[Optional] Network Visibility with NetFlow

COSC301 Laboratory Manual

> **Old Lab**
>
> This lab hasn't been run recently, so no guarantees are made regarding the commands and their output.

In this lab, we shall investigate network visibility using the NetFlow architecture. There are many tools available for NetFlow, so we shall first review the NetFlow architecture and then perform some requirements analysis to see which set of tools will be most suitable for us, looking briefly at some of the tools available to us.

1. NetFlow Architecture

NetFlow is conceptually very simple: devices such as routers or switches have some *probe* configured which looks at the packets going through its configured interfaces. These observed packets are collected into *flows*. A flow, in the case of TCP, is all the packets that make up a particular connection; for UDP, a flow is everything with the same source and destination address and port. This description of a flow is a simplification, but is intuitively what is meant.

Periodically, such as when a flow is completed (eg. a TCP connection has been torn down) or after some period of time, the flow is exported to a configured *collector*. This is where the NetFlow protocol is used, which we shall discuss in more detail in a following section.

The collector process will generally run on a powerful machine with a lot of available disk space. The collector will often simply just write the exported data to disk, perhaps filtering to capture only certain data, and typically performing maintenance such as rotating files and expiring (deleting) old capture files. You may need to have multiple collectors in order to process all the data exported from all of your probes, so you can end up with some relatively more complex architectures to ensure you can process the incoming data fast enough. NetFlow is a UDP protocol, so one of the things a NetFlow administrator will want to do is monitor how many UDP packets are being dropped by the collector[1]; this can be in indication that the collector host is either not fast enough or that the collector process is not given a high enough scheduling priority.

The data stored by the collector(s) can be used for various purposes, such as accounting, traffic analysis and diagnosis and for investigating security incidents such as Denial of Service (DoS) attacks. We shall look further at some of these later in this lab.

2. NetFlow Versions

There have been numerous versions of NetFlow over the years, many of these were to add extra fields to the flows exported by the probe, which were needed for grouping (*aggregating*) based on various attributes, such as which switch port the data entered on. This led to an explosion of various sub-versions (8.x), leading finally to a more flexible version 9 and finally

[1] This can be done using **netstat -s**.

to the open standard IP Flow Information eXchange IPFIX, which is still yet to see much support in vendor offerings.

To keep this section free of unneeded detail, here are the versions most commonly used:

Version 5
This is perhaps the most common and is generally the default version configured.

Version 7, 8
These are generally Cisco specific and add various fields. Versions 9 and 10 (IPFIX) have replaced them.

Version 9
This includes support for IPv6 and MPLS (outside the scope of this paper). The protocol format is also more flexible.

Version IPFIX (aka. Version 10)
Based on version 9, this is the IETF-based version. Allows for some enterprise-defined extensions and enhanced packet format.

3. Requirements Analysis

There is a lot of software available for implementing various parts of the NetFlow architecture. Most of the products are for various analysis tasks, but there is still a wealth of choices for collectors; a bit less for probes, since most probes will come with routers operating system (eg. Cisco IOS), although this feature may be limited to higher-end devices in some vendor offerings. Our problem is to figure out which software is best for us; to do this we need to do some requiremnts analysis. To answer this question, we need to first figure out what features are important, and which software satisfies the various features.

Table 1. NetFlow Probe Selection

Software	v.9?	Linux?	Debian?	Maintained?	Comm. Opinion	Sampling?	Source	Cost
fprobe-ulog	N	Y	Y	2005	dead	Y	F/wall	Free
softflowd	Y	Y	Y	~	Good	Y	pcap	Free
nProbe	Y	Y	nTop	Y	Y	Y	pcap	€99.99
pmacct	Y	Y	Y	Devel	Y	Y	either	Free
ipt_NETFLOW	N	Y	N	Y	~	N	F/wall	Free

Considering that we really want version 9 support for IPv6, we can choose from softflowd, nProbe or pmacct. pmacct is developmental, so may be harder to support and may have less stable community experience. nProbe seems very good, particularly aimed at performance, but the cost is quite high; although nProbe is available free for universities and research, I'd rather not have to deal with licensing issues. That leaves us with softflowd, of which I found at least one comprehensive article about building an integrated NetFlow system around, and is also widely available.

Retrospectively, softflowd does do a good job, although I do wish it would export incoming and outgoing interface; unfortunately libpcap based probes can't really get that information; it would need to be more tied to the routing subsystem.

softflowd had NetFlow version 9 support added two released ago, so should be fairly stable. It is also available via package repository, so keeping up-to-date should be less of an issue, although the software seems to be in a stable-but-slowly updated phase of its life, due to developer priorities. **softflowd** also has one extra feature that may be rather nice: the ability to read in a pcap file (as produced by **tcpdump**) and emit NetFlow records based on it, which is useful for testing and evaluation.

Okay, so we have an apparent winner in the selection for the 'probe' component of our NetFlow architecture: we still require a 'collector' and some analysis or reporting tools, which there are numerous commercial and open-source products for.

Using NetFlow version 9 as a must-have requirement, we can immediately discount much of the field, much of which may be otherwise commonly used. One fairly serious contender, which is free, is **nfdump**. This is a tool written by people actively working and using the product in projects such as Internet2, which feature technologies such as IPv6 and other assorted goodies, so it is reasonable to think that it should be capable of satisfying our needs — but note that we don't yet know how hard it might be to use. A look at resources such as mailing list archives, on-line bug reports and release history confirms that it is actively maintained and developed, with a stable and developmental branch and a friendly and responsive developer. **nfdump** can be integrated with a larger product, called NfSen, which is a web reporting frontend, though it is not required.

So now we have some components that should be suitable for the 'probe' and 'collector'… but what about the reporting and analysis tools? To answer that question, we need to do some further requirements analysis as to what we want to do with the data.

Some likely activities include network accounting / monitoring / diagnosis. Using NetFlow we can ask questions such as:

- Who (is/has been) using the bandwidth the most?

  This could be used as a data-source for targetted education of, say, users that are using resources inappropriately.

- How much data did user/computer/group X send/receive internationally/domestically/locally?

  This could be used for billing or, if such reports are generated frequently, to alert someone about potential budget blowouts.

- How much of our bandwidth use is comprised of application X?

  Perhaps I could use this data to make an informed decision about deploying caching data to move data off my expensive WAN links.

- Has our change to X reduced/impacted network load as expected? Are there any unexpected consequences?

  This is generally, like many of these, a graphing/trending application, but may also be facilitated by a weather-map.

- Are the any unusual spikes that might indicate security incidents such as Denial of Service or internet worms?

  This is a common use of NetFlow accounting data in a security setting.

As we only have limited time available, we shall concentrate on the easier areas of accounting, although there is plenty of scope to expand. It is also an area that is to be most likely in smaller networks. Note that many areas commonly require a mix of pre-determined and ad-hoc reporting facilities:

- "Hello, I'd like to know why my network bill is so high…"

  The network bill would be a pre-determined report, but the ability to drill down (ie. look at increasing levels of detail) may very likely need to be ad-hoc.

- "Okay, so it seems host X was broken into by an exploit on port Y… what other machines may have been affected this way?"

  The detection of an exploit may not have come from a NetFlow based report, but investigating traffic to port Y would generally require some degree of ad-hoc reporting, which could be done by a pre-defined parameterised report.

- "Hmmm, can we recognise new applications on our network that we don't currently track?"

  It is common to try and track the amount of data used by protocols such as HTTP, SMTP, FTP (in its various modes) and others, but there will be some Other category, which you might like to take a closer look at occassionally.

- "Argh! What is causing these spikes in traffic?"

  In this situation, NetFlow itself may not be enough, because NetFlow only gives you packet-level statistics, it doesn't give you the packet data itself. But that said, NetFlow can give you enough visibility on the network to begin making intelligent hypotheses as to what might be happening, and then you can introduce more ad-hoc monitoring in the parts of the network which are seem more able to answer the question. Finally, note that many of these LAN-activity questions will be best answered using NetFlow data from switches, of which there is likely to be an excessive amount of data.

- "Why are my Voice-over-IP sessions so slow sometimes?"

  You might use NetFlow to determine what else was happening in the network when the problem manifests itself. Perhaps there is a lot of traffic from machines automatically downloading patches or submitting backups, perhaps there are people viewing YouTube videos at the time, or perhaps someone is sending photos via e-mail which is choking upstream bandwidth. These are all questions that NetFlow can help you answer quickly.

A suite of programs called flow-tools are a common free solution for these tasks; generally coupled with other tools to do things like graphing. FlowScan is another common free tool that can do some useful reporting such as producing a graph showing the composition of traffic — such as web traffic, FTP, etc. — over time. Unfortunately, neither of these tools yet support NetFlow version 9, so we shan't cover them.

Fortunately, **nfdump** has some rather useful querying facilities, including a filter mechanism similar to **tcpdump**, that supports many of the ad-hoc queries we may wish to make, including Top-N style questions. Therefore, we shall use **nfdump** for some of our reporting and querying examples as well as our 'collector' component.

This leaves us with at least one part of our reporting system with which there are surprisingly few available tools: the accounting system. This is because such systems are commonly: complex, customised, vary greatly according to a) how the accounting data is collected, b) policies by which accounting is done, c) integration with billing and payment systems. Note

that there can be a reactive quality to b and c, such as what may happen when you introduce a quota policy. As such, accounting systems are often developed in-house; we shall see how we could use **nfdump** to collect the basic data, but processing the data and making that into a report is an exercise left to the reader.

4. Install, Configure and Test the Probe

*In this section, we shall install **softflowd** from a package repository, configure it appropriately and test that it is working.*

The probe needs to be installed either on a router, switch, or attached to a port on said device though which a copy of every frame is sent; such a port is commonly referred to as a 'mirror' or 'SPAN' port. On our Linux router, we shall be installing the probe on the router itself. Most commercial routers will have a probe facility available already installed.

Install the **softflowd** package on your router using the standard tools available.

```
# apt-get install softflowd
… note that it doesn't start yet …
Not starting softflowd
Edit /etc/default/softflowd and define the INTERFACE variable
```

The software does not yet run because it cannot run without being configured to listen on a particular interface. Because the software is based on the same software (libpcap) as **tcpdump**, we cannot listen to multiple interfaces at once. If you wanted to, you would need to start two instances of the program, which would generally involve duplicating and changing the startup script.

But which interface should we listen on? This will depend on what sort of questions we want to ask, as this affects the sort of traffic that we see. If we listen on the outside, then we see all of the traffic coming to us including the traffic that gets dropped by a packet filter on the outside interface. This is useful for determining DoS activity etc, but will not be useful for answering questions about traffic purely on the inner network or directed to the inner interface of the router.

Another, rather major, implication is the affect of NAT. If we listen on the outside interface, we see the packets after they have been Source-NAT'd, which means we see all traffic as coming from the router, not the machine itself. Clearly this is not useful for tasks such as accounting, so we shall listen on the inside interface.

As root, edit the file `/etc/defaults/softflowd`. Set `INTERFACE` to `inside` and `OPTIONS` to `"-n 127.0.0.1:9995"`.

Start **softflowd** using the standard startup script.

```
# /etc/init.d/softflowd start
Starting softflowd: softflowd.
```

Verify that it starts, and that it logs nothing of concern to syslog. You should see something like the following:

```
… softflowd[…]: softflowd v0.9.9 starting data collection
… softflowd[…]: Exporting flows to [127.0.0.1]:9995
```

We can verify that it is sending data using a tool such as **tcpdump**. After you start this, you will need to either wait for a while for some expired data to be sent, but it is easier to simply generate some yourself. You should see something like the following:

```
# tcpdump -n -i lo port 9995
listening on lo, link-type EN10MB (Ethernet), capture size 96 bytes
16:16:01.830727 IP 127.0.0.1.54991 > 127.0.0.1.9995: UDP, length 600
16:17:01.490783 IP 127.0.0.1.54991 > 127.0.0.1.9995: UDP, length 1464
16:17:01.491298 IP 127.0.0.1.54991 > 127.0.0.1.9995: UDP, length 984
```

Further activities regarding the probe would be related to performance monitoring and tuning, but that is an activity best left till the other components have been put into place.

5. Install and Test the Collector

*In this section, we install the **nfdump** product, test that it is receiving and storing the data, and verify that the data from the probe appears reasonable.*

The **nfdump** package is a suite of tools, one of which is **nfcapd**, which is the collector, and **nfdump** which is the display and analysis program. There are some other tools included as well, but those are the major commands we need to know about.

Install the **nfdump** package. After you install, check to see if any new processes are running, you should see a new process called **nfcapd** running. Let's see how it was started.

```
$ ps axo command | grep '[n]fcapd'
/usr/bin/nfcapd -D -l /var/cache/nfdump
```

Okay, and which port is it listening on (different probes default to different port numbers)?

```
# lsof -Pni | grep nfcapd
nfcapd    25755        root    3u  IPv4 2570199       UDP *:9995
```

Great, so now we have verified that the collector is working and listening on the same port as the probe is exporting to. Optionally, we could also verify that **nfcapd** is receiving the data, but you would only typically bother doing that if you were diagnosing any problems, or if you were just curious.

```
$ pidof nfcapd
25755
# strace -p 25755
Process 25755 attached - interrupt to quit
recvfrom(3, "\0\5\0\31\0\36J\217K\0273\211…"..., 65535, 0, NULL, NULL) = 1224
time(NULL)                              = 1259811721
recvfrom(3, ^C <unfinished ...>
Process 25755 detached
```

Note that you would need to wait a while for some data to be produced, when there is some data produced, it will be returned via the second argument to `recvfrom`. But what is the data saying? To answer that, let's use some very basic querying to check what it is recording.

```
$ nfdump -R /var/cache/nfdump/ | head -5
Date flow start        Duration Proto Src IP Addr:Port     Dst IP Addr:Port      Pkts   Bytes Flows
2015-01-08 16:19:14.592 35.895   TCP  139.80.206.169:3142 ->192.168.1.11:60041   1736   5.0 M  1
2015-01-08 16:19:14.592 35.895   TCP  192.168.1.11:60041  ->139.80.206.169:3128   460   27291  1
2015-01-08 16:20:16.625  8.619   TCP  139.80.206.169:3142 ->192.168.1.11:60041  27142  49.0 M  1
2015-01-08 16:20:16.625  8.619   TCP  192.168.1.11:60042  ->139.80.206.169:3128  1946   91175  1
```

Screenshot

Take a screenshot to record what you see here. Does the data make sense? Well, we have meaningful source and destination addresses which are not unduly influenced by

NAT, so that is good, and the port numbers indicate some proxy traffic, so I'm happy that it is working for now. Time to look at some reporting.

But are the collection statistics accurate? Think about how you could determine this; we'll look at it in the assessment.

6. Basic Reporting

*In this section, we use the **nfdump** product, installed in the previous section, to query the collected data and answer some basic questions about network activity which we generate. This practice is fundamental to understanding what we can do with NetFlow, so don't rush.*

Screenshot

Let's practice querying the recorded data using the **nfdump** tool. All similar tools should be able to produce much of the same, although there will be some large differences in user interface and user-friendliness for various tasks. As you do each item in the list, take a screenshot.

- First some very basic usage. Look at the manual page for **nfdump** to determine what these options mean.

```
nfdump -R /var/cache/nfdump/ -c 5
```

Screenshot

This should print out the first five flows stored in the directory `/var/cache/nfdump/`. Take a screenshot showing the output of that command. Have a look in the directory that is used. Briefly describe how the data is stored; you will need this for the assessment.

Have a look at the summary line. What useful information is listed here?

- Often we are concerned about a particular time slice. Let's repeat the previous example using a particular timeslice; you'll need to determine a suitable example timeslice of which you will have traffic recorded.

```
$ nfdump -R /var/cache/nfdump/ -c 5 -t 2014/12/04-2014/12/06
… first five flows of a two day period …
$ nfdump -R /var/cache/nfdump/ -c 5 -t 2014/12
… first five flows of December 2014 …
$ nfdump -R /var/cache/nfdump/ -c 5 -t 2014/12/04.12-2014/12/04.13
… first five flows of 4th December 2014, 12:00-13:59 …
```

- We can select a different output format that is useful for either casual use, closer inspection, or for use with other programs such as for generating billing data. We can even specify a custom output format with **nfdump**.

```
$ nfdump -R /var/cache/nfdump/ -o long
```

Have a look at the manual page for **nfdump** and play with different arguments to the `-o` option.

- Notice that in the previous example, the rightmost column of the 'extended' output mode shows the number of flows that are aggregated in that line, and that they are all 1.

Aggregation is an important concept when generating reports, as it aggregates the data from individual flows into various groups we are interested in.

Some connections, such as SSH connections, can be long-lived and so may easily be exported from a probe due to time, rather than due to flow completion. For this reason, one very simple form of aggregation is simply to aggregate based on ‘connections’, so individual flows which may have been exported because they spanned a long-time get aggregated together, which is a rather more ‘natural’ way to look at it.

```
$ nfdump -R /var/cache/nfdump/ -a -o long
```

Now you should notice that there are numerous lines that have numerous flows aggregated.

- What are the top ten hosts with regard to bandwidth use? This is another form of aggregation, only based on a particular NetFlow attribute.

```
$ nfdump -R /var/cache/nfdump/ -s srcip/bytes -n 10
```

The size of the report can be parameterised using `-n` for the number of reports, and/or using something like `-L +40M` to limit those results to those over 40 MB.

We can use two queries, one with srcip and the other with dstip, to look at data both incoming and outgoing. You can start to get a feel that we are approaching what looks like an accounting report.

- However, for accounting purposes, there is a lot of stuff that is duplicated. If we want stuff going to/from the internet and our private network, we only want stuff that goes in one interface and out another. Typically, we might use the incoming and outgoing interface ID, but our probe doesn’t supply us with that information.

  This provides us a useful opportunity to look at the filter capability of **nfdump**. This gives us great ad-hoc flexibility. The syntax is similar, but sufficiently different enough to be a little annoying, to the filter syntax used by libpcap (ie. **tcpdump** and friends).

```
… This is our upload report …
$ nfdump -R /var/cache/nfdump/ -s srcip/bytes -L +10M 'src net 192.168.1.0/24'
Byte limit: > 10485760 bytes
Top 10 Src IP Addr ordered by bytes:
Date first seen          Duration  Proto   Src IP Addr  Flows  Packets  Bytes   …
2014-12-03 16:09:07.812 96171.835  any    192.168.1.51  10074   446939  34.6 M  …
2014-12-03 16:21:44.166 104711.381 any    192.168.1.52  6561    57491   16.4 M  …

Summary: total flows: 23357, total bytes: 57.6 M, total packets: 544964,
         avg bps: 4578, avg pps: 5, avg bpp: 110
Time window: 2014-12-03 16:09:07 - 2014-12-04 21:26:55
Total flows processed: 40795, Records skipped: 0, Bytes read: 2125540
Sys: 0.016s flows/second: 2549687.5  Wall: 0.011s flows/second: 3677875.9

… This is our download report …
$ nfdump -R /var/cache/nfdump/ -s dstip/bytes -L +10M 'dst net 192.168.1.0/24'
Byte limit: > 10485760 bytes
Top 10 Dst IP Addr ordered by bytes:
Date first seen          Duration Proto    Dst IP Addr  Flows  Packets    Bytes …
2014-12-03 16:09:07.812 96170.221 any     192.168.1.51   9787   549401  710.0 M …
2014-12-03 16:21:44.166 104711.381 any    192.168.1.52   6420    73105   52.0 M …
2014-12-03 16:19:20.083 104837.615 any    192.168.1.145  2373    24907   19.7 M …

Summary: total flows: 23035, total bytes: 789.8 M, total packets: 662711,
         avg bps: 62818, avg pps: 6, avg bpp: 1249
```

```
Time window: 2014-12-03 16:09:07 - 2014-12-04 21:26:55
Total flows processed: 40795, Records skipped: 0, Bytes read: 2125540
Sys: 0.016s flows/second: 2549687.5  Wall: 0.010s flows/second: 3717084.3
```

Have a think about what you see in this report and in particular the filtering expression used. What might be missed?

7. Advanced Reporting

We don't have time in this laboratory session to work through an install of a anything particularly complicated, so we shall simply look at what one example of a NetFlow analysis and reporting product can do for us.

There are a wealth of available tools for doing reporting and analysis; many are commercial because they help provide greater business intelligence and so is something companies are willing to pay for.

Do some on-line searching (suggested terms: 'netflow white paper') and find a product that does analysis and reporting. Choose a product that is different than what others are covering. What are some of the features this product provides? You will need this for the assessment.

8. Assessment

1. When we installed the collector, you were asked to think about how you would determine if the results were accurate. Briefly describe an experiment you could undertake to evaluate the accuracy of the data.

2. Describe how the data in `/var/cache/nfdump/` is organised. What limits, if any, are in place to prevent the data from growing to fill the disk? If needed, take steps to constrain the amount of disk used; document how this could be done. You will need to do some self-guided research. (Hint: see the manual page for **nfdump** and learn about expiration of old files).

3. Earlier we practiced using **nfdump** for generating upload and download reports. What may be missing in that report. Why is input and output interface preferable?

4. What might happen to accounting when we introduce a proxy cache? What are the issues here? What could be done about this?

5. In the Advanced Reporting section, you were asked to do some on-line searching (suggested terms: 'netflow white paper') and find a product that does analysis and reporting, which is different from a product covered by your peers.

Print the whitepaper. On a single sheet of paper, answer the following, referring freely to the whitepaper or the vendor literature:

1. Which product did you look at?
2. What are the major features of this product? (ie. what makes this product better than others?)
3. What versions of NetFlow does it support? Does it support sFlow ("sampled flow") which is an improvement on NetFlow?
4. What are the suggested uses of this product? (ie. why should a business use this product?)

When you have been marked, pin the whitepaper and your answers on the board in the classroom in the place provided. In this way, we can survey the marketplace and get a feel for the various things that can be done using NetFlow data.

6. Why is it useful to have different graphs of flows, packets and bytes?

How might you recognise a Syn-Ack denial-of-service (DoS) attack by time-series graphs of flows, packets and bytes?