June 22, 2020 < https://team-cymru.com/blog/2020/06/22/quick-wins-with-network-flow-analysis/>
Dave Munson < https://team-cymru.com/blog/author/dmunson/>

Quick Wins with Network Flow Analysis

While this article focuses on the use of Team Cymru's Pure Signal™ platform — the Augury™ solution — readers will gain some great guidance on how to use flows in their analysis in general.

The Augury dataset comprises network flows records that are downloadable as CSV. Compared to the direct utility of some other Augury datasets, flows might seem fiddly, but there is often useful information lurking in flows data. This article provides some tips, ideas and incentives for looking at flows data in more detail.

Basic familiarly with network flows records, TCP/IP and CSV is assumed.

Refresher

Each flow record will generally refer to multiple packets aggregated by matching source IP address, destination IP address, source port, destination port and the capture interfaces. The directionality of the flow, which is the directionality of the constituent packets, is one-way, from source IP address to destination IP

address (and these can be IPv4 or IPv6 addresses).

An example scenario is shown below, where multiple TCP segments, corresponding to the loading of your favorite web page, are aggregated to a single flow record. A TCP connection (or *socket*), uniquely distinguished by the tuple (*src_ip_addr,src_port,dst_ip_addr,dst_port*), is usually bidirectional. But a network flows record counts constituent segments (or packets, or datagrams…) from only one direction. In the diagram below, TCP segments sent from the laptop to the web server would appear in different flow records, where the source IP address would be that of the laptop.

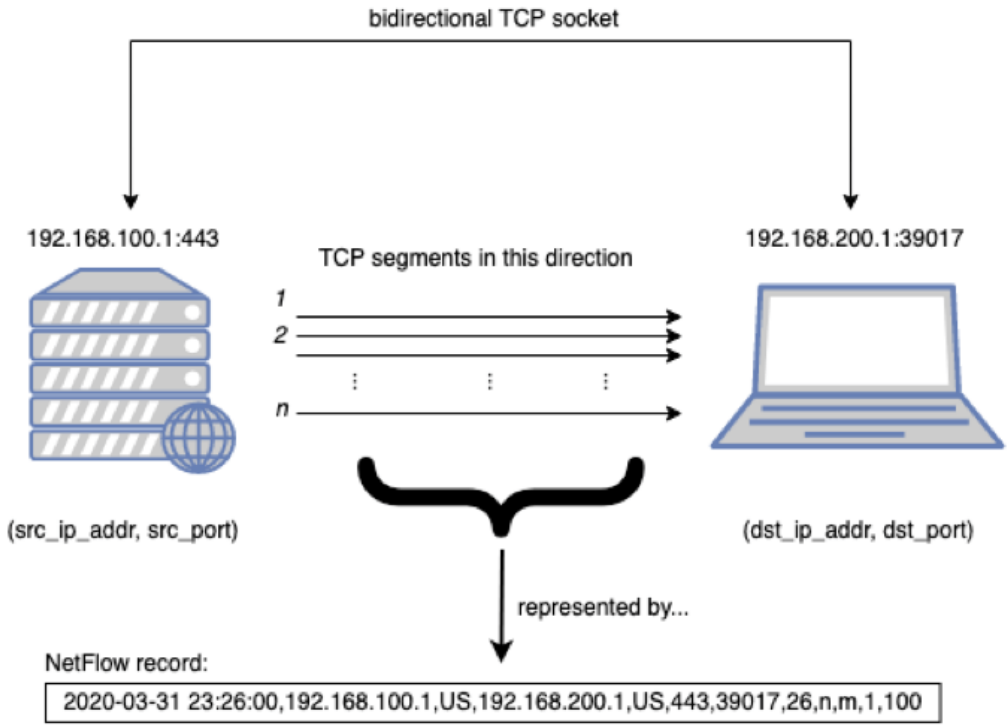


Figure 1 – Illustrating creation of a network flows record

A reminder of the CSV format used by flows that can be exported from Augury is shown in the table below.

Column Headers	Meaning	Notes
Column Headers	Meaning	Notes
	Start time that the flow event was	

start_time	observed, i.e. timestamp of the first packet of the flow.	in the UTC timezone.
src_ip_addr	Source IP address.	The **originator** of this particular packet.
src_cc	Source IP address geolocation.	
dst_ip_addr	Destination IP address.	The **destination** of this particular packet.
dst_cc	Destination IP address geolocation.	
proto	IP protocol number as defined by IANA	Used to distinguish TCP (6), UDP (17), ICMP (1) and other types.
src_port	Source port for TCP and UDP packets.	
dst_port	Destination port for TCP and UDP packets.	Also decodes as type and code for ICMP packets.
tcp_flags	For TCP this is a decimal representation of the union of all TCP flags observed during the flow. Not present for UDP. For ICMP, encodes type and code.	See …
num_pkts	Number of protocol-specific packets in this flow	
num_octets	Number of octets in this flow.	
Column Headers **sample_algo**	**Meaning** Refers to the sampling in use for this	**Notes** E.g. set to '1' no deterministic

sample_interval	Refers to the sampling in use for this flow.	deterministic round-robin. E.g. 1 in every 100 packets.

Table 1 – Explanation of the fields used in network flows exported from Augury

Caveats

Some of the features of flows as a data source that are likely to have bearing on analytical inferences are covered below.

- **Coverage** (capture interfaces) is not uniform across the world. This can manifest as skew affecting aggregations involving geolocation, observed autonomous systems, and numbers of records. Caution is necessary when comparing the results for different IP addresses that may be captured at different points.
- Generally, not every packet observed is recorded, otherwise known as **sampling**. For a one-off event, this means it may not be found even though it occurred. For regular events and more general behaviors these are much more likely to be observed.
- Given the vast amount of data collected, **retention** is currently around 30 days. Grab your network flows as soon as you realize you have a requirement for it.
- Configurations necessary at some locations might result in **anomalies** such as zero flags, and the appearance of loopback and RFC1918 addresses.
- Network flows indicates **directionality** of a flow by specifying source and destination and in most cases does not indicate the initiator of a connection.

To illustrate the latter point, in the scenario depictured in Figure 1, we assume, based on port allocations, that the laptop initiated the TCP connection to service running on TCP/443 of the server. However, the flow record does not preclude the possibility that the laptop was running a service on TCP/39017 and that the other host initiated a connection to the laptop. In reality many such situations

can be resolved simply through multiple observations – in this example, it would be as unlikely to see hundreds of home machines running a service on TCP/39017 as it would be to see a server continually use TCP/443 as a source port!

Tips

The tips below are provided in the context of searches that have been performed for a single IP address of interest that has been allocated to its own CSV file. It is of course possible to search Augury flows for multiple IP addresses in one search, and also using other attributes such as a particular port, and for these cases you would need to adapt some of the methods below accordingly.

No code samples are provided to prevent the size of this article becoming unwieldy. Python, Perl, Ruby and many other popular languages have specific CSV processing libraries, that make short work of processing flows records. It is also easy to use the Bash Shell to perform quick and dirty aggregations using the standard tools (such as cut and awk), or with specific downloadable tools such as **csvkit < https://github.com/wireservice/csvkit>** or **textql < https://github.com/dinedal/textql>** .

<Period of coverage, number of records, protocols used

A table similar to that shown below can help you understand what your coverage is and how to see differences in coverage before making comparisons between different IP addresses.

IP Address	First Seen	Last Seen	# Days Active	# Records
ip_addr_1	2020-01-22 04:22:01	2020-02-05 03:00:00	8	1200
ip_addr_2	2020-03-22 03:11:11	2020-03-31 20:42:30	5	200

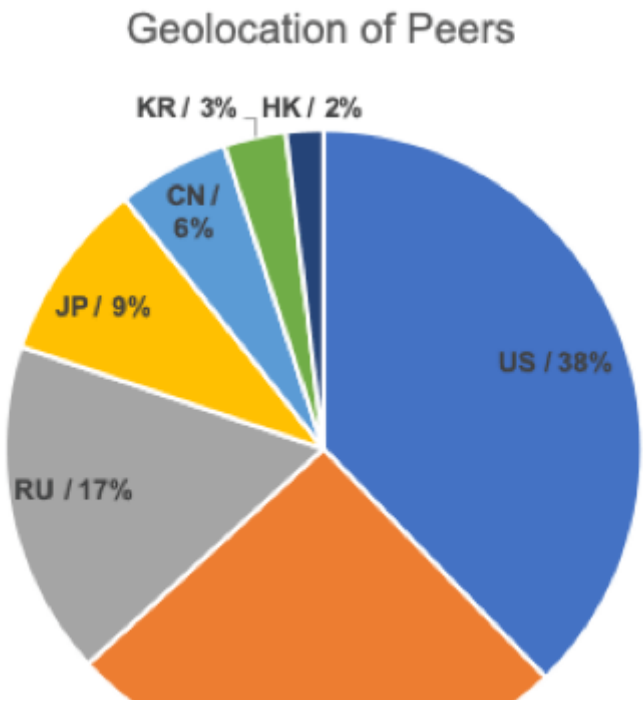
It will be of interest in some cases to note whether there is communication between different IP addresses of interest. For example, C2 servers associated with certain kinds of malware are known to contact other C2 servers and this could be the starting point of a methodology to identify additional C2s. This is discussed further below.

Getting an idea of protocols in a good first step in understanding the traffic profile of the IP address in question, which can be drilled-down on later. What % of records are TCP, UDP and ICMP? Are tunneling protocols such as ESP in use?

<Geolocation

Extracting and aggregating the country fields from the CSV could provide insight regarding any preference for the IP address of interest to communicate with peers in particular geographic regions. Patterns might be expected to emerge, for example, in the case of C2 servers where a malware campaign has targeted particular geographic or language-speaking regions.

As geolocation data is provided within the CSV file, it is easy to pick this out and count it (– extract and join the third and fifth columns, remove the IP address of interest then count each country code) and then to chart it.



<Peers

Creating a table of peers for each IP address of interest is a good starting point to understanding its traffic profile and for comparing the communication of different IP addresses of interest.

It's easy to do this quickly by cutting out the second and fourth columns and removing the IP address of interest itself. Two indicators can then be quickly produced (without regard to the TCP or UDP ports):

- the number of unique peers, and,
- how frequently peers repeat.

At this point the list of unique peers can also be piped through other tools and searches that you may use internally. For example, a tool to perform bulk Whois lookups and extract details such as netblock name and organization. Whois details can often distinguish peers associated with broadband connections versus those associated with hosting and VPS.

The table below, containing fictional data, shows peers for the IP address of interest ***ip_addr_1*** and shows where three peers, responsible for the bulk of traffic to the IP address of interest, were found to be at the same hosting company.

IP Address	Peer	Whois Org	# Records
IP add 1	10.200.12.100	Bob's Hosting, US	400
	10.200.19.120	Bob's Hosting, US	350
	192.18.55.99	Bob's Hosting, US	312
	192.168.5.4	Alice Broadband LLC	10
	192.168.80.55	Fast ADSL GmbH, DE	7
IP Address	Peer	Whois Org	# Records

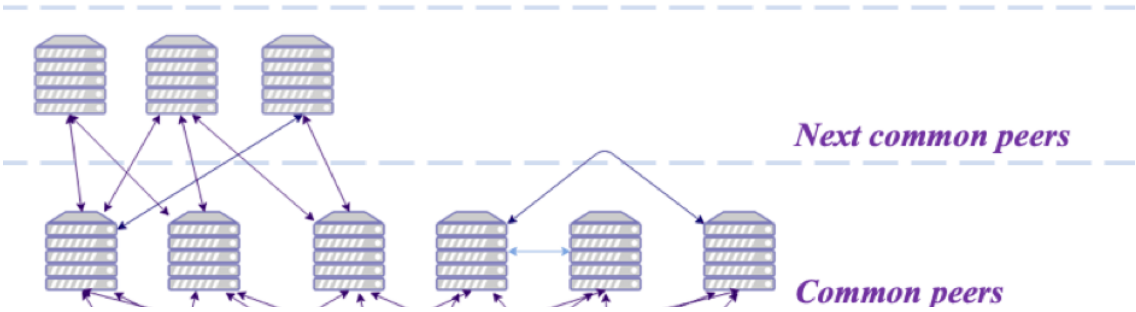
192.168.27.4	A.N. Other ISP	7
10.7.3.8	..	5
172.20.55.2	..	4
10.27.44.2	..	3
172.17.17.9	..	2

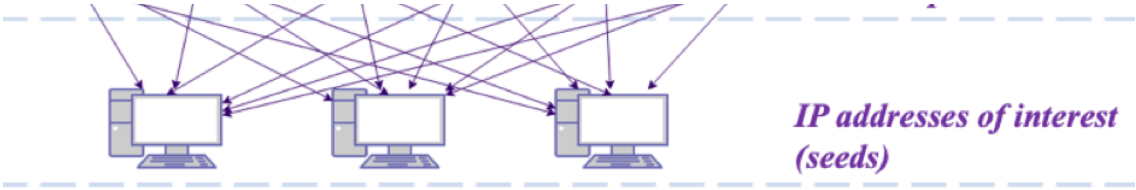
Another common scenario is to attempt to further characterize the peers, or a subset of them, by performing a series of additional Augury searches. At the risk of sidelining into an Augury tutorial, common examples include:

- PDNS name-to-address searches, e.g. for a sample of the peers contacted on TCP/80 and TCP/443 to search for possible commonalities in sites visited.
- PDNS DNS query searches, e.g. to see whether, and, what other entities tried resolving the same hostnames.
- X.509 searches to identify whether certificate information provides CN, contact details and so on.
- Open ports data to build up a picture of the services provide by the peer.

Going back to our list of peers, is now also possible to determine the intersect, i.e. which were common peers across two or more IP addresses of interest?

When the IP addresses of interest are infected clients, common peers may include C2 servers. In turn, common peers of the identified C2 servers may help identify another hierarchy of actor infrastructure. This scenario is illustrated below with some peer-to-peer communication also observed.



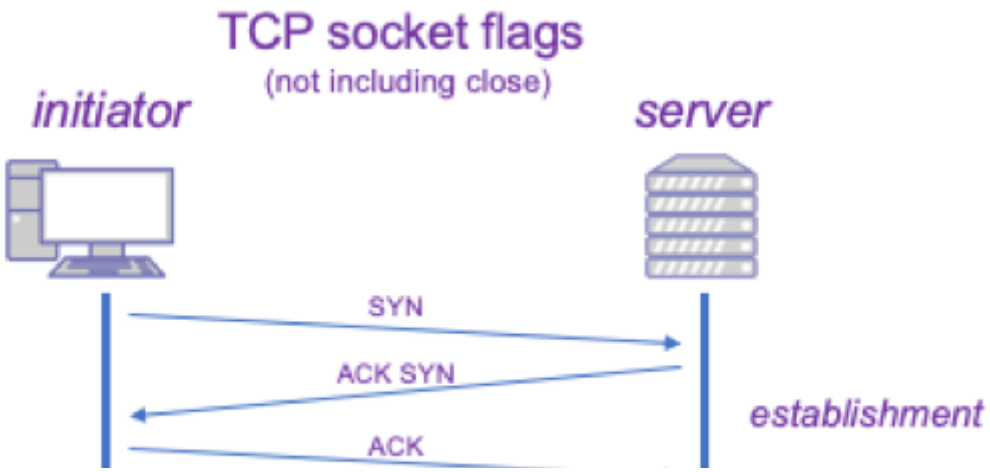


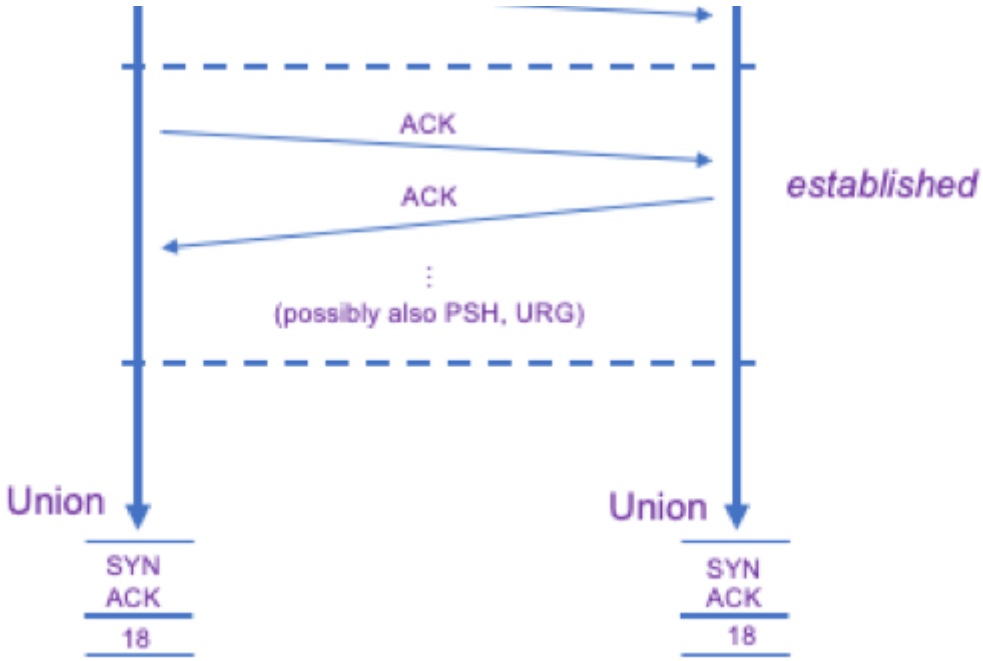
<Passive service profile and connection activity

Whilst Augury already has a dataset that can indicate open ports for a particular host, it is sometimes useful to see open ports at the exact time of interest in network flows.

For most conventionally configured operating-systems, the TCP/IP stack will tend to allocate high numbered ports as the source port of the initiator (for example, many Linux kernels use port 32768 to 61000, so-called ephemeral ports). Ports in the range 1 – 1023 are usually reserved for well-known system services, and there are numerous four-digit ports frequently used such as 3128 (web proxies), 6667 (IRC) and 8080 (web proxies and servers). It is more unusual to see services running on 5-digit TCP ports, unless dynamically allocated with something like RPC. Threat actors may allocate service ports high in the port range to help disguise them amongst ephemeral port traffic.

For TCP flows, a socket was established if the union of flags contains more than SYN and RST (represented by decimal 6). Since the union of flags will contain SYN and ACK on both sides if the flow contains connection establishment, and probably mainly ACKs on both sides during the established phase, flags aren't a great help in determining initiator and responder (see diagram below).





To try and infer the service port we can use:

- the convention on port numbering (as discussed above);
- multiple observations (as discussed above), and, possibly,
- bytes transferred (if the suspected service is associated with significant asymmetric data transfer).

With the methods above, it is possible to have a script scan a file of network flows, noting and counting sockets. This forms the basis of the passive service profile. Example output from the script I use is shown below.

```
1912 records during 2020-02-08 09:21:47 - 2020-03-18 08:29:41
385 peers: 1765 lines of TCP, 9 lines of UDP, 137 other

TCP services:
        Port,Count
        222,158
        1433,54
          …

TCP outbound ports:
        Port,Count
        443,1059
        80,197
          …

UDP services:
        Port,Count
        6881,5

UDP outbound ports:
        Port,Count
        53,1
```

This indicates that the IP address of interest was likely running two services and that there were multiple sockets associated with those services. TCP/222 is not a well-known port but may be an alternative port for SSH (usually on TCP/22). The host also has TCP/1433 open, the well-known port for Microsoft SQL server. We might accrue this observation towards inferring that the host might be running a Windows OS.

We can also see that likely outbound activity is largely web browsing (TCP/443 and TCP/80). Perhaps this is a personal machine. If the figures were very high one might guess that the host is a proxy or endpoint for NAT.

More detailed data can be output as needed, for example, a breakdown of connections and clients associated with services:

TCP services:

192.168.0.1:222

– 158 connections from [“10.70.60.162”, “10.80.60.243”,

“10.10.150.50”, “10.20.30.40”]

…

Possible UDP services is limited to an unknown service running on UDP/6881. Searches suggest that this might be related to online gaming or BitTorrent. Outbound UDP activity is limited to UDP/53 – likely DNS lookups.

Looking further at passive service profiles can be useful for identifying behavior consistent with proxies and scanners.

<Elaborating on sessions of interest

Some protocols are of special interest, for example, SSH sessions, both initiated and serviced[1]. Since SSH scanning across the internet is widespread, we will want to try and discount as many scanning sessions as possible. For inbound SSH (i.e. most likely associated with connections to TCP/22), I’ve found a useful way to analyze the traffic is to first isolate TCP/22 inbound into probably client and server as described above, then to cut this data into the following views:

By Source Port	**Peers where the source port was constant across multiple records (and data was transferred – so more than just SYNs). Recall that this should indicate a single socket. This method removes SYN scanning and brute-forcers that make a connection per credential guess.**
# Records	**Total number of records might indicate how popular (or abused) the service is.**
# Peers	**Peers making multiple connections in short periods of time with different source ports are probably not genuine administration sessions. Are there more peers from particular countries or AS? Are peers from residential or server-based netblocks?**
# Bytes	**Bytes transferred with TCP/22 by each peer. Bear in mind that brute-forcing of credentials can result in significant volumes of traffic.**
# Days	**How many of the peers repeat over multiple days?**

[1] Note that Augury also contains separate “SSH connections” and “SSH info” datasets.

<Simple timeline

For an IP address of interest, parsing the date column into a scale of interest forms the basis of a timeline. In many cases discrete days are a useful scale to look at activity over a month or so, but other divisions such as hourly may have application for your analysis. Sticking with the daily example, onto this scale it is possible to plot number of records per day, number of peers per day, probable outbound-initiated and probable inbound (service) traffic.

Output by records for a recent analysis is shown below, indicating that activity for this particular host took off from double-digit figures prior to 22 March to four-figure afterwards. This was useful in identifying when the host became actively used in a campaign.

used in a campaign.

Output on this kind of activity can obviously identify key dates of interest, but may also drive further analysis, such as looking at the composition of the peers before and after the jump.

← Network Perimeters in the Age of Social Distancing < https://team-cymru.com/blog/2020/06/17/network-perimeters-in-the-age-of-social-distancing/>

→ Apply to Attend The Underground Economy (UE) Conference 2020 < https://team-cymru.com/blog/2020/07/08/apply-to-attend-the-underground-economy-ue-conference-2020/>