Sources and Applications of Performance and Security-Augmented Flow Data

Avi Freedman, CEO
FloCon 2016
January 2016
Background: NetFlow/IPFIX/sFlow

• NetFlow is:
  – A 20-year old technology now supported in some variant by most network devices, hosts, and sensors.
  – And much smaller than storing all packets, so useful for longer term metadata storage and search.

• sFlow came later, is simpler and more accurate in real-time because it’s just packet sampling.

• IPFIX and Netflow v9 are extensible via templates, and allow sending more than just ‘basic flow’ data via those templates.

• Both via IPFIX/v9 and other formats, there are many sources of app semantics + performance data that work will with flow-like analysis patterns.
‘Classic’
Flow
‘Classic’ Flow

• Classic flow records contain byte and packet counters, TCP Flags, AS, next-hop, and other data aggregated by (usually) the ‘5 tuple’ of (protocol, srcip, dstip, srcport, dstport). VLAN, mac, MPLS, packet size histogram and other data often available.

• Most devices support a fixed sampling rate.

• Despite the relative simplicity of data, there are many use cases for basic flow data for monitoring availability, efficiency, and security of networks, hosts, and applications.
Classic Flow Use Cases

• Classic use cases include:
  – Congestion analysis for providers and/or customers
  – Peering analytics
  – Trending, planning and forecasting
  – (d)DoS detection (primarily volumetric)
  – Basic forensic/historic (who did an IP talk to)
  – Modeling of TE, what-if analysis
  – Customer cost analysis (Flow + BGP communities)
  – Many security use cases for even ‘classic’ Flow:
    • Convolve with threat feeds, DNS, BGP
    • Finding extrusion (or at least indicators), fast flux, botnet c+c, service scanning, long-lived low-bw comms, service compromise, anomalies in many dimensions...
Classic View: Traffic by Source ASN

Bits/s by AS_src

Devices Search

Select All / None
Selected: 1
- cat2_cloudhelix_com
- core_nyc_lsp
- com
- nx1_cloudhelix_com

Filters
Add Group  Clear All

Group 1
- src_as <> 6450

Bits/s by AS_src

cat2_cloudhelix_com

Click to select, Shift+Click to multi-select

<table>
<thead>
<tr>
<th>src_as</th>
<th>Avg Mb/sec</th>
<th>Percent Total</th>
<th>95th Percentile</th>
<th>Max Mb/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>HWINGSS - Highwinds Network Group, Inc.,US (29798)</td>
<td>1,970</td>
<td>58.32</td>
<td>2,324</td>
<td>2,443</td>
</tr>
<tr>
<td>HWNG Eweka Internet Services B.V.,NL (12989)</td>
<td>981</td>
<td>29.02</td>
<td>1,247</td>
<td>1,315</td>
</tr>
<tr>
<td>XSNEWS-AS XS News B.V.,NL (48345)</td>
<td>52</td>
<td>1.53</td>
<td>92</td>
<td>96</td>
</tr>
<tr>
<td>COGENT-174 - Cogent Communications,US (174)</td>
<td>26</td>
<td>0.76</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>MICROSOFT-CORP-MSN-AS-BLOCK - Microsoft Corporation,US (8088)</td>
<td>22</td>
<td>0.65</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>HURRICANE - Hurricane Electric, Inc.,US (6939)</td>
<td>22</td>
<td>0.65</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td>INTERLINK INTERLINK,UA (43586)</td>
<td>20</td>
<td>0.58</td>
<td>73</td>
<td>81</td>
</tr>
</tbody>
</table>
Classic View: Interface -> Interface Traffic

Bits/s by InterfaceTopTalkers

Device Search

Select: All / None

Selected: 1

cat2
cheez
j1
com
com
com
rx1
sup

Filters

Add Group  Clear All

Click to select, Shift+Click to multi-select

<table>
<thead>
<tr>
<th>input_port_all</th>
<th>Avg Mb/sec</th>
<th>Percent Total</th>
<th>95th Percentile</th>
<th>Max Mb/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>GigabitEthernet5/7</td>
<td>56</td>
<td>11.13</td>
<td>62</td>
<td>64</td>
</tr>
<tr>
<td>GigabitEthernet4/17</td>
<td>50</td>
<td>9.92</td>
<td>61</td>
<td>62</td>
</tr>
<tr>
<td>GigabitEthernet1/1</td>
<td>49</td>
<td>9.70</td>
<td>63</td>
<td>72</td>
</tr>
<tr>
<td>GigabitEthernet5/17</td>
<td>45</td>
<td>8.89</td>
<td>63</td>
<td>75</td>
</tr>
</tbody>
</table>

Export  SQL  Add to Dashboard
Classic View: Traffic by top AS_PATHs
<table>
<thead>
<tr>
<th>Key</th>
<th>Alert Name</th>
<th>Criticality</th>
<th>State</th>
<th>Key Type</th>
<th>Output 1 Name:Value</th>
<th>Output 2 Name:Value</th>
<th>Alert ID</th>
<th>Start</th>
<th>End</th>
<th>Time Over Threshold</th>
<th>Recent Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>many_src_ip_s_to_1_dst</td>
<td>ACK_REQ</td>
<td>Major</td>
<td></td>
<td>ipv4_src_dst_addr</td>
<td>pps: 189</td>
<td>src_ips : 3277</td>
<td>3536</td>
<td>2015-08-26</td>
<td>20:25</td>
<td>20:46</td>
<td></td>
</tr>
<tr>
<td>high_fps_per_dst_ip</td>
<td>ACK_REQ</td>
<td>Major</td>
<td></td>
<td>ipv4_src_dst_addr</td>
<td>fps: 110</td>
<td>pps: 118835</td>
<td>3537</td>
<td>2015-08-26</td>
<td>20:25</td>
<td>20:45</td>
<td></td>
</tr>
<tr>
<td>all_dst_53_or_src53_to_1_ip</td>
<td>ACK_REQ</td>
<td>Major</td>
<td></td>
<td>ipv4_src_dst_addr</td>
<td>pps: 51166</td>
<td>mbps : 576</td>
<td>462</td>
<td>2015-08-26</td>
<td>20:25</td>
<td>20:45</td>
<td></td>
</tr>
<tr>
<td>udp_src_cdst0</td>
<td>ACK_REQ</td>
<td>Major</td>
<td></td>
<td>ipv4_src_dst_addr</td>
<td>pps: 86391</td>
<td>mbps : 914</td>
<td>452</td>
<td>2015-08-26</td>
<td>20:25</td>
<td>20:44</td>
<td></td>
</tr>
<tr>
<td>many_src_ip_s_to_1_dst</td>
<td>ACK_REQ</td>
<td>Major</td>
<td></td>
<td>ipv4_src_dst_addr</td>
<td>pps: 137</td>
<td>src_ips : 13517</td>
<td>3536</td>
<td>2015-08-26</td>
<td>20:37</td>
<td>20:47</td>
<td></td>
</tr>
</tbody>
</table>
‘Augmented’ Flow
‘Augmented’ Flow

• ‘Who talked to who’ data is great, but if we can get:
  – Semantics (URL, DNS query, SQL query, …)
  – Application performance info (latency, TTFB, …)
  – Network performance info (RTT, loss, jitter, …)
  from passive observation, it unlocks even more/more interesting use cases!

• With many of the same basic report structures.

• Some of this is already available via IPFIX/V9 from many devices. Or via nprobe and argus for host/sensor. Or as flow-like sources fo data.
Sources of ‘Augmented’ Flow

• Server-side
  – OSS sensor software: yaf, nprobe, argus
  – Commercial sensors: nBox, nPulse, and others
  – Packet Brokers: Ixia and Gigamon (IPFIX, potentially more)
  – IDS (bro) – a superset of most flow fields, + app decode
  – Web servers (nginx, varnish) – web logs + tcp_info for perf
  – Load balancers – advantage of seeing HTTPS-decoded URLs
  – CISCO AVC, Netflow Lite – generally only on small devices

• Common challenge: Some of the exporters don’t support sampling, and many tools can’t keep up with un-sampled flow. And many tools can’t easily map + store augflow fields. (Tradeoff: speed vs flexibility)
augflow Examples: yaf

- [https://tools.netsa.cert.org/yaf/docs.html](https://tools.netsa.cert.org/yaf/docs.html)
- [http://linux.die.net/man/1/yafdpi](http://linux.die.net/man/1/yafdpi)
- ‘Base’ yaf supports entropy, packet size distribution, estimation of TCP setup time (reverseFlowDeltaMilliseconds), app ID, and a few other fields.
- yafdpi can extract and send in IPFIX many varieties of application semantics: FTP, HTTP, IMAP, RTSP, SIP, SMTP, SSH, DNS, IRC, NNTP, POP3, SLP, TFTP, MySQL
- No performance data added
augflow Examples: nprobe

- [http://ntop.org](http://ntop.org)
- template.c in nprobe (and elsewhere)
- Includes both network and with pro plugins, some application performance, and many kinds of application semantics: DHCP, MySQL, RADIUS, DHCP, HTTP, SMTP, SIP, RTSP, GTP
- sysdig perhaps most interesting for APM-like use cases
- Can export in JSON and other formats in addition to IPFIX

```c
{ 0, BOTH_IPV4_IPV6, FLOW TEMPLATE, SHORT SNAPLEN, NTOP ENTERPRISE ID, NTOP BASE ID+110, STATIC FIELD_LEN, 4, numeric_format, dump as uint, "RETRANSMITTED OUT_PKTS", "", "Number of retransmitted TCP flow packets (dst->src)" },
{ 0, BOTH_IPV4_IPV6, FLOW TEMPLATE, SHORT SNAPLEN, NTOP ENTERPRISE ID, NTOP BASE ID+101, STATIC FIELD_LEN, 2, ascii_format, dump as ascii, "SRC IP COUNTRY", "", "Country where the src IP is located" },
{ 0, BOTH_IPV4_IPV6, FLOW TEMPLATE, SHORT SNAPLEN, NTOP ENTERPRISE ID, NTOP BASE ID+86, STATIC FIELD_LEN, 4, numeric_format, dump as uint, "APPL LATENCY_SEC", "", "Application latency (sec)" },
{ 0, BOTH_IPV4_IPV6, FLOW TEMPLATE, SHORT SNAPLEN, NTOP ENTERPRISE ID, NTOP BASE ID+82, STATIC FIELD_LEN, 4, numeric_format, dump as uint, "CLIENT NW_DELAY_SEC", "", "Network latency client <-> nprobe (sec)" },
```
augflow Examples: argus

- Custom format to preserve richness that IPFIX does not allow
- Data can be sent in CSV for use (with augflow) with other tools
- Rich network-layer performance data (jitter, latency sliced many ways), infer topology, ...
augflow Examples: Cisco AVC

docwiki.cisco.com/wiki/AVC-Export:PfR#PfR_NetFlow_Export_CLI

Client: Option Active Performance
Exporter Format: NetFlow Version 9
Template ID: 268
Source ID: 0
Record Size: 61
Template layout

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Offset</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow end</td>
<td>153</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>pfr br ipv4 address</td>
<td>39000</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>reason id</td>
<td>39002</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>counter packets dropped</td>
<td>37000</td>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>transport packets lost counter</td>
<td>37019</td>
<td>20</td>
<td>4</td>
</tr>
<tr>
<td>transport round-trip-time</td>
<td>37016</td>
<td>24</td>
<td>4</td>
</tr>
<tr>
<td>transport rtp jitter mean</td>
<td>37023</td>
<td>28</td>
<td>4</td>
</tr>
<tr>
<td>mos worst 100</td>
<td>42115</td>
<td>32</td>
<td>4</td>
</tr>
<tr>
<td>counter packets dropped permanent short</td>
<td>37001</td>
<td>36</td>
<td>4</td>
</tr>
<tr>
<td>transport packets lost counter permanen</td>
<td>37020</td>
<td>40</td>
<td>4</td>
</tr>
<tr>
<td>long-term round-trip-time</td>
<td>39006</td>
<td>44</td>
<td>4</td>
</tr>
<tr>
<td>flow class wide</td>
<td>95</td>
<td>48</td>
<td>6</td>
</tr>
<tr>
<td>interface output snmp short</td>
<td>14</td>
<td>54</td>
<td>2</td>
</tr>
<tr>
<td>pfr status</td>
<td>39001</td>
<td>56</td>
<td>2</td>
</tr>
<tr>
<td>flow active timeout</td>
<td>36</td>
<td>58</td>
<td>2</td>
</tr>
<tr>
<td>ip protocol</td>
<td>4</td>
<td>60</td>
<td>1</td>
</tr>
</tbody>
</table>
augflow Examples: Citrix AppFlow


tcpRTT
The round trip time, in milliseconds, as measured on the TCP connection. This can be used as a metric to determine the client or server latency on the network.

httpRequestMethod
An 8-bit number indicating the HTTP method used in the transaction. An options template with the number-to-method mapping is sent along with the template.

httpRequestSize
An unsigned 32-bit number indicating the request payload size.

httpRequestURL
The HTTP URL requested by the client.
augflow Examples: nginx, bro

- [https://www.bro.org/sphinx/logs/index.html](https://www.bro.org/sphinx/logs/index.html)
- Not ‘flow’ but can be translated and stored similarly!

**nginx**: log_format combined ' $remote_addr - $remote_user [$time_local] ' "$request" $status $body_bytes_sent ' "$http_referer" "$http_user_agent"' '$tcpinfo_rtt, $tcpinfo_rttvar, $tcpinfo_snd_cwnd, $tcpinfo_rcv_space';

```
# cat conn.log | bro-cut id.orig_h id.orig_p id.resp_h duration
141.142.220.202  5353   224.0.0.251 -
fe80::217:f2ff:fed7:cf65  5353   ff02::fb -
141.142.220.50  5353   224.0.0.251 -
141.142.220.118  43927  141.142.2.2  0.000435
141.142.220.118  37676  141.142.2.2  0.000420
141.142.220.118  40526  141.142.2.2  0.000392
141.142.220.118  32902  141.142.2.2  0.000317
141.142.220.118  59816  141.142.2.2  0.000343
141.142.220.118  59714  141.142.2.2  0.000375
141.142.220.118  58206  141.142.2.2  0.000339
[...]
```
Storing and Accessing Augmented Flow

• Data back-ends need to be able to understand and ingest the extra fields.
• Often requires integration (for OSS/big data tools) or vendor support.
• And if the tools aren’t ‘open’ via API, SQL, or CLI, data can be trapped and not as useful.
• Many first use cases are ad-hoc to prove effectiveness, then drive to UI reports/dashboards.
• Holy grail: semantics + end user app perf + net perf + net flow + host perf + app internals instrumentation.
• Note: Semantics also useful for performance and performance data useful for security!
One Extensible Flow Storage: fastbit

- [https://sdm.lbl.gov/fastbit/](https://sdm.lbl.gov/fastbit/)
- [https://github.com/CESNET/ipfixcol/](https://github.com/CESNET/ipfixcol/)
- [http://www.ntop.org](http://www.ntop.org)

(nprobe CLI)

```bash
fbquery -c
'DST_AS,L4_SRC_PORT,sum(IN_BYTES) as
inb,sum(OUT_BYTES) as outb' \  
-q 'SRC_AS <> 3 AND L4_SRC_PORT <> 80' \  
-g 'DST_AS,L4_SRC_PORT' \  
-o 'inb' \  
-r -L 10 -d .
```
Storing Augmented Flow in Fastbit
Use Case: Network Performance

• If the flow system can aggregate by arbitrary dimensions by AS, AS_PATH, Geo, Prefix, etc...

• Then looking at raw network performance from passive sources can be very useful.

• Ex: TCP rexmit by AS_PATH (i.e. from nprobe for a server or, via span/tap, a sensor).

• Important to weight absolute relevance (not just % loss if a few 3 pkt flows).
SQL -> Fastbit Querying for rexmit

Retransmits > .1\% by ASN at prime-time for ASNs with > 10k pkts:

```sql
SELECT i_start_time, src_AS, dst_AS,
sum(tcp_retransmit) AS f_sum_tcp_retransmit,
sum(out_pkts) AS f_sum_out_pkts,
round((sum(tcp_retransmit)/sum(out_pkts))*1000)/10
AS Perc_retransmits FROM _com WHERE
i_start_time >= '2015-01-09 22:00:00' AND
i_start_time < '2015-01-10 06:00:00' GROUP BY
src_AS, dst_AS, i_start_time HAVING sum(out_pkts) > 10000 AND (sum(tcp_retransmit)/sum(out_pkts))*100 > 0.1 ORDER BY Perc_retransmits DESC;
```
Augmented Flow: rexmit by Dest ASN
Augmented Flow: rexmit by AS_PATH
Use Case: Application-Level Attacks

• With URL and performance data, many kinds of application attacks can be detected.
• To get URL info in an HTTPS world, will need to get data from load balancers or web logs.
• Simplest is WAF – looking for SQL fragments, binary, or other known attack vectors.
• Can hook alerts to mitigation methods, even if running OOB (for example, send TCP FIN/RST in both directions)
Use Case: ‘APM Lite’

• Combining network with application data, you can answer questions like:
  – Show/aggregate cases where application performance is impaired but we know there is no network-layer issue (very useful), and agg by POP, server, app section.
  – Or where there is impairment in both.
  – And ignore network-layer issues where users are unaffected.

• Easy first use case: API perf debugging for web page assembly, or debugging CDN origin pull.
Use Case: Bot detection

• With performance information combined with URL, basic e-commerce bot detection is possible.
• Many attacks are advanced so may require a packet approach to get complete visibility, but basic visibility can often demonstrate a problem.
• Can sometimes be done with syslog analytics, but flow tools often aggregate in interesting ways (geo, AS) that syslog analytics don’t, at least out of the box.
Modern ‘Flow’ Format: kflow

• At today’s speeds, async(ly)-templated formats may not be the most efficient (space/CPU) implementation.

• See also: http://www.ntop.org/nprobe/yes-there-is-life-after-netflow/ (but Kentik prefers binary to JSON/Kafka)

• Working on an open-spec format called kflow with open source tools to take to and from NetFlow, sFlow, IPFIX, nginx and bro logs, and Cisco, Citrix, ntop, and other vendor formats.

• Based on Cap’n Proto, which is a ‘serialization’ lib that is basically a struct with 0-packing - https://capnproto.org/

• Drawback: Can’t delete fields, just 0-pack them.

• Will shortly be live at https://github.com/Kentik
Flow with Cap’n Proto kflow v1

```c
struct kflow_v1 {
    version @44: Int64;
    timestampNano @0: Int64;
    dstAs @1: UInt32;
    dstGeo @2: UInt32;
    dstMac @3: UInt32;
    headerLen @4: UInt32;
    inBytes @5: UInt64;
    inPkts @6: UInt64;
    inputPort @7: UInt32;
    ipSize @8: UInt32;
    ipv4DstAddr @9: UInt32;
    ipv4SrcAddr @10: UInt32;
    tcpRetransmit @27: UInt32;
    dstBgpAsPath @34: Text;
    dstBgpCommunity @35: Text;
    ...
};
```
kflow v2

- Cap’n Proto is fast and compact.
- But to support many extended fields (imagine every bro and argus field) could still become unwieldly.
- And can’t easily delete fields/version.
- Don’t love templating, but ading OOB vs inband+async templates. Generated .H from server-side tools:

```c
static short KFLOW_CUSTOM_COLUMNS[] = {KFLOW_CUSTOM_FOO_FID,
                                        KFLOW_CUSTOM_MYINT_FID,
                                        KFLOW_CUSTOM_MYFLOAT_FID};
static short KFLOW_CUSTOM_COLUMN_TYPES[] = {KFLOW_CUSTOM_FOO_TYPE,
                                            KFLOW_CUSTOM_MYINT_TYPE,
                                            KFLOW_CUSTOM_MYFLOAT_TYPE};

(*kfrec)->dst_as = 10;
(*kfrec)->src_as = num;
(*kfrec)->sample_rate = 1028;
(*kfrec)->src_bgp_as_path = strdup("100 200 300");
(*kfrec)->custom[KFLOW_CUSTOM_MYINT_OFF].val.i_val = 128;
(*kfrec)->custom[KFLOW_CUSTOM_MYFLOAT_OFF].val.f_val = 64.5;
(*kfrec)->custom[KFLOW_CUSTOM_FOO_OFF].val.s_val = strdup("FOO");
```
Summary/Takeaways

• Many sources of ‘augmented flow’
• Even web and bro/snort/suricata logs
• Finding a tradeoff between flexibility and speed in storage can be a challenge
• But with unified augflow data, the same flow forensics repositories can do triple or more duty with operational, performance, and additional security analytics
Comments / Questions?

Avi Freedman
avi (at) kentik.com