DAVE JOSEPHSEN

iVoyeur: pockets-o-packets, part 1

Dave Josephsen is the author of *Building a Monitoring Infrastructure with Nagios* (Prentice Hall PTR, 2007) and is 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

THE WIND IS EVERYWHERE, A CONSTANT roar that threatens to rip my hat off my head and send it tearing 500 feet into the canyon below. Even the birds can't seem to fly in it, so I'm surprised by the shadow cast by the turkey vulture that's now circling us above the pass. It's the first bird I've seen aloft in days, and it's obviously having a hard time remaining so. We're nearly to the top of the Caprock Canyons south prong when my pager goes off. I glance up at my wife who, thankfully, didn't hear it through the wind.

I take the pager from my pocket and have a quick look. Office users are complaining of network slowness; I should ignore them. This is supposed to be a day off in a place 300 miles distant. A place of rocks and juniper. But a small voice in the back of my head says simply, "But you can." It's true . . . here in the middle of a scramble up to the ridgeline and despite this infernal wind, it's possible for me to diagnose network latency at the office. I can. The thought simultaneously amuses and offends me.

I bring up the ssh client on my pager, ssh to the pcap box, and run a quick racluster [1] command. Seeing the result, I hastily type my reply ("Tell Larry to stop downloading My Little Pony episodes"), just before my wife looks down and asks if I'm okay.

"Yep," I shout in reply.

"Tell me you're not on that pager," she yells, raising an eyebrow.

"I'm not," I shout, returning it to my pocket.

As we continue our hike my mind mulls all the pieces that make what just happened possible, from the cellular infrastructure back on down to Ken and Dennis. So much work. It occurs to me to wonder if it's a miracle or a curse. I suspect the latter but can't say for sure. One thing I do know, however, is good article fodder when I see it, so let's spend the next couple of issues talking about the pieces of a decent packet capture framework.

Collecting IP packets for offline analysis is a bread-and-butter sort of monitoring infrastructure. If you aren't centrally collecting packets, you probably have lots of little tools that work for a single type of device and can tell you only about a small piece of the network. One gains a lot from centralizing packet and flow data. IDS/IPS, network utilization

and trending, and a slew of other buzzword activities become greatly simplified, and efforts that used to be complicated and intertwined can be made to function to each other's benefit.

For example, Snort IDS alerts are great [2]. They catch all sorts of questionable network behavior, but without a centralized packet repository and the tools to analyze it, these alerts lack context. For example, is it in fact aberrant that server A conducted what appears to be a replay attack against server B, or is that just what happens the first Wednesday of every month because of some weirdo backup software doing weirdo things? If the packets are localized in such a way that Snort and Argus [1] can both use them, then you don't need to spend hours running down context behind Snort alerts (if you can at all).

Having an offline packet repository can really be a life-saver in all sorts of ways. Getting DoS'd and can't log into the router? Ask Argus. Broken project planners asking for answers by tomorrow to questions that take months to answer? Ask Argus. Ex-girlfriend who also happens to be the head NOC sysop ignoring your BGP looking-glass RFIs? Ask . . . well, you get the point.

The easiest way to collect IP packets in a central location and to ensure that the greatest number of tools can make use of them is, in my opinion, to simply redirect them all to a single interface (or series of interfaces for larger environments) on a single host. This can be a bit of a trick, but it can be done, and once you're there, you're golden, because pretty much any tool designed to analyze packet traffic can listen to a named interface.

Normally, the goal is to capture any packet that traverses a network segment, and for most folks there are three ways to do that. Assuming you use Cisco gear or something else that supports netflow, you could use flow data instead of raw packet dumps and export the flows to a pcap box. Several tools, including Argus, can read flow data, but this does limit your options later on and incurs a bit of utilization on the router or firewall in question.

Next, you could use a span port on the switch. Span ports are great; you get real packets, you can consolidate packet dumps from several devices to one port, and they don't cost anything extra. Their primary disadvantage is that they may impact the performance of the switch, and this is highly architecture-dependent. A breakdown of span port impact on performance for various Cisco switch architectures may be found at [3]. If you have a mid- to high-end Cisco switch, you're fine.

The third and most expensive is a hardware network tap. These are really great; they're inserted between a device and the switch and provide a duplicate of every packet on a separate port (or set of ports). We use aggregating taps from NetOptics [4]. They're rack-mountable boxes that can tap multiple 10/100 links and aggregate them all to a single 1gbps link. They fail open, so they're not a point of failure if something happens to them (short of physical explosion). There are much larger, much more expensive taps [5] that can aggregate multiple gbps interfaces, for ISPs and very large environments (you guys know who you are), but in my experience it's easy to overestimate what you actually need here. Most environments, surprisingly, can get their pcap traffic down to a single interface on a single box without much trouble.

If you use software routers, then you have an additional option: a software tap. We use OpenBSD routers quite a bit, and I very much like daemonlogger [6] for this purpose. Daemonlogger, written by Marty Roesch (who also wrote Snort), can be thought of as a daemonized tcpdump. It listens to a network interface and either logs the packets to disk or sends them to a remote machine. Daemonlogger comes in handy on the pcap machine too,

since it's likely the pcap machine will need to be plugged into a combination of span ports, network taps, and other devices. Daemonlogger can be used to consolidate all of these interfaces by starting an instance per interface and telling them all to forward to the same interface. It also comes in handy for those super-sensitive boxes for which it's not good enough to only capture traffic that traverses a network segment. If you need every packet that a given database server sends and receives, Daemonlogger is a great way to go.

Since our goal is to aggregate all of our captured packets to a single interface, we should put some thought into that interface. The DAG network cards from Endace [5] are just the thing here. They're expensive but are generally considered to be the best available for pcap and network audit work [7].

So now that we have packets from span ports or flow data or taps or all of the above coming to our pcap box, what do we do with them? There are many answers to this question, but the first three that spring to mind are Argus, Snort, and Daemonlogger (yet again). And none of these are mutually exclusive; all of these tools can listen to the same port at the same time and get what they need, provided the machine has the horsepower to run them. Some other popular answers, in no particular order, are Wireshark [8], NTOP [9], and Bro [10].

I mention Daemonlogger here again because in its disk-logging mode it writes binary pcap files, just like tcpdump, and has built-in options for log file naming and rotation, so it's a great way to provide a lowest-common-denominator online archive of pcap data. Every tool that works with packet-data supports this format, and Daemonlogger is so lightweight it's just about free.

Snort and Bro are both awesome IDS tools that will do a bang-up job listening to the pcap interface. In our setup, Snort is configured to listen to the pcap interface and alert via syslog.

Argus describes itself as a Real Time Flow Monitor that is designed to perform comprehensive data network traffic auditing. In my opinion it's about the coolest network-centric monitoring tool that was ever invented and the entire reason this infrastructure should be built.

Argus may be run as a daemon, reading live packets from a network interface, or as a user program, reading packets from a packet capture file. The default behavior is to run as a daemon, which is what we're interested in here. Point the Argus daemon at your pcap interface, and it'll read in and transform the incoming packets into stream data which it then stores in a database. To analyze the data you need the Argus client ra ("read Argus"), which is available as a separate Argus-clients package on most OSes and distros.

The client programs may either read from an Argus server's data files on localhost, or from a remote Argus server if the server has been set up to listen to remote requests. If you want Argus to listen over the network for client requests, simply pass it a `-P` switch. There are obvious security ramifications to doing this, so Argus may be compiled with SASL support to provide authentication and authorization.

Ra has several partner programs, the two most important being rasort and racluster. There are also a slew of third-party Argus clients to do everything from logging to graphing and visualization. The sky is the limit with the Argus clients; you can interact with them to discover anything you might want to know about the network traffic contained within them. Questions such as, "How much data was transferred in the last 20 min?" "Who are the top 10 users of the bittorrent ports?" and "What hosts are trying to infect

other hosts with virus X?" are all straightforward queries to racluster and rasort.

Stay tuned for much more detail on Argus in my next article, including file management basics and a primer on using ra. Between you, me, and the vulture, the racluster command I typed on the south prong trail was:

```
racluster -M rmon -m saddr -r <my_data_file> - ip | rasort -M bytes -r - -w - | ra -N 10
```

. . . or, in English, "Give me a list of the top 10 bandwidth users sorted by byte-count in the last hour."

Take it easy.

REFERENCES

[1] Argus Real-Time Flow Monitor: http://www.qosient.com/argus.

[2] Snort IDS: http://www.snort.org.

[3] Catalyst Switched Port Analyzer (SPAN) Configuration: http://www9.cisco.com/en/US/products/hw/switches/ps708/products_tech_note09186a008015c612.shtml.

[4] NetOptics hardware taps: http://www.netoptics.com/.

[5] Endace high-speed packet capture devices: http://www.endace.com/high-speed-packet-capture-hardware.html.

[6] Daemonlogger: http://www.snort.org/users/roesch/Site/Daemonlogger/Daemonlogger.html.

[7] Endace recommendation details: http://www.qosient.com/argus/sensorPerformance.htm.

[8] Wireshark Packet analyzer: http://www.wireshark.org.

[9] NTOP Network Management framework: http://www.ntop.org/news.php.

[10] Bro IDS: http://www.bro-ids.org.