Security Applications for Cisco NetFlow Data

Jana Dunn
July 23, 2001

Introduction

Good network security requires good network monitoring. Network monitoring provides baseline information about normal network behavior and can alert staff to potential problems. During or after a security incident, the data collected with network monitoring tools can assist network managers in determining what has happened, what remediation needs to be done, and how to prevent future occurrences.

Network Traffic Monitoring Tools

Traditional SNMP-based traffic monitoring tools such as **mrtg** and **cricket** can provide some warning that not all is well--a unexpected increase in traffic may indicate a security incident in progress--but SNMP-based tools only provide information about levels and changes in traffic volume. For security purposes, this provides insufficient detail. Sniffers and related tools can provide far greater detail, but storing and analyzing the generally large volume of data is not practical in all environments, and ever increasing bandwidth makes this kind of data capture and analysis increasingly problematic. In addition, technologies such as ATM can hinder access to network information. Most network managers need tools that provide them with something more along the lines of a happy medium with regards to data quantity and level of detail. Flow profile data and flow analysis has the potential to partially fill this niche, providing the network manager with a set of useful utilities to add to his security toolbox.

Flow Profiling and Cisco NetFlow Services

Networking researchers define a network flow as a unidirectional sequence of packets between two network endpoints--a source and a destination. Researchers and networking equipment vendors have developed tools and techniques for flow profiling in order to better understand the nature of Internet traffic. Flow data provides a relatively detailed source of data about network traffic. Cisco's NetFlow services, the most commonly-available flow profiling system, provides the measurement base for most flow-based network analysis. Although Cisco initially developed NetFlow, other vendors such as Juniper and Extreme Networks have released similar implementations.

The NetFlow records exported by routers consists of call-record-like flow information; the content varies slightly between NetFlow versions. A NetFlow-enabled router creates a NetFlow record when the router first sees traffic new between two endpoints. Note that the router only creates flow records for *incoming* traffic. A NetFlow version 5 (V5) record contains the following:

- IP addresses of the endpoints
- Transport layer application port numbers for both endpoints
- IP protocol type

- Type of Service (ToS)
- Input and output interfaces of the reporting router
- Next hop router address
- Packet and byte counts for the flow
- Start and end of flow timestamps
- Source and destination autonomous system (AS) numbers
- Source and destination subnet masks

The router expires and exports flow records when one of the following conditions is met:

- Flows which have been idle for a specified time are expired.
- Long-lived flows are expired; by default this is set at thirty minutes.
- The cache becomes full, and so heuristics are applied to age groups of flows to expire and export those flows.
- The TCP connection associated with the flow has reached its end (FIN) or has been reset (RST).

The router groups the records of expired flows into NetFlow Export UDP datagrams for exportation to a collection station. Export datagrams generally contain multiple flow records. The router exports datagrams at least once per second, or when a full datagram becomes available. Flows are unidirectional, so a typical TCP connection might consist of a flow from the initiating host to a second host, and an answering flow from that second host back to the original host. Longer TCP connections may be represented by multiple flow records due to the expiration rules for the flow cache.

Strictly speaking, the term *flow* refers to an actual sequence of packets; a NetFlow record is the associated accounting information that corresponds to the flow of packets. However, users of NetFlow data tend to refer to the accounting record as a flow, as that record is the data they're actually dealing with.

Collection and Analysis Tools
Cisco provides commercial NetFlow collection and analysis tools: NetFlow Collector and NetFlow Analyzer. There are also a number of no-cost collection and analysis tools. As the majority of flow-related security work appears to use the latter, this paper focuses on the free tools. These tools consist of collectors, various analysis tools, and visualization tools.

Collection and Analysis with a Flow Focus
The Office of Information Technology Enterprise Networking Services group at Ohio State University (OSU) has developed an extensive suite of tools, **Flow Tools**, for collecting and analyzing NetFlow data. These tools are grouped roughly as follows:

- Capture tools
- General analysis tools

- Security tools

At OSU, the networking staff uses flow logs for the following tasks:

- Network planning
- Performance monitoring
- Usage billing
- Security incident response
- Intrusion detection

The **flow-capture** tool receives the NetFlow records exported by the routers. To accommodate the slight differences in NetFlow versions, the capture tool receives the records in native format, and then rewrites them into an internal format that contains the field set that corresponds to NetFlow V5. **Flow-capture** then writes the reformatted record to a log file. Periodically the capture tool rotates to a new file to control the size of the logs. **Flow-capture** can also provide a real-time feed of records.

The flow records, in their internal format, are kept in their entirety in the log files; all the V5 NetFlow data is preserved. The OSU staff keeps these flow records for several months to facilitate analysis. Note that **Flow Tools** does not store aggregated data; reporting tools aggregate data after the fact. Since **Flow Tools** generates a record for each flow, it produces a significant amount of data. OSU reports receiving three to four gigabytes per day for a single busy border router. With compression, this nets 90 MBytes data. The amount of data produced by any given router varies considerably over time. Denial-of-service attacks, particularly SYN floods, greatly increase the number of records produced.

Flow Tools Security Tools
NetFlow logs do not contain the content of the packets associated with the flow, and so are not useful for content-based intrusion detection. NetFlow records also lack the detailed packet header information useful for signature detection. However, flow logs can be used to detect policy violations, to report on the network activities of compromised hosts, and to detect some forms of scanning and denial-of-service attacks.

For investigating suspected incidents that are no longer in progress, the OSU staff members use **flow-scan-report** to select and print from archived flow records activity to and from a given host during a particular time frame. The staff then correlates this network activity with activity detected by other tools. For incidents still in progress, the staff uses **flow-dscan** to connect to **flow-capture** to detect and report network events in near real-time.

Correctly interpreting a "live" flow feed can be tricky. Due to the way flows are exported, flow record arrivals do not necessarily correspond to the order in which the flow traffic arrived at the router. Flow records are exported after the flow expires; for a long flow, this may be thirty minutes after the traffic associated with the flow began. This means that the collector can essentially receive the flow records "out of order"; a record associated with a short flow can arrive before the record associated with a long flow, even though some of the packets included in the long flow passed through the router before the packets that were included in the short flow. This artifact of the exportation process complicates interpretation of a near-real-time stream of flow records. The following quote is from a paper by Mark Fullmer and Steve Romig on the OSU **Flow Tools** suite:

> "For example, you may suspect that an intruder is logging into a host through a backdoor of some sort, and you can see network activity coming from the host (scans, exploit attempts, IRC connections, etc.) that lead you to believe that the intruder is active. But the flows associated with the backdoor connection may not show up in the flow logs until thirty minutes after the backdoor traffic actually started, so other activity may actually occur first in the log."[1]

Flow-dscan can also be run against archived records. To mitigate the flow order problem, **flow-sort** will pre-sort flows by the flow start times.

OSU uses the tools **flow-filter**, **flow-stat**, and **flow-print** for incident response. To investigate an alleged incident, the networking staff uses the flow logs for the following tasks:

Confirm that the incident happened on campus (i.e. is represented in the logs).
- If an OSU host was compromised, determine what hosts the compromised host contacted.
- Determine if the compromised host is/was being controlled from elsewhere.
- If the host used to compromise the OSU host can be determined, search for other OSU hosts the attacking host might have compromised.

Determining the client/server relationship between two hosts can be problematic. Flows are unidirectional and do not contain an indication of which host initiated the connection. Recall that a single TCP connection is represented by at least two flow records, one for each direction. Ordinarily one would use the TCP flags—the host that generated the first SYN packet for the TCP three-way handshake initiated the connection. Although flow records do contain a TCP flags field, the field contains a logical OR of the flags for the packets represented in the flow. During the course of the three-way handshake, the initiating host will send a SYN and an ACK and the responding host will also send a SYN and an ACK. With the bits logically OR'ed together, we can't use the TCP flags to determine the initiating host.

Analysts can use other heuristics to assist with determining the client/server relationship. One can look at the source and destination ports to make educated guesses as to which host was the server and which the client. In some situations, NetFlow records can be correlated with information from other sources, such as **syslog** or **tcp_wrapper** logs to determine which host initiated the connection. Alternatively, one can look at the starting timestamps of the flows to determine the order of the flows. The source host for the first flow initiated the connection. OSU uses **flow-connect** to attempt to sort and string together flows using the flow timestamps to facilitate incident analysis.

Asymmetric routing can also complicate this kind of flow-by-flow analysis. If outbound traffic passes through a different router than inbound traffic, both routers must be exporting flow records, the routers must be producing synchronized timestamps, and the records will need to be correlated for the analysis to make sense.

Collection and Analysis with an Aggregation Focus
The Cooperative Association for Internet Data Analysis (CAIDA) offers **Cflowd**, a well-developed set of NetFlow collection and analysis tools. The **Cflowd** developers designed this program suite primarily as a capacity planning tool; **Cflowd** focuses on how networks communicate with other networks. **Cflowd** offers several aggregation schemes and data views, but unlike Cisco's NetFlow Collector, **Cflowd** aggregation schemes do not provide any host level granularity; its most granular aggregation scheme is the network matrix.

The **Cflowd** package, like **Flow Tools**, consists of a suite of tools. Two programs, **cflowdmux** and **cflowd**, together comprise the collector or listener piece; these tools collect and tabulate flows records, writing them into an internal format. Another tool, **cdfcollect**, gathers the flow data from **cflowd** and aggregates the data, writing the aggregated data to disk in a specialized database format for later processing by other tools.

On the plus side, aggregation has the advantage of greatly reducing the amount of data collected. On the minus side for security uses, all the interesting flow-by-flow detail is no longer available for forensic study. From a security standpoint, the aggregated data provides primarily baseline and gross change information. Abrupt or unexpected changes in traffic behavior could signal the need for further investigation.

However, the **Cflowd** package can be configured to provide data for flow-by-flow analysis. If **cflowd** is configured to save flows, it will write per-flow data in an internal format to disk; these files are called "flow dumps". To keep these log files from growing too large, **cflowd** rotates the flow dumps. To manage space, **cflowd** rotates out and overwrites the old dump information. For "live feed" examination, **Cflowd** provides a tool named **flowwatch**; **flowdump** examines flow dumps on disk.

Visualization
Dave Plonka, at the University of Wisconsin-Madison, has written a patch for **cflowd** that enables **cflowd** to produce flow dump files with a timestamp in the file name to facilitate per-flow investigations. This patch does not affect other **Cflowd** processing and can allow for near-real-time visualization via **FlowScan**, also by Dave Plonka. **FlowScan** binds together the patched **cflowd** for flow collection, a database back end, **RRD**, and a visualization tool, **RRDtool**. **FlowScan** examines flow data from the time-stamped flow

dumps and maintains counters reflecting what it found. **FlowScan** stores the counter data in the **RRD** back end, and then builds graphs. **FlowScan** can provide graphs of input and output traffic by source network, by protocol, by well-known-service, and by autonomous system number. These tools provide a near-real-time view of network traffic with considerable detail. For sample graphs, see http://wwwstats.net.wisc.edu/.

Dave Plonka designed **FlowScan** to give a visual representation of the traffic passing through a network's border router or routers. If several routers make up the border (and all are exporting NetFlow data), **FlowScan** combines the data to produce a composite picture of the traffic entering the network and exiting the network. **FlowScan** is not specifically designed to be a per-router analysis tool or to monitor traffic that does not pass through the network border, but it can be configured to do so. The simplest way to monitor routers or groups of routers separately is to run a separate set of the **cflowd** listeners (**cflowdmux** and **cflowd**) and a corresponding copy of **FlowScan** for each router set one wants to monitor. For simplicity (and performance), each set of **cflowd-cflowdmux**-**FlowScan** should be put on a separate physical system. It is possible to configure more than one collection set on a single server. In my test of this setup, I have two collector sets running on a small Solaris system. While the two collectors appear to be co-existing reasonably well, I suspect that there are likely to be performance issues with scaling beyond a small number of collectors. The steps are as follows:

- For each set of routers to be monitored separately, select a NetFlow export port.
- Configure an instance of **cflowdmux** to listen on each of those export ports.
- Give each instance of **cflowd** a separate port on which to listen for the **cfdcollector**
- Use a different table socket name for each instance of **cflowd**
- Use DNS CNAMEs or equivalent to trick **cfdcollector** into thinking each instance of **cflowd** is running on a separate system
- Configure each copy of **cflowd** to put its flow dumps in a separate directory
- Configure a separate copy of **FlowScan** for each directory full of flow dumps

In addition to general network monitoring functions, **FlowScan** can also be used for more specifically security-oriented functions.

Detecting Denial-of-Service Floods

FlowScan can pick up traffic floods that might go unnoticed on bandwidth usage graphs. For example, suppose a flood of small packets is set to a group of campus addresses. The flood may be more visible on the **FlowScan** graph of flows than on a packet graph, as the spike produced by the flood is a much larger percentage of total flows than it is of total packets.

Network managers can apply other heuristics when looking for DoS attacks. The following quote is from a paper by Dave Plonka about **FlowScan**:

"[…] in our experience with **FlowScan**, we have learned that a discrepancy between the number of inbound and outbound flows or packets is an indication of abusive traffic, such as a DoS flood. Sudden changes in packet counts, especially when constrained to one protocol, are usually indications of a flood as well."[2]

Not all spikes indicate abusive traffic. **FlowScan** may report a quantity of traffic in a given five-minute period that's actually larger than the traffic passed by the router in that time frame. **FlowScan** collects data at five minute intervals. For simplicity, **FlowScan** assumes that whatever data counts are in the flow records it collects actually occurred in the given five-minute interval. As that set of flow records may include records of very long flows that represent large amounts of data, this may result in a data spike that could even be greater than the physical capacity of the link. While flow records do contain start and stop times, they do not contain any information about the distribution of delivery times of the packets in that flow. Incrementing counters as if all the flow packets arrived in the last five minutes of the flow is as accurate (or inaccurate) as any other assumption about the packet delivery. For a traffic mix that consistently contains long flows, it might be useful to adjust the timeout parameters of the exporting devices to minimize spikes.

Detecting Policy Violations

In some networking environments, the use of file sharing programs such as Napster or Gnutella signals a possibly compromised host. In other environments this might be a legitimate computer user violating policy. In either case, the network manager would like to be able to detect and monitor the traffic. **FlowScan** spots Napster usage through stateful inspection of flow records. In order to avoid mis-identifying data, **FlowScan** "remembers" what it has seen. The following comes from Dave Plonka's web page describing **FlowScan**'s analysis of Napster traffic:

"**FlowScan** watches for traffic from campus machines to the Napster.com servers. When it sees this, it remembers the identities (IP addresses) of the server in the outside world and the client on our campus, and also remembers the time at which the traffic was observed. We call the identified server a "NapServer" and the campus machine with which it interacted a "NapUser". Subsequently, when **FlowScan** sees traffic between a machine in the outside world and a NapUser it concludes, based on some rules about protocols ports and packet sizes, whether or not that machine is a remote NapUser, and therefore that this traffic represents the passing of data between Napster application users. Byte and packet counts for both the traffic between NapServer and NapUser, and between a NapUser and remote NapUser are maintained, and graphed in near-real-time and presented on our NetStats web site. After a period of time (e.g. 30 minutes), NapUsers are "retired" if they have not since talked with a NapServer. This reduces the likelihood of **FlowScan** misidentifying unrelated traffic as Napster traffic."[3]

The reports and modules that come with the **FlowScan** distribution specifically monitor Napster traffic, but it would be possible to generalize the approach for other file sharing programs.

Detecting Worms and Trojans

Finding the flows indicating active trojan programs (i.e. trojans actively passing traffic) does not necessarily require stateful inspection. One could simply scan flows for a suspicious port, for example, 31337. Incoming traffic with suspicious destination ports might be scans or initial connections to compromised hosts; outgoing traffic with a suspicious source port might be an exploit in progress. This sort of simplistic scanning would be relatively easy to do with the tools that come with both collector tools or with the **flowdumper** utility that comes with **FlowScan**. In either case, detecting a suspicious-looking port could be a signal for further analysis; the network manager might proceed by getting permission to do a host scan, for example. This sort of simple flow scanning can turn up false positives; using knowledge of the exploit to build more sophisticated filtering statements or use stateful inspection of flows would reduce the amount of mis-identified traffic.

An Example: The Code Red Worm

The following examples illustrate how the ad-hoc flow data scanning tools can be used to search logs for specific intrusions; the examples come from the **FlowScan** mailing list archive.[4] The first examples shows Dave Plonka's work on for detecting a successful infection by the Code Red worm.

Examining the files with the command **flowdumper –s**, the initial infection looks like this:

```
2001/07/19 10:17:14 invector.2179 -> infectee.80 6(PUSH|SYN|FIN|ACK) 7 4327
2001/07/19 10:17:14 infectee.80 -> infector.2179 6(PUSH|SYN|ACK) 5 212
```

After the target host ("infectee") has been compromised, it begins random-destination-IP scans of port 80:

```
2001/07/19 10:17:15 infectee.4321 -> 52.22.95.40.80 6(PUSH|RST|SYN|ACK) 11 4487
2001/07/19 10:17:15 infectee.4322 -> 91.167.212.99.80 6(PUSH|RST|SYN|ACK) 11 4487
2001/07/19 10:17:15 infectee.4323 -> 130.56.74.159.80 6(PUSH|RST|SYN|ACK) 11 4487
```

The following command selects just the records matching the initial infection "signature":

```
% flowdumper  -se  '6 == $protocol &&
                    80 == $dstport &&
                    7 == $pkts &&
                    4327 == $bytes &&
                    ($TH_FIN & $tcp_flags)'  raw_flow_file
```

This second example shows how Mark Fullmer, primary author of the **Flow Tools** package, used his tools to identify the infected hosts on his campus that were attacking www.whitehouse.gov:

First he created the filter listing the victim hosts' IP addresses:

```
% cat <<EOF > filter.acl
ip access-list standard wh permit host 198.137.240.91
ip access-list standard wh permit host 198.137.240.92
EOF
```

And then used the above filter in the following command line:

```
% flow-cat  raw_flow_files |  flow-filter  -f  filter.acl  -D  wh  |  flow-stat  -f9
```

Big Talkers
Just as the use of peer-to-peer or file sharing programs can indicate a possibly-compromised host, a formerly-quiet host that suddenly becomes a major consumer of bandwidth may have been hacked. **FlowScan** comes with a report, **TopN**, that shows the "big talkers" on a per-network basis for each five-minute flow sample period. A similar tool**, toptalkers.pl**, available via the **FlowScan** mailing list archive,[5] shows the overall "big talkers" through the network border. Similarly, the **Flow Tools** utilities **flow-cat**, **flow-filter**, and **flow-stat** can be configured to produce a report of top talkers.

Detecting Scans
A scan can be detected using a thresholding function. A host scan would appear as a source IP contacting more than a selected number of ports on a single host; a network scan would appear as a host contacting more than a selected number of destination hosts. Note that to reduce false positives, web-based advertisement servers and some game servers may need to be excluded from the list of possible scanning hosts. In the **Flow Tools** suite, **flow-dscan** can be configured to detect potential scans. For **FlowScan**, a scan can show up as a spike on a graph of flows.

Baselining and Firewall Planning
NetFlow can be used as a sort of passive host scanner. The OSU **Flow Tools** suite includes **flow-host-profile**; this tool builds a list of network services active (i.e. passing traffic) on each host. This allows staff to prepare a profile of network activities for a particular network or host, and then watch for changes, i.e. the addition of new services or hosts. It would also be interesting to look for changes in activity levels, particularly at increases. This sort of profiling works best, and is easiest for, small networks or groups with fairly static usage patterns[iii]

These kinds of network and host profiles can be used for firewall planning; knowing what hosts use what services to talk to which other hosts would be invaluable if one plans to add a firewall to a network. Knowledge of what services are in use can assist in managing user expectations and can assist in the construction of a policy that preserves functionality.

Data Integrity Issues
Flows are exported using UDP. Each of the UDP datagrams contains a 32-bit sequence

number, but the collector has no way to signal retransmission if it detects a missing NetFlow datagram. Data can be lost due to overloaded network segments between the exporting router and the flow collector, or due to overload on the collector itself. Either case of overload could be due to a benign increase in traffic, the bursty nature of NetFlow traffic, or due to an attack in progress. For the benign case, one could move the collector nearer to the exporting router or provide the collector with improved connectivity. For planning purposes, CAIDA says to expect thirty Ethernet MTU packets per second per DS-3 for V5 flow export, and approximately fifty packets per second for an OC-3. For collector overload, load can be split between collectors or data can be exported to more than one port on the collector, which will provide more socket buffer room. It is also possible that attackers could intentionally overload the collector or network segments between the exporting router and collector during an attack so as to obscure the attackers' other activities. In addition, attackers could conceivably spoof NetFlow datagrams; appropriate anti-spoofing filters or Unicast Reverse Path Forwarding can mitigate the problems.

In the case of some denial-of-services attacks, near-real-time analysis tools can fall behind, even if the collector manages to keep up. In some flood attacks the abusive traffic consists of many packets with different forged source or destination addresses. Each new source/destination pair will trigger a new flow, increasing the flow export rate dramatically.

Future Directions

As of December, 2000, the OSU staff was working on more powerful filtering tools. Alert capabilities for **FlowScan** are under development. It would be useful to be able to combine the best features of the tool suites built on both collection systems. Towards this end, the **Flow Tools** suite now contains a translator that allows it read **cflowd** flow dumps, and there is an experimental module that allows **FlowScan** to operate on **Flow Tools** data.[iv]

Conclusions

Cisco did not design NetFlow services with security analysis in mind; the problems inherent in the flow data demonstrate that. Nonetheless, NetFlow data has been and can be profitably used for security-related analysis. NetFlow can provide otherwise hard-to-gather information; it allows network managers to view traffic patterns without having to deploy sniffers or LAN probes on every segment. While neither silver bullet nor Swiss army knife, NetFlow data can provide network managers with a rich source of fairly compact data for security-related tasks.

References

Cisco NetFlow

Cisco Systems, "NetFlow Services and Applications"
http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm

Cflowd

http://www.caida.org/tools/measurement/cflowd/
http://www.caida.org/tools/utilities/rrdtool/
http://www.caida.org/outreach/isma/9901/slides/security_apps/mt0003.htm

OSU Flow Tools

ftp://ftp.net.ohio-state.edu/users/maf/cisco/

Romig, Steve and Fullmer, Mark "Cisco NetFlows and the OSU **Flow Tools** Package", Usenix LISA, December 6, 2000, URL: http://www.net.ohio-state.edu/security/talks.shtml#2000-12-06_osu-flow-tools_lisa

Romig, Steve, "Computer Security and CISCO NetFlow Logs at the Ohio State University", URL: http://www.net.ohio-state.edu/security/talks/1999-10-22_netflow_ohecc/ohecc.pdf

Romig, Steve "Distributed Attacks and CISCO NetFlow Logs", URL: http://www.net.ohio-state.edu/security/talks/2000-02-09_ddos-and-netflow_oartech/

FlowScan

Plonka, Dave "FlowScan: A Network Traffic Flow Reporting and Visualization Tool", LISA XIV, New Orleans, Dec. 2000, URL: http://net.doit.wisc.edu/~plonka/lisa/FlowScan/out.ps.gz
Other FlowScan URLS:
http://www.caida.org/tools/utilities/flowscan/
http://net.doit.wisc.edu/~plonka/FlowScan/
http://net.doit.wisc.edu/~plonka/list/flowscan/archive
http://net.doit.wisc.edu/~plonka/cflowd/
http://wwwstats.net.wisc.edu/.
http://net.doit.wisc.edu/data/Napster/

Leinen, Simon "Network Monitoring and Analysis" URL: http://www.switch.ch/tf-tant/floma/

Brumley, David "Detecting Network Scans", URL: http://www.theorygroup.com/Theory/scans.html

Cricket

http://cricket.sourceforge.net/

MRTG
http://people.ee.ethz.ch/~oetiker/webtools/mrtg/

RRDTOOL
http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/

Other router vendors
http://www.juniper.net
http://www.extremenetworks.com

Information about common trojan ports
http://www.robertgraham.com/pubs/firewall-seen.html

---

[1] http://www.net.ohio-state.edu/security/talks.shtml#2000-12-06_osu-flow-tools_lisa

[2] http://net.doit.wisc.edu/~plonka/lisa/FlowScan/out.ps.gz

[3] http://net.doit.wisc.edu/data/Napster/

[4] http://net.doit.wisc.edu/~plonka/list/flowscan/archive/0933.html

[5] http://net.doit.wisc.edu/~plonka/list/flowscan/archive