DAVID JOSEPHSEN

iVoyeur: mystical flows

David Josephsen is the author of *Building a Monitoring Infrastructure with Nagios* (Prentice Hall PTR, 2007) and Senior Systems Engineer at DBG, Inc., where he maintains a gaggle of geographically dispersed server farms. He won LISA '04's Best Paper award for his co-authored work on spam mitigation, and he donates his spare time to the SourceMage GNU Linux Project.

dave-usenix@skeptech.org

YOU KNOW YOU HAVE A HEALTHY, well-implemented monitoring system when people in other departments begin approaching you with their information requirements. Monitoring is usually something we prefer to do ourselves. It's just unlikely that some other organizational unit will collect the information you want in the way you want it collected without making life difficult for you in the form of bloated, unstable, and/or insecure agent software. It's rare in my experience for tech staff to even consider whether anyone else in the organization is already doing systems monitoring before implementing their own tools.

So when someone voluntarily comes to you and asks if your system can monitor this or that, you know you're doing things well. By this metric I've had more than my share of failures thus far in my career. I take solace in the words of an old martial arts instructor of mine who once said, "Nobody ever learned anything from winning a fight," and man, have I learned plenty. But rather than dwell on my "temporary setbacks," I'd like to share with you a success I've had, because I think there's something to be learned from it as well. My crowning achievement—arguably the pinnacle of my success in implementing monitoring systems—came one day when I was asked by a Microsoft SQL Server DBA for Nagios to monitor the tablespace on his database servers. It doesn't sound like much, but then you haven't met my DBAs.

This came shortly after a meeting to determine "why the monitoring system was sending false alarms to the DBA teams." Even given the meeting subject, I was not expecting the hostility that greeted a fellow sysadmin and me when we walked into the conference room. The DBAs had brought pages of specific instances of "false alarms," and I hadn't even brought a laptop. Oops. So I must have looked a bit panic-stricken when the VP of software development passed out a copy of the monitoring system's "errors" and began to discuss with those in attendance how the "obviously flawed" system was sending bogus disk capacity warnings and that if it couldn't even monitor disk capacity correctly, it would need to be replaced. They already had their own monitoring software, so they weren't bucking for budget. Something political was probably going on, but to this day I'm not sure what their actual goal was.

I was only a few words into stammering out something like "I'll have to look into this and get back to you" when my teammate, who hadn't forgotten his laptop, interrupted with "These aren't false alarms." He had generated a Nagios trends graph of the disk service for one of the database servers and was visually correlating this with an RRDtool graph of disk utilization for the same server. He could see at a glance that a disk partition was filling up every few days at a certain time, and then was being emptied shortly thereafter.

Faced with the graphs, the VP turned to the DBA team and asked, "Well, is one of you clearing space on partition X on server Y every Tuesday and Thursday?" A moment of silence and head-shaking ensued followed by a "Huh?" from the corner of the room. It came from a DBA who until this moment had been face down typing furiously into a laptop. The VP repeated his question, to which the DBA replied, "Yeah I clear out a bunch of temp files whenever I get notifications from the monitoring system. It happens a couple times a week. I put in a request for more SAN but haven't heard back yet." Evidently he hadn't paid much attention to the meeting subject line either.

From that day on, the DBA team projected upon the monitoring system a sort of mythical all-knowingness. They often assumed we had data that we didn't (though when they did this we usually quickly added it; there's little difference between presumption and permission IMO). I was in the hallway on the way back to my desk from that meeting when one of the DBAs approached and asked if he could get added to the "table-space notifications." We weren't monitoring the table-space at that point, but needless to say we began that afternoon. Our DBAs as a group were a fiercely protective bunch. I took it as a huge compliment that he had assumed we were monitoring the innards of their precious databases and that he was OK with that.

There are a handful of monitoring technologies that can pack large amounts of very specific, historically relevant data in an easy-to-use, accessible format. These are the tools that give your monitoring system an all-knowing air, which, as I learned from this episode, is a wonderful thing to have on your side. So, this being a security-focused issue, I thought I'd take the opportunity to talk about one of the best of this class of tools: NetFlow. Just as RRDtool made us witch doctors to the DBAs, NetFlow data can make you a mystic to your security and NOC staff.

NetFlow began life as a Cisco proprietary protocol for traffic accounting information. There is a fledgling industry standard called IPFIX [1], which is based on Cisco NetFlow v9. This standard, defined in RFC3917 [2], is generally reverse-compatible with Cisco's proprietary NetFlow protocol without modification, and it has already been implemented by several routing vendors.

Most modern routers from Cisco and Juniper, as well as various open-source implementations such as pfflowd [3] and nprobe [4], can export NetFlow data. It is not, however, supported by Cisco PIX firewalls or Cisco switches. Cisco Layer-3 switches such as the 6000 series can export NetFlow data, but since such switches offload and cache routing decisions to ASICs (Application Specific Integrated Circuits), the flows for packets that traverse the routing processor may be in a different format from those processed by the ASICs. In some cases it may not be possible to export flows for any packets but those that traverse the routing processor, so NetFlow data from these devices may be incomplete.

The overall NetFlow architecture may be superficially thought of in terms of a specialized, task-efficient syslog implementation. Routers or routing sys-

tems emit UDP NetFlow data to one or more centralized NetFlow collectors, where they are aggregated, stored, and possibly processed. There is no transport-layer encryption or signing. NetFlow emitters are called "probes." Probes are generally given the network socket of a listening collector and may be configured with options that change the details of the flow data.

A flow is loosely defined as a series of related, unidirectional packets, which represent half of a two-way conversation between two network entities. NetFlow data contains summary metadata about the connection it represents, and therefore flow data is only exported to the collector once the flow has ended and can be summarized. By default in Ciscoland, a flow begins when the relevant traffic is first detected and ends when one of the following criteria is met:

- For TCP traffic, when the connection is terminated (e.g., an RST or a FIN is encountered)
- When no related traffic has been seen in the last 15 seconds
- When the flow has continued for more than 30 minutes
- When the memory buffer containing the flow has filled up

The IPFIX standard allows for some user-defined criteria for detecting the beginning and end of a flow. In my experience, the Cisco criteria are usually sufficient.

Flow summary data is encapsulated into a flow record. Every flow is unique but may be represented by multiple flow records if, for example, the router's memory buffer fills up before the connection is terminated. Flow records are really great; they contain oodles of info about the connections they represent, including source and destination IP and port numbers, protocol type, type of service (TOS), number of octets and packets transmitted, time/day stamps for the beginning and end of the flow, source and destination AS numbers, input and output interface names, and even a bitmask representing the TCP flags that were set during the connection.

You may have noticed that flow records don't contain any application-layer data, but the network-layer data that's available is more than enough to get great visibility into what's happening on the network as well as detecting some much-hated and historically difficult to diagnose problems and attacks such as DDoS, worms, and viruses. But I'm getting ahead of myself. First, let's talk tools.

There are quite a few NetFlow collectors out there, some of which are commercial, such as Cisco's NetFlow Collector (NFC) [5], and hundreds of open-source tools of all description. When I look for tools in a monitoring context, I tend to optimize for flexibility so that I can easily add the data in question to the existing monitoring interface. I abhor one-off interfaces for every little thing, and for this reason I'm likely to choose lightweight command-line tools that don't have a lot of dependencies and don't make it difficult for me to get at the data.

There are at least a couple of new NetFlow tool papers a year, and I don't keep up with them as well as I probably should, because Ohio State University's flow-tools package [6] is a category killer for me. OSU flow-tools, a collection of small single-purpose tools that are designed to interoperate with each other via pipes, run the gamut of everything you might want to do with NetFlow, including collect data from a probe, "tee" data to real-time analyzers, perform query-based analysis on archived flow records, replay archived flows, and reassemble connections contained in multiple flow records. Judging by the quantity of graphical front-ends for visualizing data from flow-tools, I'm not the only one who considers it a category-killer.

The flow-tools install is a typical ./configure && make && sudo make install. Then, your routers need to be configured to export their NetFlow data. If they are Cisco routers, the following should work:

```
ip cef distributed
ip flow-export version 5 origin-as
ip flow-export destination 1.2.3.4 9800

interface FastEthernet0/1/0
 no ip directed-broadcast
 ip route-cache flow
 ip route-cache distributed
```

Once the data is being exported, the flow-capture tool can collect NetFlow data from the routers and archive it to disk. The flow-capture tool requires only a localip/remoteip/port tuple, and it automatically handles log file naming, compression, and rotation. Command-line options can change most aspects of its behavior, including the -D switch, which forces it not to daemonize so that you can run it under daemontools or the superserver of your choice. It's possible to connect to the flow-capture daemon on a TCP port to receive a real-time data feed, suitable for feeding to your favorite parsing engine as well.

Several tools in the package can do creative things with the flow data as it arrives. Flow-fanout is a tool for redistributing the data via the NetFlow protocol to additional collectors, and flow-mirror and flow-rsync copy the flow logs themselves to backup collectors.

The backbone of NetFlow analysis is the combination of the tools flow-cat, flow-filter, and flow-print. Because flow-tools writes flow data to binary log files, which are also optionally compressed, and each log file has a metadata header, a special cat tool called "flow-cat" is required to concatenate them. Output from flow-cat may be passed directly to flow-print, which outputs the records in a human-readable format, or it may be piped first to flow-filter, which filters the output using the criteria of your choosing.

Before you reach into your bag for cut, sort, awk, and grep, I should mention flow-stat. This tool can sort, summarize, and segment the output from flow-filter. There's a bit of a learning curve, but if you play around with it for a while I think you'll find a few formats you like, and once you have them in your head, the tool will save you a bunch of time. Formats 8, 9, and 10, the IP source/destination-based formats, are the ones I tend to use the most often. It's worth reading the man page to get an idea of what other formats are available.

Several conversion programs exist to move the data to external formats. The flow logs may be exported directly to delimited ASCII formats of various types with flow-export, the headers may be viewed with flow-header, and there are even a couple of tools included for exporting the data directly into RRDtool.

There are several special-purpose analysis tools such as flow-report and flow-dscan. With flow-report you get summary statistics in a predetermined format for a given collection of NetFlow logs. The IDS flow-dscan is intended to spot aberrant behavior in real-time traffic flows. But I'll leave it to you to play with those two; I'd like to focus on the core tools and show you some ways they can help you gain some visibility into your network traffic.

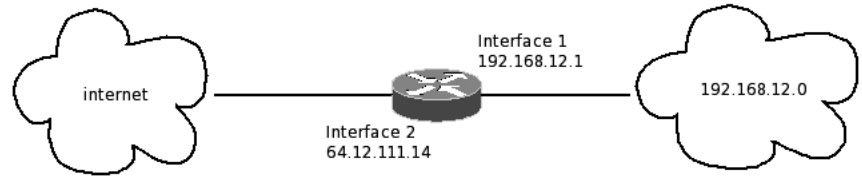

FIGURE 1: A ROUTER SEGMENTING AN RFC 1918 NETWORK FROM THE INTERNET

Let's start with a relatively simple example. The router in Figure 1 segments an internal 192 network from the "internet." Assuming the flow records for this router were in /var/flows/, with the following command you could find all of the internal Web servers that serviced external hosts:

```
flow-cat /var/flows/ | flow-filter -i2 -P80 | flow-stat -f8
```

In pseudocode, that's "filter the flow data for flows coming into interface 2 (-i2) destined for port 80 (-P80) and format the reports using the destination IP format (-f8)." Adding an "-S3" (sort on field 3) to the flow-stat command would have sorted the resulting report by the host that sent the most data.

The "-i" and "-P" switches are reversible via case sensitivity. In other words, had I specified "-I" instead, I would have filtered the data for flows exiting interface 2 instead of entering interface 2, and had I specified "-p80" I would have filtered the data for flows originating on port 80 instead of destined for port 80.

As you can probably imagine, flow-filter can also filter on host IPs and network ranges. To make this a bit simpler on the command line, you can create an ACL file and give macro-style names to IPs and ranges using Cisco-standard ACL syntax. For our network we might create the following macros in a file called my.acls:

```
ip access-list standard inside permit 192.168.12.0 0.0.0.255
ip access-list standard not_inside deny 192.168.12.0 0.0.0.255
```

Then we could, for example, find the top 10 bandwidth users on our network with something like:

```
flow-cat /var/flows | flow-filter -f./my.acls -Sinside | flow-stat -f9 -S3 | grep -v \# |
head -10
```

Perhaps, upon finding that the top bandwidth user was 192.168.12.42, you might want to know to which hosts this user was sending data. After adding

```
ip access-list standard topDude permit host 192.168.12.42
```

to our ACL file, the command

```
flow-cat /var/flows | flow-filter -f./my.acls -StopDude | flow-stat -f8
```

enables us to find out what ports this user is connecting to by using "-f5" in the flow stat command here. Upon finding out that topDude's traffic was destined for port 1434 (which is slammer worm behavior) on various remote hosts, we might search for any other internal hosts exhibiting the same behavior with, for example:

```
flow-cat /var/flows | flow-filter -f./my.acls -Dnot_inside -P1434 | flow-stat -f9
```

Since NetFlow data is in an offline-archived format, this technique can give you answers without requiring that you talk to the router directly, which is sometimes not possible. In scenarios such as DDoS attacks, where the NOC staff will be spending its time waiting at router ssh prompts trying to get a

handle on what is happening, NetFlow makes it trivial to quickly isolate the offending traffic and take action. At the same time it's a powerful capacity planning and forensics tool, putting months or years of detailed traffic information at your fingertips.

The NetFlow/flow-tools combination as a solution is very scriptable, easy to install, and gets along excellently with popular monitoring tools, including RRDtool and Nagios. But its "magic smoke" lies in the means it gives you to answer very specific questions about the network in an "on-demand" fashion. With NetFlow, you can create facts from hunches and make decisions out of options in a few keystrokes—the kind of thing that soothsayer reputations are built upon. If you aren't using NetFlow or something like it currently, you're missing out, and I highly recommend you give it a try.

Take it easy.

REFERENCES

[1] http://tools.ietf.org/wg/ipfix/.

[2] http://tools.ietf.org/html/rfc3917.

[3] http://www.mindrot.org/projects/pfflowd/.

[4] http://www.ntop.org/nProbe.html.

[5] http://www.cisco.com/en/US/products/sw/netmgtsw/ps1964/.

[6] http://www.splintered.net/sw/flow-tools/.