

# Netflow For Incident Detection<sup>1</sup>

*Michael Scheck / Cisco CSIRT*

*mscheck@cisco.com*

## **Introduction**

Netflow is often deployed for network billing, auditing, and accounting. However, Netflow can also be for incident detection as well as network forensics. This white paper will cover best practice guidelines for collection, analysis and detection.

## **Netflow Basics**

Netflow was originally developed to help network administrators gain a better understanding of what their network traffic looked like. Once Netflow is enabled on a router, it simply keeps track of any IP sessions, without storing any of the actual data used in that session. Because the contents of the session are not stored in Netflow logs, security professionals sometimes considered this a limiting factor in its use. However, that very behavior allows security administrators to install and use Netflow in places where a network sniffer could not handle the bandwidth, storage requirements, or CPU overhead.

Because of the many different uses for Netflow, there are several protocol versions and applications available for analysis. For the purpose of this guide, the tools discussed will be two of the most common, non-commercial tools. These tools are flow-tools, developed by the Ohio State University (now updated through the Google code project), and nfdump, developed by Peter Haag.

When deciding which Netflow tool to deploy for collection and analysis, keep in mind that there are several advantages and disadvantages for each tool. flow-tools is popular because it was one of the first freely available tools to

---

<sup>1</sup> This paper was chosen as a winner in the Best Practices Contest 2009, which was sponsored by the CERT® Coordination Center and the Forum of Incident Response and Security Teams (FIRST) in conjunction with the 2009 FIRST annual conference. CERT and FIRST make no warranties about the content of the paper.

support Netflow, and it has a rich feature base that includes analysis, statistical reports, and an easy to use method to run queries against standard named ACL files.

A limiting factor for the flow-tools package is the current lack of support for the Netflow v9 protocol. Netflow v5 is the most popular Netflow format, but it is not compatible with IPv6. Because of this, Netflow v5 is slowly being replaced by Netflow v9, which supports IPv6. In addition, the IETF is working on a new version of Netflow called IPFIX (sometimes referred to as Netflow v10).

Nfdump supports both v5 and v9. Nfdump also has a great deal of features that make it extremely popular. The syntax for nfdump pattern matching is very similar to tcpdump. This similarity makes it very easy for experienced engineers to adopt. Nfdump includes its own reporting features, as well as the ability to read any files written with the flowtools package. Because of this, it can be useful to install both flow-tools and nfdump, leaving the option to use whichever tool best suits a situation.

## **Collection**

There are several things to consider when deciding what your Netflow collection architecture should look like. These factors include collector disk space, CPU power, exported Netflow packets per second, and network topology. If your environment is simple enough, a single collector may be sufficient (Figure 1).

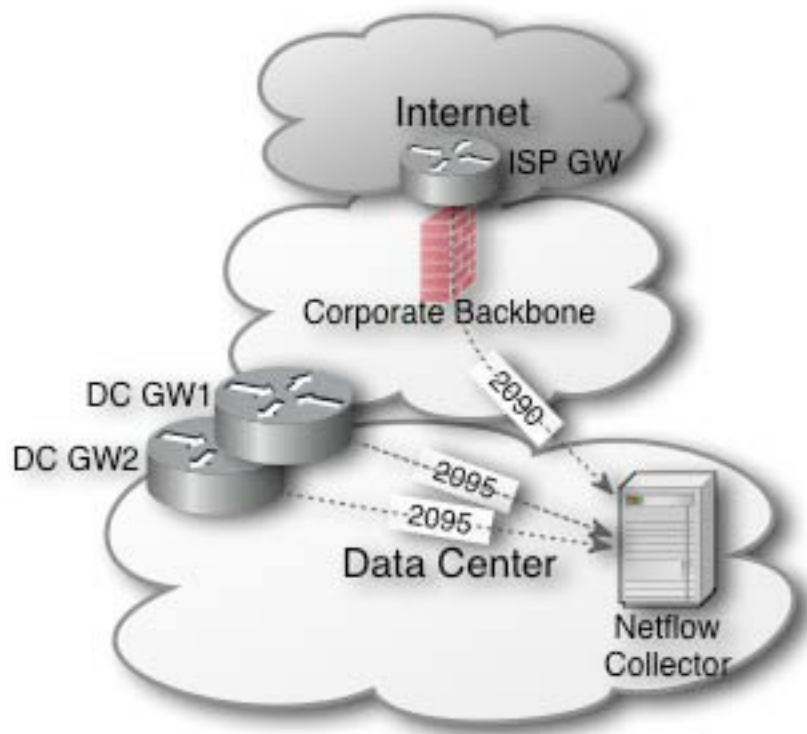


Figure 1

Disk space and I/O speed is always one of the primary concerns when preparing to deploy Netflow. Because of the nature of Netflow, it is difficult to predict what a specific network will require for Netflow storage. A general rule of thumb is 1MB of storage for every 2Gb of network traffic. This may not seem like a lot, but on an enterprise network this can add up quickly, especially given the fact that a single network transaction could potentially pass several devices exporting Netflow.

There are several open source and commercial tools that will take Netflow feeds and write the data directly to a database. Keep in mind that because of built-in compression done by the flow collector daemons, this will require considerably more space to store the same amount of data. In addition, it can also be easier to present a raw Netflow log as evidence when needed.

Because of these issues, if a database is used to store flows, it is often useful to have tables with limited capacity to store a small amount of flows, and keep the original Netflow logs as an archive. This solution will also allow you to

use any of the tools already written to analyze Netflow log files at your discretion.

An easy way to refine estimates on storage requirements for Netflow collection is to set up a temporary collector and begin exporting flows. Based on the size of the Netflow logs and the number of logs per day (this is a collector-specific setting that we will cover in the examples), it is easy to get an estimate for daily storage requirements.

Network topology also needs to be considered when choosing where to physically deploy your Netflow collector. Sending Netflow records from a remote WAN device to a central collector can put additional strain on a link unnecessarily, especially during a denial-of-service attack. Netflow packets are UDP based, so using high bandwidth connections can lead to exported flows being lost.

Although the collection of Netflow itself requires little CPU, many of the tools available to analyze Netflow data files will consume large amounts of processor power. Using existing collection infrastructure to also run queries on Netflow data takes advantage of additional CPU as well as disk I/O.

If there are multiple routers exporting flows to a single collector, it can be beneficial to have multiple Netflow collection daemons running on different ports writing to different directory structures. Segmenting the data in this way will allow queries to be run that are specific to the network segment associated with a specific router.

Many devices also have a limited number of Netflow destinations available for export. This factor, along with the fact that many products require their own Netflow stream, means that flows must often be replicated. This will also allow select traffic to be redistributed to additional applications by “fanning out” the flows (Figure 2).

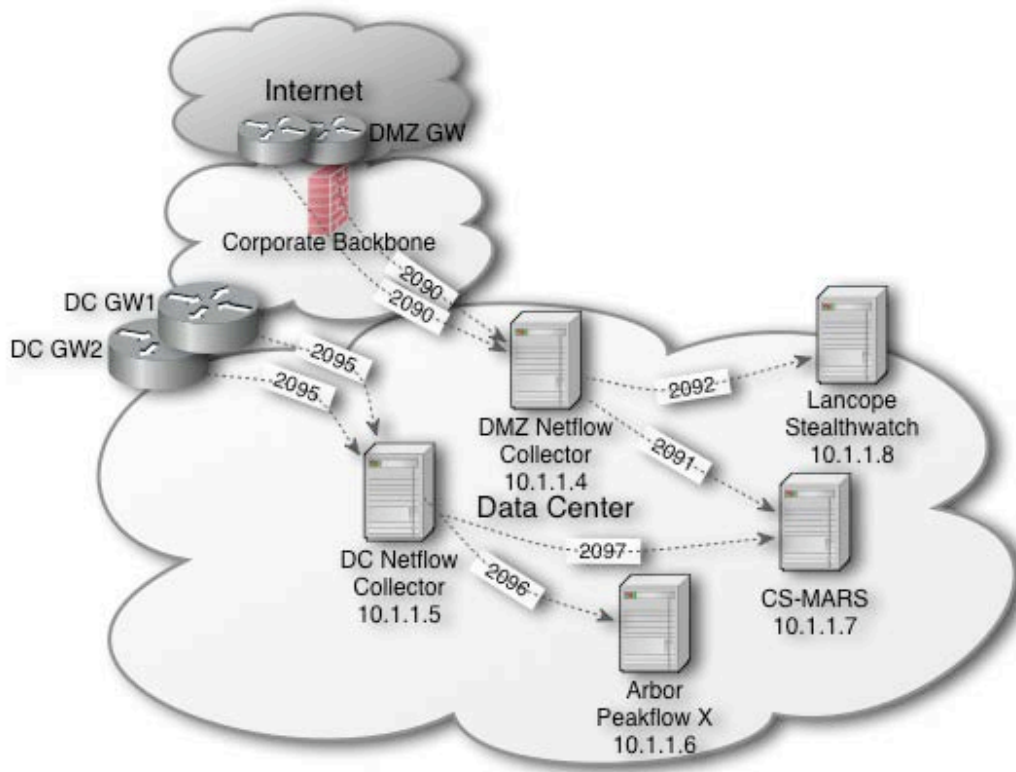


Figure 2

The CPU required for the collection daemon is fairly modest. The collector used in these examples is a 3.6 Ghz Linux machine collecting about 20GB of Netflow logs a day.

```
[mscheck@nfc-10 ~]$ top
```

```
Tasks: 136 total, 2 running, 134 sleeping, 0 stopped, 0 zombie
```

```
Cpu(s): 3.4% us, 0.9% sy, 0.0% ni, 94.9% id, 0.8% wa, 0.0% hi, 0.0% si
```

```
Mem: 4086496k total, 3897484k used, 189012k free, 55716k buffers
```

```
Swap: 4192924k total, 1696k used, 4191228k free, 3536688k cached
```

```
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
16459 netflow 15 0 4568 2304 644 S 5.9 0.1 8630:27 flow-capture
16478 netflow 15 0 3480 1884 560 R 2.0 0.0 1872:17 flow-fanout
```

The CPU consumption is about 6% for flow-capture, and the flow-fanout process distributing a copy of this stream to additional applications consuming Netflow is using an additional 2%.

## Collection Examples

In this section, we give specific examples that can be used to configure Netflow collection with flow-tools and nfdump, as well as to redistribute Netflow to additional collectors or applications.

To start the flow-tools capture process on a Linux machine, determine which filesystem you will be writing data to, and run the following command:

```
[mscheck@nfc-10 ~]$ flow-capture -w /var/local/flows/data -V5  
0/0/2055 -N1 -n288
```

- w Location on the Filesystem to write the data files

- V version

- 0/0/2060 localip, remoteip, port

- N nesting levels (1 directory per day)

- n rotations 5 minute intervals, 24 hours

- p PID file location

- E the maximum size of the log files before they are expired

On the same Linux machine, we will start the nfdump capture process listening on port 2060:

```
[mscheck@nfc-10 ~]$ nfcapd -w -D -l /var/local/flows/dcgw1 -p 2060
```

- p port

- w sync file rotation

- D fork to background

- l log directory

Both of the capture processes above will listen on the specified port and write any Netflow packets received to the configured directory. If additional copies of the Netflow logs are needed for analysis with other tools, this can be

accomplished by “fanning out” the Netflow data. To send a copy of Netflow with flow-tools from the collector to a remote device:

```
[mscheck@nfc-10 ~]$ flow-fanout 0/0/2060  
192.168.185.68/10.81.252.139/2060  
192.168.185.68/10.102.12.45/2055
```

Format is LOCALIP/REMOTEIP/PORT.

First IP address is “in”, all additional IP addresses are “out”.

Nfdump also has a method for forwarding flows as they are received, as well as to “replay” a file back out to the network. This allows specific files that match a specific requirement to be forwarded to an additional tool for analysis. It is not recommended to use nfreplay to forward all packets, because every log must be written and then reread. This is what the nfcapd -R option is designed for. To forward all incoming flows with nfcapd:

```
[mscheck@nfc-10 ~]$ nfcapd -w -D -l /var/local/flows/dcgw1 -p 2060  
-R 10.102.56.76/2060
```

-R Destination host/port to receive flows

There will be times when sending a copy of Netflow records from a file to another collector will be useful, especially when testing products that specialize in anomaly detection. Netflow records that may have been collected during past incidents can provide insight into how the application may respond during similar incidents in the future. Previous flows can also be used to train these devices so that less time is needed to learn an environment. To read a file with nfdump and send the data to a remote collector, you can nfreplay:

```
[mscheck@nfc-10 ~]$ nfreplay -r ft-v05.2009-04-27.100255-0400-H  
10.102.56.76 -p 2060
```

-r file to read (without it uses STDIN)

-H host to send replay

-p port on remote host

## Exporting Netflow

Netflow can be exported from a variety of devices, including enterprise and small office routers, dedicated appliances, as well as passive monitors that

work in the same manner as a network sniffer. Netflow should be exported from devices located at points of convergence, much like an IDS sensor. Having well-placed Netflow exporters reduces the number of devices needed to cover the entire network and reduces storage requirements.

It is highly recommended to use NTP and a common time zone for time synchronization. Setting the time zone of all your network devices to a common format such as UTC takes very little effort and prevents the need to reconcile date and time issues during an incident.

Flow sampling is often used for network utilization and trending. Sampling works by selecting a subset of packets that pass through an interface. Sampling is configured as a ratio. If sampling is set to 100, the ratio is 1 to 100. Because not all flows are collected, sampling should not be used when setting up Netflow for incident detection.

Netflow records are “timed out” from an exporting device based on the state of the flow. Flows are classified as “active” or “inactive.” A flow becomes inactive when an IP connection is terminated. The inactive timeout setting determines how long (in seconds) a flow is held in cache after it becomes inactive.

The active timeout setting determines when a currently active flow is removed from cache and exported to a collector. Once a flow is exported from cache, a new flow is created as soon as a packet from the session crosses the exporting device. Because of this behavior, a single IP session can have multiple flows. The nfdump tool has an aggregation feature that reassembles Netflow records split because of active timeouts.

The processing overhead from enabling Netflow is minimal. On a router outputting 20GB of Netflow data a day and 800+ active flows, the Netflow process consumed less CPU than the SSH process during a 5-minute period.

Determining the CPU overhead of Netflow:

```
dmzbb-gw1>sh proc cpu | incl Flow
```

```
278    0    2    0 0.00% 0.00% 0.00% 0 IP Flow Backgrou
313    0    1    0 0.00% 0.00% 0.00% 0 NetFlow Agg Task
```

```
dmzbb-gw1>sh proc cpu | incl SSH
```

```
45    168   240   700 0.31% 0.14% 0.05% 1 SSH Process
141   3296 969001    3 0.00% 0.00% 0.00% 0 SSH Event handle
```

## Export Examples

Exporting Netflow from a router is actually very simple. It basically consists of telling the router which version of Netflow to export and where to send it. To export Netflow from a Cisco router at the config, use the following commands:

```
Router(config)#ip flow-export version 5
Router(config)#ip flow-export source FastEthernet 2/48
Router(config)#ip flow-export destination 10.1.1.1 2060
Router(config)#interface FastEthernet2/48
Router(config-if)#ip route-cache flow
```

Juniper JunOS routers also supports Netflow. Below are the configuration lines needed to start exporting Netflow:

```
sampling {
  input {
    family inet {
      rate 100;
      run-length 9;
      max-packets-per-second 7000;
    }
  }
  output {
    cflowd 10.1.1.1 {
      port 2060;
      source-address 10.0.1.1;
      version 5;
      no-local-dump;
      autonomous-system-type origin;
    }
  }
}
```

```
}  
}  
}
```

In addition to exporting Netflow from network infrastructure, a Linux machine can use its network interface in promiscuous mode to capture traffic and convert the stream to Netflow. This will likely require a monitor mode (span) session on the switch so that the Linux machine will see all traffic. It is obviously limited to the interface speed of the Linux machine. To Export Netflow from a Linux machine with fprobe:

```
[root@nfc-10 ~]# fprobe-ulong -B4096 -r2 -q10000 10.0.1.1:2060
```

- B Increase kernel capture buffer size is most adequate way to prevent packets loss

- r Real-time priority

- q Pending queue length

## Analyzing Netflow

Netflow can be used to determine when a specific network transaction has occurred, such as a connection to a malicious destination, or traffic that does not meet a certain criteria. It can include IP services that were not authorized, as well as large file transfers and connections to botnet command and control servers.

Having multiple collectors allows global queries to be run directly on each collector in parallel (Figure 3). This option will take advantage of the additional CPU and disk I/O of each collector, and it will increase the speed of the query. In addition, if the traffic only passed through a single collector (such as a single internal host establishing an outbound connection) only that collector has to be queried, reducing the amount of Netflow logs that need to be processed.

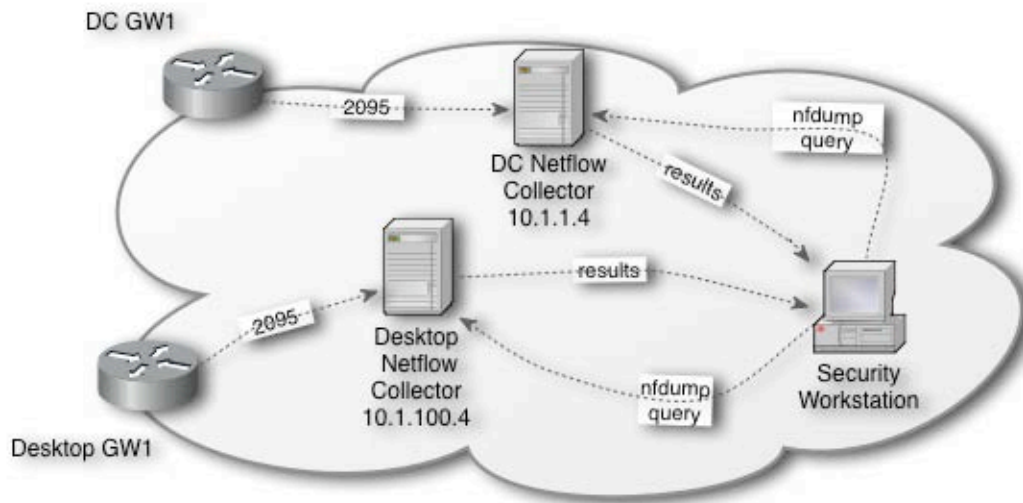


Figure 3

Botnet detection is an ideal task for Netflow. Once a compromised host is detected on the network, it is simple to determine all IP connections that the compromised host made. It is often fairly easy to determine which of these connections are legitimate and which may belong to a command and control server.

Although the tools discussed here are flow-tools and nfdump, nearly any Netflow tool that can be queried based on the fields contained in the Netflow headers can be used to detect the same incidents. The fields most often used for incident response are time, source IP/port, Destination IP/port, number of packets, and bytes.

### Analysis Examples

During an investigation, something as simple as determining which external hosts and ports an IP address connected to during a specific time period may prove to be a valuable piece of evidence.

Once a compromised host has been identified, a list of destination hosts and ports is critical to identify new botnet command and control servers. Even if the investigator has access to the compromised host, it is possible that the system will contain a rootkit that will hide network connections. Netflow is a reliable way to examine the network connections. Because of the nature of botnets, the compromised host will nearly always have a current connection open to the command and control server.

Having multiple suspected hosts will also decrease the time required to determine the command and control server. A simple correlation of destination hosts/ports in common will limit the destinations that need to be investigated to the common destinations.

In this example, we use flow-tools to determine all of a host's outbound connections using a named ACL called "infectedhost" as the source (using the -S option):

A. [mscheck@nfc-10 ~]\$cat ip access-list standard infectedhost permit ip 192.168.100.1 >flow.acl

B. [mscheck@nfc-10 ~]\$flow-cat /var/local/flows/data/2008-12-01/ft\* | flow-filter -f ~/flow.acl -Sinfectedhost | flow-print -f5

<i>Start</i>	<i>End</i>	<i>Sif</i>	<i>SrcIPaddress</i>	<i>SrcP</i>	<i>Dif</i>	<i>DstIPaddress</i>	<i>DstP</i>	<i>P</i>	<i>Fl</i>	<i>Pkts</i>	<i>Octets</i>
1130.23:54:57.489	1130.23:54:57.489	97	10.31.33.7	8080	58	10.102.121.149					
5897	6	0	1	78							
1130.23:55:00.497	1130.23:55:00.689	98	10.25.5.15	4466	58	10.31.33.7	8080				
6	0	2	148								

We can run the same query with nfdump using simple tcpdump style expressions. Here we simply say any traffic that matches source IP 192.168.100.1:

[mscheck@nfc-10 ~]\$nfdump -r /var/local/flows/data/2006-12-01/ft\* 'src ip 192.168.100.1'

<i>Date flow start</i>	<i>Duration</i>	<i>Proto</i>	<i>Src IP Addr:Port</i>	<i>Dst IP Addr:Port</i>	<i>Packets</i>	<i>Bytes</i>	<i>Flows</i>
2009-04-23 12:15:15.267	23.258	TCP	192.168.100.1:50034	-> 10.125.93.100:80	3	1291	1

As mentioned previously, there may be times when a flow is collected and stored in the flow-tools format, and it is desirable to analyze that data with features only available in the nfdump suite. This situation can be addressed

by using the “ft2nfdump” executable distributed with the nfdump source code. Here is an example of this hybrid approach to determine a host’s outbound connections with files created with flow-tools and converted to nfdump:

```
[mscheck@nfc-10 ~]$ flow-cat ft-v05.2009-04-23.15* | ft2nfdump |nfdump 'srcip 10.102.194.55'
```

<i>Date</i>	<i>flow start</i>	<i>Duration</i>	<i>Proto</i>	<i>Src IP Addr:Port</i>	<i>Dst IP Addr:Port</i>	<i>Packets</i>
<i>Bytes</i>		<i>Flows</i>				
2009-04-23	15:10:24.831	65.536	TCP	10.102.194.55:50075	-> 10.125.93.100:80	
7	1493	1				
2009-04-23	15:12:40.539	0.000	UDP	10.102.194.55:123	-> 10.151.16.23:123	
1	76	1				

This syntax can be used with any of the features available with flow-tools, including flow-filter. It allows the use of ACL files to search through Netflow data, which can be useful for firewall audits.

```
[mscheck@nfc-10 ~]$/ flow-cat ft-v05.2009-04-23.15* | flow-filter -f ~/flow.acl -Sinfectdhost |ft2nfdump |nfdump -a
```

<i>Date</i>	<i>flow start</i>	<i>Duration</i>	<i>Proto</i>	<i>Src IP Addr:Port</i>	<i>Dst IP Addr:Port</i>	<i>Packets</i>
<i>Bytes</i>		<i>Flow</i>				
2009-04-23	15:31:28.426	0.000	TCP	10.102.194.55:50218	-> 10.55.149.121:80	
1	46	1				
2009-04-23	15:31:29.037	0.000	TCP	10.102.194.55:50226	-> 10.55.149.118:80	
1	46	1				

A collector is not always needed for basic analysis. Before a flow is actually expired and exported from a router, you can also log directly into the router examine current flows. This will work as long as Netflow is enabled, even if an export destination is not configured.

```
dmzbb-gw1>sh ip cache flow | incl 10.102.194.55
```

Gi4/10	10.102.194.55	Local	10.81.255.15	06 C4B9 0016	164
Gi4/10	10.102.194.55	10.81.255.15	tcp	50361	22 0
Gi4/10	10.81.255.15	10.102.194.55	tcp	22	50361 133

Once a command and control host is identified, the next step is to determine which hosts are compromised. Again, because of the nature of botnets, it is fairly simple to run a Netflow query to determine all hosts connecting to the command and control channel. Most of the botnets today use a password

protected IRC server/channel to issue and receive commands from bots, so there is rarely any valid reason to connect to the destination host/port.

An ACL file can be used with flow-tools to query any connections to the command and control server. This could be a regularly scheduled job that queries an ACL file with all known command and control servers. The syntax is similar to the previous ACL example, except `-D` is used to use the ACL as a destination:

```
A. [mscheck@nfc-10 ~]$cat ip access-list standard botnet permit ip
10.31.33.7 >flow.acl
```

```
B. [mscheck@nfc-10 ~]$flow-cat /var/local/flows/data/2006-12-
01/ft* | flow-filter -f ~/flow.acl -Dbotnet | flow-print -f5
```

<i>Start</i>	<i>End</i>	<i>Sif</i>	<i>SrcIPaddress</i>	<i>SrcP</i>	<i>Dif</i>	<i>DstIPaddress</i>	<i>DstP</i>	<i>P</i>	<i>Fl</i>	<i>Pkts</i>
<i>Octets</i>										
1130.23:54:57.489	1130.23:54:57.489	97	10.31.33.7	8080	58	10.102.121.149				
5897	6 0 1	78								
1130.23:55:00.497	1130.23:55:00.689	98	10.25.5.15	4466	58	10.31.33.7	8080			
6 0 2	148									

The query is also very simple with `nfdump`. In this example, we once again take advantage of `nfdump`'s `tcpdump`-like syntax:

```
[mscheck@nfc-10 ~]$ nfdump 'dst ip 10.102.42.161'
```

<i>Date flow start</i>	<i>Duration</i>	<i>Proto</i>	<i>Src IP Addr:Port</i>	<i>Dst IP Addr:Port</i>	<i>Packets</i>
<i>Bytes</i>	<i>Flows</i>				
2009-04-27 10:07:47.180	0.064	TCP	10.205.56.145: 50213	->	
10.102.42.161:31337	4	1057	1		
2009-04-27 10:07:47.373	0.064	TCP	10.205.56.171: 52215	->	
10.102.42.161:31337	4	453	1		
2009-04-27 10:07:46.711	0.448	TCP	10.113.146:52207	-> 10.102.42.161:31337	
45	58558	1			

In addition to using Netflow to determine connection to any suspected or known malicious hosts, it can also be useful to detect incidents by determining traffic that does not match expected traffic. For example, if a web server was deployed, you could use Netflow to look for any traffic that does not appear related to web services.

In this example, we use `nfdump` to only show connections that are not expected traffic from a web farm subnet. Any flows that are not port 80 or 443 should immediately be investigated. This could also be reversed to see if

there are any outbound connections from the web farm subnet where the source is not port 80 or 443.

```
[mscheck@nfc-10 ~]$ flow-cat ft-v05.2009-04-23.15* | ft2nfdump |  
nfdump -a 'not dst port 80 and not dst port 443 and dst net  
10.11.85.0/24 '
```

```
2009-04-23 15:00:12.656 3.008 TCP 10.1.1.235:19020 -> 10.11.85.196:23 2  
96 1
```

The source and destination are not always the most useful criteria for discovering an incident. There may be cases in which the amount of data transferred should be closely watched. This is often the case for monitoring intellectual property loss. The transfer of large files from a datacenter or code development environment is often worth investigating.

In this example, nfdump is used to read files written by flow-tools, and any file transfers greater than 25MB are reported. In addition, ports between 5900 and 6000 are ignored as part of this query. NFDUMP is also using the “-a” command to do flow aggregation and reassemble any flows exported from the router before they were finished. This will detect long-running transfers that would otherwise fall under the file size threshold.

```
[mscheck@nfc-10 ~]$ /usr/local/netflow/bin/ft2nfdump -r ./ft-  
v05.2009-04-26.235502-0400|/usr/local/netflow/bin/nfdump -a  
'bytes > 25000000 and not ( srcport < 6000 and srcport > 5900)'
```

<i>Date flow start</i>	<i>Duration</i>	<i>Proto</i>	<i>Src IP Addr:Port</i>	<i>Dst IP Addr:Port</i>	<i>Packets</i>
<i>Bytes</i>	<i>Flows</i>				
2009-04-26 23:54:36.059	305.536	UDP	10.102.253.73:4500	->	
10.79.125.121:8284	44809	58.2 M	1		
2009-04-26 23:54:47.065	179.904	TCP	10.102.242.195:80	->	
10.209.56.10:38727	45886	62.1 M	1		
2009-04-26 23:58:11.548	29.696	TCP	10.67.91.49:80	->	
10.44.248.127:46955	25541	34.6 M	1		
2009-04-26 23:56:05.596	208.064	TCP	10.210.27.33:30936	->	
10.135.250.12:443	41808	32.2 M	1		

In addition to running Netflow queries to look for specific behavior, each Netflow tool suite has several tools to run reports based on utilization. This can be useful for targeted monitoring of specific network segments, as well as detecting network misuse, such as DOS attacks or portscans.

In this example, we will use nfdump’s “-s” flag to generate statistics.

```
[mscheck@nfc-10 ~]$nfdump -r ft-v05.2009-04-27.100753-0400 -o
extended -s srcip -s ip/flows -s dstport/pps/packets/bytes -s
record/bytes
```

*Aggregated flows 21428*

*Top 10 flows ordered by bytes:*

```
Date flow start      Duration Proto  Src IP Addr:Port  Dst IP Addr:Port  Flags
Tos Packets  Bytes  pps  bps  Bpp Flows
2009-04-27 10:05:47.287 135.424 TCP   10.102.242.195:80 -> 10.52.195.58:6269
..... 0 67899 97.1 M  501  5.7 M 1499  1
2009-04-27 10:07:03.190  54.400 TCP   10.166.218.43:38282 ->
10.135.250.12:65041 ..... 0 43058 32.1 M  791  4.7 M 782  1
2009-04-27 10:07:49.230  6.400 TCP   10.135.250.12:22 -> 10.70.145.36:755 .....
0 10972 14.8 M 1714 18.5 M 1413  1
```

In this example, the flow-tools' "flow-stat" utility is used to generate a report based on the source IP address, sorted by the # of octets transferred.

```
[mscheck@nfc-10 ~]$/usr/local/netflow/bin/flow-cat ft-v05.2009-04-
27.100753-0400 | /usr/local/netflow/bin/flow-stat -f9 -S2 |head -20
```

```
# IPaddr  flows      octets      packets
10.102.242.195 1          101810625    67899
10.166.218.43  2          33703108     43079
10.135.250.12  4          15520392     11011
```

The flow-top-talkers command can be configured on a Cisco router to display the top connections. This only shows flows that have not been expired from cache. This feature can be used as a quick investigative method during a DOS attack.

```
Router(config)#ip flow-top-talkers
```

```
Router(config-flow-top-talkers)#top 10
```

*The following is the 10 ten talkers in network sorted by packets:*

```
R3#show ip flow top-talkers
SrcIf SrcIPaddress DstIf DstIPaddress Pr SrcP DstP Pkts
Et1/0 10.1.10.2 Et0/0 10.16.100.5 06 0087 0087 3483
Et1/0 10.1.10.2 Et0/0 10.16.100.8 06 0089 0089 837
Et1/0 10.1.10.2 Et0/0 10.16.100.6 06 0185 0185 372
```

*Et1/0 10.1.10.2 Et0/0 10.16.100.7 06 00B3 00B3 200*

## **Conclusion**

We have covered several uses for Netflow, but these examples are really just the tip of the iceberg. As long as there is a need to understand what transactions take place on the network, security professionals will have a use for Netflow.

## **References**

Nfdump homepage:

<http://nfdump.sourceforge.net/>

FlowTools development page:

<http://code.google.com/p/flow-tools/>

Cisco netflow documentation:

[http://www.cisco.com/en/US/products/ps6601/products\\_ios\\_protocol\\_group\\_home.html](http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html)

Juniper cflowd documentation:

<http://www.juniper.net/techpubs/software/junos/junos94/swconfig-policy/cflowd.html#id-11435795>