output from these commands, whether sent to the standard output or to a file on disk, is not a direct dump of NetFlow data in the format produced by the router. The Flow-Tools package stores flow data in its own format, which, among other things, allows it to easily store data from different versions of NetFlow in a single format.

In the following examples, both the flow-capture and the flow-receive programs will be run from the root account. This is not required; the tools can be run from a user level account instead, but with two caveats. First, if you wish to listen for flows on a port that is number 1024 or lower, you will not be able to do it from a user-level account. You must either modify your router configuration to send flows to a higher numbered port or run the tools from a root account. Second, `flow-capture` will attempt to write a file in `/var/run` containing the process ID of the program. If you are not root, you may not be able to write to this file, depending on your system configuration. However you can change this behavior with the `-p` option described below.

Try using the `flow-receive` program in conjunction with the `flow-print` program to check if flows are being received properly from the router:

```
Linux# flow-receive 0/0/9995 | flow-print
flow-receive:  setsockopt(size=262144)
flow-receive:  New exporter: time=1048976578 src_ip=10.255.255...
flow-receive:  New exporter: time=1048976578 src_ip=10.255.255...
flow-receive:  New exporter: time=1048976578 src_ip=10.255.255...
srcIP          dstIP         prot srcPort dstPort octets packets
10.26.196.41   10.221.0.157  17   2706    1497    35     1
10.175.106.3   10.221.0.157  6    4028    1497    81     2
10.132.140.22  10.253.1.161  17   1035    4665    46     1
10.253.1.161   10.132.140.22 1    0       771     74     1
```

```
10.67.8.175    10.203.0.180  6    47813   3150    81     2
10.253.1.191   10.246.38.98  17   65145   6671    138    3
10.24.233.164  10.221.0.157  17   1500    1497    35     1
10.119.5.60    10.243.0.64   17   33487   33487   53     1
10.181.135.108 10.236.0.107  6    80      2990    40     1
10.253.5.165   10.19.139.162 6    1680    80      647    4
10.88.171.11   10.235.0.166  17   1256    3813    35     1
10.253.1.191   10.213.33.193 17   65468   4665    46     1
10.61.67.199   10.253.2.21   17   6257    6257    317    4
```

This program will continue to run, filling your screen with output, until you break it with Control-C.

The three "new exporter" lines at the beginning are printed to the standard error by `flow-receive` and inform you every time the program detects a new device sending flows. The slash-delimited argument to `flow-receive` is a generic syntax that other Flow-Tools programs use as well. The format is *localip*/*remoteip*/*port*. In this case, *localip* is the local IP address that the program should listen on,[1] *remoteip* is the IP address that flows must be received from, and *port* is the local port the program will listen on to receive flows. In either of the address fields, a zero indicates that any address is acceptable. So in our example, `0/0/9995` means the program will accept flows to any local IP address, coming from any IP address, and destined for port 9995. Compare this with the following:

[1] Some machines have more than one address. If a zero is used for the local address, any of the addresses can be used; otherwise, the program will use the specific address listed, and flows received at other IP addresses on the machine will be ignored.

```
Linux# flow-receive 0/10.255.255.16/9995 | flow-print
flow-receive: setsockopt(size=262144)
flow-receive: Unexpected PDU: src_ip=10.255.255.15 not configured
flow-receive: New exporter: time=1048977556 src_ip=10.255.255...
flow-receive: Unexpected PDU: src_ip=10.255.255.15 not configured
flow-receive: Unexpected PDU: src_ip=10.255.255.22 not configured
srcIP          dstIP        prot srcPort dstPort octets packets
10.176.155.249 10.216.0.132 17   6257    6257    388    4
10.182.153.110 10.133.0.180 6    2164    3150    122    3
10.97.215.204  10.252.2.21  17   6257    6257    316    4
10.188.11.57   10.36.5.66   6    5190    1037    132    2
10.137.199.176 10.216.1.123 6    1594    1344    13132  316
10.188.11.236  10.221.0.105 6    5190    2428    489    4
```

Here flows must be received from a device with the IP address 10.255.255.16. There is still a message to standard error when flows from this source are found, but there are also warnings about flows unexpectedly coming from other addresses. Flows from those other addresses are ignored. Note that the addresses checked here are those of the routers or other devices sending and collecting the flows, not the addresses contained within the flows.

It is convenient to use `flow-print` to visualize flows on the screen, and it can even be useful in diagnosing an ongoing network problem, but for typical use, you will want to store flow data on disk. This is the job of the `flow-capture` program. Not only will `flow-capture` store flow data in files, it will also compress it, make sure to separate it into reasonably sized files, and rotate the files to keep the total amount of used disk space under control.

There are many options to `flow-capture`, but a simple invocation would be:

```
Linux# mkdir /var/tmp/flows
Linux# flow-capture -w /var/tmp/flows 0/0/9995
```

This example demonstrates the only required options. The path following the `-w` flag is the name of the directory in which flow files are to be stored. The program will not create it for you, and if it does not exist, it will silently exit. The slash-delimited argument at the end is the port on which flows should be received, using the same syntax as the `flow-receive` program.

Once started, `flow-capture` will run in the background. Files will be named according to the date they are created. If a file is currently being created (data is still actively being added to it), the name will begin with `tmp-`; otherwise, the file name will begin with the prefix `ft-`.

By default, `flow-capture` will:

- Create a new file every 15 minutes
- Create a nested directory structure so that each year, month, and day of files has a separate directory
- Never remove files (the total amount of data will grow without bound)
- Compress files at a medium compression level

All of these settings can be changed, and they often are. The command line arguments for changing these settings are listed below, along with a few other important options. All of the options to `flow-capture` are listed in Figure 5.2.

Figure 5.2. Options to the Flow-Capture Program.

Flag	**Function**
-A	Replace 0 AS values
-b	Specify output byte order
-c	Enable TCP client connections
-C	Add a comment
-d	Turn on debugging
-e	Maximum number of files
-E	Maximum data size to store
-f	File name of filter list
-F	Specify active filter
-h	Display help
-m	Use a privacy mask
-n	Specify number of new files per day
-N	Set nesting level
-p	Specify PID file
-R	Execute a program after rotation

Flag	Function
-S	Syslog timestamps at specified interval
-t	File to load tags from
-T	Tags to use
-V	Specify output format
-w	Specify directory to store data files in
-z	Set compression level

Bounding the Data Size

The `-E` argument allows you to specify the maximum size that data files may use in total. You can specify the size in bytes, kilobytes, megabytes, or gigabytes by using the letters b, K, M, or G, respectively. For example:

```
Linux# flow-capture -w /var/tmp/flows -E 1G 0/0/9995
```

would ensure that the data does not exceed 1 gigabyte of disk storage. Flow-capture accomplishes this by deleting the older data files. The program uses a time stamp stored in these files so that you need not worry about copied files being deleted or overlooked for deletion because the Unix time stamp is incorrect.

Also note that only files named with the Flow-Tools naming convention will be deleted. If you rename the file `ft-v01.2003-03-30.135300-0500` to `flow.save`, it will not be cleaned up as usual.

Changing Directory Nesting

By default, the data files are nested into a hierarchical directory structure by year, month, and day. For example:

```
Linux# find /var/tmp/flows
/var/tmp/flows
/var/tmp/flows/2003
/var/tmp/flows/2003/2003-03
/var/tmp/flows/2003/2003-03/2003-03-29
/var/tmp/flows/2003/2003-03/2003-03-29/tmp-v01.2003-03-29.190000-0500
/var/tmp/flows/2003/2003-03/2003-03-29/ft-v01.2003-03-29.184647-0500
```

There is a directory for the year 2003, one below that for the month of March 2003, and one below that for the day March 29, 2003. In that directory are two flow files, one being created and one already completed. If this level of nesting does not suit you, there are six other options available, each represented by a number. From the `flow-capture` man page:

```
-3   YYYY/YYYY-MM/YYYY-MM-DD/flow-file
-2   YYYY-MM/YYYY-MM-DD/flow-file
-1   YYYY-MM-DD/flow-file
 0   flow-file
 1   YYYY/flow-file
 2   YYYY/YYYY-MM/flow-file
 3   YYYY/YYYY-MM/YYYY-MM-DD/flow-file
```

Though the documentation states that the default nesting level is 0, it is 3 in every version I have tested. The nesting depth is controlled with the `-N` flag. If you wish to run `flow-capture` so that no nesting is performed and all files are stored in a single, flat directory, type:

```
Linux# flow-capture -w /var/tmp/flows -N 0 0/0/9995
```

Changing the File Rotation Rate

By default, `flow-capture` will write data into a file for 15 minutes and then close it and create a new file. This is a good way to break up the data into files that you can work with easily. There are also other programs in the Flow-Tools distribution that allow you to merge these files or separate them into even smaller files.

If you wish to change the interval at which new files are created, you can use the `-n` flag. The argument it takes is the number of times per day that `flow-capture` should create a new file. The default is 95. That is, 95 times a day it will create a new file, which is 96 files total. If you want to create a file for each hour, you need to ask for a new file to be created 23 times each day, as in:

```
Linux# flow-capture -w /var/tmp/flows -n 23 0/0/9995
```

Changing the Process ID File

Because `flow-capture` runs as a daemon, it stores its process ID in a file for the convenience of administrators or other programs that may wish to send it a signal. By default, this file is `/var/run/flow-capture.pid`. You may not be able to write to this directory if you do not run the program from a root account. In this case, you may specify a different file with the `-p` option:

```
Linux# flow-capture -w /var/tmp/flows -p /var/tmp/pid 0/0/9995
```

If you use the filename "`-`", no PID file will be created.

Using Compression

`Flow-capture` compresses flow data so that it takes up less space on disk. While taking up less disk space is an advantage, the trade-off is that it takes computational power to perform the compression. If your server is short on disk space but has lots of processing power, you may wish to use a higher level of compression than the default. On a server with lots of disk space and a slow CPU, you may want less compression or perhaps none at all. The compression level is a value from 0 to 9, where 0 is no compression and 9 is the highest level of compression. The default level is set by the zlib library, typically to 6. If you wish to disable compression, use:

```
Linux# flow-capture -w /var/tmp/flows -z0
```

Killing Flow-Capture

If you need to stop `flow-capture`, you can send it a QUIT signal, as in:

```
Solaris# ps -ef | grep flow-capture
    root 10858     1  0 14:20:06 ?    0:00 flow-capture -w /var...
Solaris# kill -QUIT 10858
```

This will cause `flow-capture` to properly close and rename the data file that it is currently writing. If you kill `flow-capture` without specifying a signal (which defaults to the TERM signal), the file will be left with its temporary name. All of the Flow-Tools daemons write a file that contains the process ID of the running program. Instead of using `ps` to look up the process ID, we could have done:

```
Solaris# kill -QUIT 'cat /var/run/flow-capture.pid.9995'
```

The 9995 on the end of the filename indicates that this `flow-capture` is running on port 9995, in case more than one copy of `flow-capture` is running at once.

Note that if a router is sending flow data to a port and there is no program like `flow-capture`, `flow-receive`, or `flow-fanout` listening on that port, the server will respond with ICMP port unreachable messages back to the router. While there is no harm in sending or receiving a small number of these messages, a large number sent at a rapid rate could have an operational impact either on network bandwidth or on the CPU. If you plan to leave `flow-capture` down for a long time, you may want to turn off the source of the flows or run `flow-receive` and send the data to `/dev/null`.

If you would like to force `flow-capture` to stop writing to the current file, close it, and start a new file, you can send it a HUP signal:

```
Solaris# ps -ef | grep flow-capture
    root 10862     1  1 14:24:01 ?  0:00 flow-capture -w /var...
Solaris# kill -HUP 10862
```

Allowing Remote Clients

The `-c` option to `flow-capture` will allow a remote client to make a TCP connection to the program to receive flow data. Currently, the implementation does not work very well; the connection is often closed unexpectedly. With luck this will be fixed in a future release. Also note that if you do not use an external wrapper, *any* client connecting to the server can receive access to your NetFlow data. The number listed after the `-c` simply specifies the maximum number of clients that can connect at any one time:

```
Server# flow-capture -w /var/tmp/flows -c 10 0/0/9995
Client% nc server.example.com 9995 | flow-print
srcIP          dstIP         prot  srcPort dstPort octets packets
10.254.70.105  10.61.171.218 6     3531    139     144    3
10.250.226.251 10.154.0.230  6     4662    1089    1218   24
10.50.61.48    10.91.178.156 17    1026    137     78     1
10.87.169.94   10.91.190.251 17    1025    137     78     1
```

The nc program is described in Chapter 9, Basic Tools. An alternative to allowing remote clients to receive flow data via TCP is to forward UDP flows to them using `flow-fanout`, as described in the section on manipulating flow data.

5.6.2. Viewing Flow Data

Flow-Tools includes four programs for viewing flow data. They are:

- **Flow-print.** Prints flow data in ASCII. A number of predefined reporting formats are available.
- **Flow-report.** Generates reports with comma-separated field values, suitable for feeding into another program to produce graphs or to perform other data processing.
- **Flow-stat.** Available only in later versions of Flow-Tools. This program generates reports based on statistics of interest to the administrator, such as high bandwidth use by port.
- **Flow-dscan.** Analyzes flow data to find suspicious network traffic such as port scans. The program notifies the user if any are found.

Flow-Print

Earlier we saw how `flow-print` could be used to examine data received in real time from `flow-receive`. Now you can feed it flow data stored in files from `flow-capture`:

```
Solaris# cat ft-v01.2003-03-30.133000-0500 | flow-print
srcIP          dstIP          prot srcPort dstPort octets packets
10.109.125.64 10.228.96.175 6    80      1862    35879  26
10.188.165.57 10.202.1.99   6    80      4633    695    4
10.73.86.70   10.243.41.92  6    80      1188    218    3
10.130.91.7   10.249.5.139  6    80      1458    596    5
10.210.86.68  10.226.112.52 6    3456    60263   40     1
10.17.247.8   10.252.25.243 6    80      33135   633    3
```

If you use the `-n` flag, `flow-print` will print symbolic names wherever possible:

```
Solaris# cat ft-v01.2003-03-30.133000-0500 | flow-print -n
srcIP          dstIP          prot srcPort dstPort octets packets
10.109.125.64  10.228.9.175  tcp  http    1862    35879  26
10.223.91.7    10.249.50.139 tcp  http    nrcabq- 596    5
10.141.51.240  10.153.55.151 tcp  Gnutell 3244    120    3
10.124.2.66    10.53.110.86  udp  3283    3283    33     1
10.143.66.23   10.53.212.40  udp  1297    ms-sql- 404    1
10.148.139.199 10.205.16.115 tcp  eDonkey 3913    144    3
10.85.4.150    10.205.12.92  tcp  imaps   49332   167    1
10.172.39.24   10.243.43.218 tcp  http    1310    11689  11
10.19.139.162  10.252.7.65   tcp  http    1307    40     1
10.73.86.70    10.243.0.92   tcp  http    1191    4505   5
```

Here you can see the IP protocol name has been printed, and when a port number is a well-known service, the name instead of the number is printed.

In addition to the simple output format above, the `flow-print` program has over 20 other formats available. The desired format can be specified with the `-f` option. For example:

```
Solaris# cat ft-v01.2003-03-30.133000-0500 | flow-print -n -f 5
Start             End               Sif   SrcIPaddress    SrcP  DIf  DstIPaddr...

0330.13:29:56.046 0330.13:29:56.398 3     10.109.125.64   80    40   10.228.0....
0330.13:29:56.046 0330.13:29:56.098 3     10.188.165.57   80    41   10.202.1....
0330.13:29:56.058 0330.13:29:56.198 3     10.73.86.70     80    30   10.243.0....
0330.13:29:56.082 0330.13:29:56.650 3     10.130.91.7     80    7    10.249.0....
0330.13:29:56.086 0330.13:29:56.086 3     10.210.86.68    3456  42   10.226.0....
0330.13:29:56.098 0330.13:29:57.846 3     10.17.247.8     80    10   10.252.2....
```

The formats, as described in the Flow-Tools documentation, are listed in Figure 5.3. Note that some formats print two lines per flow while others print only one.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

Figure 5.3. Formats for `Flow-Print`.

No.	Format
0	1 line, interfaces, hex ports
1	2 line (includes timing and flags)
2	2 line candidate TCP syn attack flows
3	1 line, no interfaces, decimal ports
4	1 line with AS number
5	1 line, 132 column
6	show ip accounting emulation
7	1 line, 132 column +router_id
8	1 line, 132 column +encapsulation
9	1 line with tag values
10	AS aggregation
11	Protocol Port aggregation
12	Source Prefix aggregation
13	Destination Prefix aggregation
14	Prefix aggregation
15	Destination aggregation (Catalyst)
16	Source Destination aggregation (Catalyst)
17	Full Flow (Catalyst)
18	ToS AS Aggregation

No.	Format
19	ToS Proto Port aggregation
20	ToS Source Prefix aggregation
21	ToS Destination Prefix aggregation
22	ToS Prefix Aggregation
23	ToS Prefix Port aggregation

Flow-Report

While `flow-print` produces output that is easy to read on the screen, it is not particularly well suited for other use. If you wish to process the data in some automated way, be it performing a statistical analysis or simply importing the data into a graphing program, you will want to use the `flow-report` program to produce data in a simple comma-separated format.

In order to produce a flow report, you must first create a configuration file that specifies a few options. The following sample configuration file can be placed in `/var/tmp/report.conf`:

```
stat-report myreport
  type ip-source/destination-address/ip-source/destination-port
  output
    records 50
    sort +octets

stat-definition stat1
  report myreport
```

The first section defines a report called myreport. The next line specifies which general type of report it should be. This is a required line and the value must be one of the 80 or so valid report types available. The names for all of them are listed in the `flow-report` Web page and man page that comes with the Flow-Tools distribution.

Next, the output section of the report lists options associated with the output. Here we have specified that a maximum of 50 records should be printed. After the first 50, the program will end the report. Next, the `sort +octets` line asks for the report to be sorted by octets, with the largest numbers first.

Run the report with:

```
Solaris# flow-report -s /var/tmp/report.conf \
   -Sstat1 < ft-v01.2003-03-30.153000...
# recn: ip-source-address,ip-destination-address,ip-source-po...
10.116.200.35,10.1.14.3,50100,50100,15,1544300277,1351782,3613224
10.116.200.34,10.1.14.1,50000,50000,14,1499274579,1244856,3612348
10.48.78.93,10.45.17.174,1058,5004,14,724389220,626401,3606776
10.252.12.21,10.96.1.28,445,1424,3,162981116,110004,905944
10.116.200.35,10.1.14.4,17100,17100,15,102237923,131847,3611996
10.116.200.34,10.1.14.2,17000,17000,15,102097558,131666,3611024
10.127.126.7,10.203.0.70,3519,59,3,78775788,74993,904048
```

```
10.240.64.63,10.2.213.188,32949,21088,18,74854576,57740,4514472
10.181.50.25,10.209.0.48,119,1038,4,74036545,70162,959184
```

The first line of output, beginning with a comment character, lists the order of the fields for the rest of the report. Here the first field is the source IP address of the flow, the second is the destination IP address, and so on. After that, each line is a line of data. Note that each line does not correspond to a single flow. For this report, each line corresponds to one unique combination of IP source address, IP destination address, source port, and destination port. This set of data objects is called the **key** for the report. The values for octets and packets listed on each line are the cumulative values for all flows matching a particular key.

There are a number of useful options to the `stat-report` and `output` commands in the configuration file. For example, you can display additional data such as the number of packets per second transmitted. You can also choose to view data such as the byte and packet counts as percentages of all observed data instead of viewing the raw numbers. All of the options are listed in the documentation.

Flow-Stat

The `flow-stat` program is available only in later versions of Flow-Tools. It is a convenient way to produce the kind of data that `flow-report` generates, but it does not require a configuration file, and it prints the results in a readable format. Like `flow-print`, it uses a number of predefined formats. Here is an example of one of the many available reports. This one lists traffic by port number:

```
Solaris# flow-stat -f7 -w -P -S2 < ft-v01.2003-03-30.153000-0500
#  --- ---- ---- Report Information --- --- ---
#
# Fields:   Percent Total
# Symbols:  Disabled
# Sorting:  Descending Field 2
# Name:     UDP/TCP port
#
# Args:     flow-stat -f7 -w -P -S2
#
#
# port          flows   octets   packets   duration
#
50100           0.007   18.431   8.360     0.208
50000           0.007   17.893   7.695     0.159
5004            0.036   5.015    3.119     1.190
80              6.637   4.421    2.939     3.105
1058            0.044   4.323    1.937     0.066
1214            2.690   3.438    1.775     3.560
4662            2.214   2.194    1.899     4.257
56464           0.226   2.178    12.750    6.706
17100           0.007   1.223    0.829     0.212
17000           0.005   1.220    0.821     0.160
445             0.253   1.037    0.924     0.188
1424            0.011   0.994    0.526     0.044
5005            0.075   0.989    3.856     2.385
```

The `-f7` argument designates the format type. Other possible formats are listed in Figure 5.4. The `-w` flag specifies that wide columns can be used, `-P` asks for values to be reported as percentage of the total, and `-S2` causes the rows to be sorted by the second data column, in this case the octets column. In this example, traffic to or from port 50100 or 50000 is taking up 36.3 percent of the total bandwidth in use.

Figure 5.4. Formats for `Flow-Stat`.

No.	Format
0	Overall Summary

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

No.	Format
1	Average packet size distribution
2	Packets per flow distribution
3	Octets per flow distribution
4	Bandwidth per flow distribution
5	UDP/TCP destination port
6	UDP/TCP source port
7	UDP/TCP port
8	Destination IP
9	Source IP
10	Source/Destination IP
11	Source or Destination IP
12	IP protocol
13	Octets for flow duration plot data
14	Packets for flow duration plot data
15	short summary
16	IP Next Hop
17	Input interface
18	Output interface
19	Source AS

No.	Format
20	Destination AS
21	Source/Destination AS
22	IP ToS
23	Input/Output Interface
24	Source Prefix
25	Destination Prefix
26	Source/Destination Prefix
27	Exporter IP
28	Engine Id
29	Engine Type
30	Source Tag
31	Destination Tag
32	Source/Destination Tag

Flow-Dscan

The `flow-dscan` program attempts to detect unusual network traffic like port scanning and host scanning. Before you run the program, you will have to create two files in the current working directory:

```
Solaris# touch dscan.suppress.dst dscan.suppress.src
```

`flow-dscan` can use these files to suppress warnings about hosts that should not be reported. For now, the files can remain empty, but they must be present nonetheless.

You can instruct `flow-dscan` to run and print its warning messages to standard error with the `-b` flag:

```
Solaris# cat ft-v01.2003-03-30.153000-0500 | flow-dscan -b
flow-dscan: load_suppress 0
```

```
flow-dscan: load_suppress 1
flow-dscan: host scan: ip=10.124.2.66 ts=1049056190 start=0330...
flow-dscan: host scan: ip=10.3.21.106 ts=1049056204 start=0330...
flow-dscan: port scan: src=10.97.0.214 dst=10.8.15.1 ts=104906...
flow-dscan: host scan: ip=10.116.90.206 ts=1049056209 start=03...
flow-dscan: host scan: ip=10.57.101.250 ts=1049056215 start=03...
flow-dscan: host scan: ip=10.49.139.154 ts=1049056223 start=03...
flow-dscan: host scan: ip=10.186.240.211 ts=1049056226 start=0...
flow-dscan: host scan: ip=10.252.53.145 ts=1049056229 start=03...
```

If the `-b` flag is not used, `flow-dscan` will instead run in the background and send alert messages to syslog.

5.6.3. Manipulating Flow Data

The various methods for manipulating flow data are discussed in the following sections.

Flow-Cat and Flow-Merge

As seen earlier, the default behavior of `flow-capture` is to store 15 minutes' worth of flows in a single data file. What if you need to work with more than just 15 minutes? Perhaps you would like to produce a report on traffic from an entire day. Or maybe you wish to look for certain kinds of traffic over the entire month.

The `flow-cat` and `flow-merge` programs can be used in these circumstances. Both programs will combine flow data from separate files into a single set of flow data. This data can then be written to a single large data file or used directly in a pipeline. The difference between the two programs is that `flow-merge` will sort the data so that the new file contains all flows in their original chronological order. In contrast, `flow-cat` simply appends the data in the order the files are listed on the command line.

Here is a simple example using flow-merge to view data from multiple files:

```
Solaris# flow-merge ft-v01.2003-03-30.140900-0500 \
   ft-v01.2003-03-30.140937-0500 | flow-print
srcIP         dstIP         prot  srcPort  dstPort  octets  packets
10.124.25.66  10.53.9.94     17    3283     3283     33      1
10.124.25.66  10.53.9.96     17    3283     3283     33      1
10.53.1.135   10.0.0.251     17    5353     5353     116     1
10.124.25.66  10.53.12.101  17    3283     3283     33      1
10.124.25.66  10.53.12.102  17    3283     3283     33      1
```

You can also specify a directory name instead of a list of files, and the programs will include all flow data files beneath that directory. The following will concatenate all flows from March 29 and store them in a new file:

```
Solaris# flow-cat -o mar-29-flows \
   /var/tmp/flows/2003/2003-03/2003-03-29
```

Note that both programs are resource intensive and can take a long time to run when processing large amounts of data. `flow-merge` typically takes longer to complete than `flow-cat`, so if you do not need the flows merged by time, use `flow-cat` instead. Most of the time, the files you combine will be in chronological sequence anyway. Additionally, if you give the `-g` option to either program, it will ensure that the files on the command line are sorted by capture time before processing.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003

Flow-Split

The `flow-split` program performs exactly the opposite operation of `flow-merge` and `flow-cat`: It takes flow data and splits it into separate smaller files, based on criteria you specify on the command line. For example, the `-T` flag instructs it to split flows after a given number of seconds:

```
Solaris# cat ft-v01.2003-03-29.231500-0500 | flow-split -T 300
```

Here the original data file happened to store 15 minutes' worth of flows. `flow-split` will create three files, each storing 5 minutes (300 seconds) of flow data. By default, the files will be named `split.0`, `split.1`, and so on. You can change the base name for the files with the `-o` option.

Flow-Expire

`flow-expire` is a utility that will expire old flow data exactly as `flow-capture` does. It is useful in circumstances where you wish to store NetFlow data on a separate machine from where it is collected. The following will remove old data files in order to keep the total data size at less than 10 gigabytes:

```
Solaris# flow-expire -E 10G -w /var/tmp/flows
```

Flow-Header

The `flow-header` program prints meta information that `flow-capture` stores in the header portion of a Flow-Tools data file. For example:

```
Solaris# cat ft-v01.2003-03-30.160000-0500 | flow-header
#
# mode:                 normal
# capture hostname:     server.example.com
# capture start:        Sun Mar 30 16:00:00 2003
# capture end:          Sun Mar 30 16:15:00 2003
# capture period:       900 seconds
# compress:             on
# byte order:           big
# stream version:       3
# export version:       1
# lost flows:           0
# corrupt packets:      0
# sequencer resets:     0
# capture flows:        422712
#
```

Be aware that some programs, like `flow-cat` and `flow-merge`, require an additional argument in order to preserve header data between the input data and the output data.

Flow-Fanout

Cisco routers can send exported NetFlow data to only a single host and port, which can be a serious limitation. In our environment, for example, there are two groups that need access to flow data: the Network Operations group and the Network Security group. Say that each group wants to use a different tool to capture flow data because different tools suit their different needs. The only practical solution is to send two feeds of NetFlow data, one to each group. Since the router cannot perform this task, `flow-fanout` will do it instead.

The `flow-fanout` program listens on a port, receives flow data packets, and then sends that data as is to as many other host/port pairs as you wish to configure. This is the only program in the Flow-Tools package that does not need to know anything about the flows themselves; it simply receives UDP packets and forwards them to other hosts.

The following will listen for NetFlow data on port 9995 and then send it on to two other hosts:

```
Solaris# ./flow-fanout 0/0/9995 0/192.0.2.3/9991 0/192.0.2.4/9992
```

Note that one piece of information is lost when flows are forwarded in this way. The NetFlow data packets do not contain a field indicating which router generated the flow. Typically, this can be determined from the source IP address of the exported UDP packets, but not if a flow is forwarded from one host to another. In the above example, the machines 192.0.2.3 and 192.0.2.4 will see flows coming from the IP address of the server that is forwarding them.

The `flow-fanout` program does have a trick for preserving the originating router information. If you use the `-s` flag on the command line, it will forge the source addresses of the outgoing UDP packets for each flow to be that of the original address that sent the flow. For example:

```
Solaris# ./flow-fanout -s 0/0/9995 0/192.0.2.3/9991 \
   0/192.0.2.4/9992
```

Other Flow-Tools Programs

There are a few other useful tools in the Flow-Tools package. For example, the `flow-send` program will read flows from a data file and send them back in NetFlow format. The `flow-filter` command (and in later releases, `flow-nfilter`) allows you to filter flows based on ports, IP addresses, and more.

5.7. References and Further Study

In addition to the man pages and Web pages for each tool in the distribution, Flow-Tools includes pages called flow-tools and flow-tools-examples that give an overview of all of the tools and examples of specific use for each one.

The Cisco Web site at http://www.cisco.com/ has detailed information on the NetFlow protocol and configuration examples on both routers and Catalyst 5000 series switches. Because NetFlow is a Cisco product, Cisco’s Web site will have the most definitive information about the latest releases and protocol specifications.

Open Source Network Administration By James Kretchmar
ISBN: 0-13-046210-1 Publisher: Prentice Hall Print Publication Date: 9/22/2003