Sets, Bags, and Rock and Roll*

Analyzing Large Data Sets of Network Data

John McHugh

CyLab and CERT Network Situational Awareness Center,
Carnegie Mellon University, Pittsburgh, PA 15313, USA
jmchugh@cert.org

Abstract. As network traffic increases, the problems associated with monitoring and analyzing the traffic on high speed networks become increasingly difficult. In this paper, we introduce a new conceptual framework based on sets of IP addresses, for coming to grips with this problem. The analytical techniques are described and illustrated with examples drawn from a dataset collected from a large operational network.

1 Introduction

It is not unusual for relatively modest networks today to exhibit trans border flows on the order of megabits per second. Monitoring even a small network with a few hundred hosts can generate many gigabytes of TCPDUMP data per day. Capturing only headers can reduce the volume somewhat, and more compact formats based on abstractions such as Cisco's NetFlow can reduce the volume further. Even so, the volume of data collected is sufficient to overwhelm many analysis tools and techniques. In general, the problem is one of grouping and classifying the data in such a way that uninteresting phenomena can be pushed aside, allowing the investigator to extract and further scrutinize data that is of interest. Recently, the CERT Network Situational Awareness Center has been involved in the analysis of large sets of NetFlow data. To support this effort, they have developed a set of tools, collectively known as the SiLKtools[1]. In the remainder of the paper, we begin by sketching our thesis and analysis

* The mantra "Sex, Drugs, and Rock and Roll" enjoyed currency in the 1960s. To the ears of an older generation, Rock and Roll was just a particularly unpleasant form of noise. Since the general theme of this paper is separating signal from noise in network data, the title is not too strained. This material is based upon work partially supported by the National Science Foundation under Grant No. 0326472. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This work is also supported by the Army Research Office through grant number DAAD19-02-1-0389 ("Perpetually Available and Secure Information Systems") to CyLab at Carnegie Mellon University

[1] The SLK are the initials of the late Suresh Konda who was instrumental in the initial development of the tool set

P. Samarati et al. (Eds.): ESORICS 2004, LNCS 3193, pp. 407–422, 2004.

approach. We then digress to describe the NetFlow data collected, noting that the analysis can be applied equally well to TCPDUMP or other data forms with a bit of preprocessing. The basic functionality of the SiLKtools suite and some of the extensions made in support of our analysis efforts are then described. The remainder of the paper will present examples of the analyses that we can perform using the tools supplemented by relatively simple programs to characterize and organize the reduced data. The paper concludes with a discussion of our plans for further extensions of the tools and additional analyses.

2 The Thesis and Approach

Sets and set theory are abstractions that facilitate reasoning about many classes of problems. We have been exploring the use of sets to provide a compact way of describing and reasoning about the Internet and about traffic observed at various points on it. For example, it is useful to consider such things as the set of hosts on a given network that are active during a given time interval. It might also be useful to consider the set of external hosts that are observed performing a questionable activity such as scanning during such an interval. Similarly, one might want to identify the set of users of some service provided by the local network to the outside world (e.g. web services) during the interval. In the case of the first set, the set of active machines, we could attempt to obtain the answer by asking the system administrators, by examining the state of the DHCP server responsible for leasing IP addresses to the network, by consulting the responsible DNS server, or we could approximate the result by observing those hosts within the network that either originate traffic or respond to connection requests. If we can observe the traffic passing in and out of the network at some transition point such as a border router, the observations may constitute a reasonable approximation of the active set[2] of hosts. The set of active hosts, can be used to partition incoming traffic into that addressed to active hosts (hits) and that which is not (misses). Arguably, the later partition consists of a mix of malicious traffic, indirect evidence of malicious traffic, and, possibly, some amount of benign, but misdirected traffic. This partition can be further processed to identify interesting classes of senders. For example, originators attempting TCP connections will send packets containing a SYN flag. If we select flows containing SYN flags and count the number of flows per source address using a "bag[3]", we can sort the bag and identify high volume scanners. It is not uncommon see a single host performing scans of an entire /16 network in the course of a relatively few minutes. Having identified such a scanner, it is trivial

[2] We assume that all traffic in and out of the monitored network passes through an observation point. Multiple border crossing points are possible. For the moment, we assume that hosts within the monitored network do not spoof addresses and that multiple responders such as honeypots are not deployed within the monitored network

[3] A bag is a counted set or multiset in which the number of occurrences of each member of the basis set is recorded

to find the flows from the scanner to the active or hit partition and create the set of active machines included in the flow. At that point, it is useful to determine if any of the targets responded to the scanner, and, if so to examine the traffic between the scanner and the target (and the subsequent behavior of the target) to determine if the target has changed its behavior in ways that might indicate that it has been compromised.

As can be seen from the example of the previous paragraph, the use of sets and bags, combined with simple filtering based on properties of the data records themselves allows the clustering of data with some particular security (or other) properties in common. Since we are dealing with many thousands of flows per minute on large networks, the constructions of sets and bags allows us to abstract from individual behaviors to clusters of activities. As the paper develops, we will elaborate on this thesis and develop the tools and techniques that we need in more detail, however, we have a number of utilities available including:

- An efficient set representation that allows us to represent IPv4 address sets directly in memory. There is also a compact disk representation that can be read and written efficiently
- An extension of the set representation, a bag, that allows a 32bit counter to be associated with each IP address. It too has an efficient disk representation.
- Routines that allow set unions and intersections to be computed, producing additional set files.
- Routines that allow sets and bags to be created from files containing network flow data.
- Routines that allow sets and bags to be created from ASCII lists of IP addresses in both "dotted" form (possibly containing wild cards), in CIDR block form, and in unsigned integer form.
- Routines to list the contents of sets and bags at various levels of detail, including the network structure (subnet relationships) of a set.

These are sufficient for our initial analysis, though we plan to add other programs to the suite as the need for them becomes clear.

3 NetFlow and Other Data Sources

NetFlow was developed by Cisco as a mechanism for gathering traffic statistics to support billing and network management. NetFlow operates on routers and switches to report traffic statistics on a per interface basis. Although it is not standardized, it is supported in more or less compatible ways by a number of other router and switch manufacturers. According to Cisco[4], the detailed traffic statistics collected by NetFlow include:

- Source and destination IP addresses
- Next hop address

[4] `http://www.cisco.com/en/US/products/sw/netmgtsw/ps1964/products_user_guide_chapter09186a00801ed569.html`

- Input and output interface numbers
- Number of packets in the flow
- Total bytes (octets) in the flow
- First and last time stamps of packets that were switched as part of this flow
- Source and destination port numbers
- Protocol (and flags as appropriate)
- Type of service (ToS)
- Source and destination autonomous system (AS) numbers, either origin or peer (present in V5 and select V8 datagrams)
- Source and destination prefix mask bits (present in V5, V7, and V8 datagrams)

Note that NetFlow records include one or more packets, and represent unidirectional flows. As such, NetFlow lies somewhere in between TCPDUMP records which contain data about individual packets and connection records which would abstract an entire TCP session to a single record. Because NetFlow is resource intensive, there is a limit to the number of open flow records that the router can maintain at one time. New records are created whenever a new flow is seen. A flow is deemed to be new if it contains a (source/destination/protocol[5]) tuple that is not currently being monitored. A flow is closed if it has been inactive for a prescribed period of time(typically some seconds), if it has been explicitly closed (TCP FIN or RST), or if it has been open and active for a prescribed period of time (typically some minutes). Note that this has the effect of breaking up long, steady TCP sessions as well as intermittent TCP sessions with long pauses. It also creates pseudo sessions from sequences of UDP packets as might be associated with streaming media.

The individual flow records are aggregated and sent in batches encapsulated in a UDP packet to a central collection point for processing. In our case, the processing point stores the flow records in a compact format that can be sequentially searched and extracted based on times and a number of match criteria as discussed in the next section. Since our tools operate from this format, it is worth considering whether other forms of data might be stored in the same format and processed with the tools. The answer is a qualified yes. It would be trivial to extract most of the required data from packet based sources such as TCPDUMP or Argus records. Since packet data is typically captured on a link, router specific information such as interfaces, AS numbers, and next hop addresses are not available, but these seldom appear in our analysis. If we were to aggregate data from a number of collection points, these fields could be used to indicate the collection point and directionality of the packet.

We are currently investigating two approaches for obtaining flow data from packet data. There is a `libpcap` based flow accumulation program, `fprobe`[6] that will observe a link and create NetFlow records that can be sent to a collector. In addition one of our customers is building a high performance collector based

[5] In the case of TCP and UDP, ports are included

[6] Available from `http://sourceforge.net/projects/fprobe`

on a commercial TCPDump hardware capture board[7]. Both of these approaches would allow us additional flexibility in consolidating flows and would allow minor enhancements to the NetFlow format, e.g. recording whether the opening packet of a tcp flow was a SYN, SYN/ACK, or something else.

As far as we can determine, the first attempt to develop tools for security analysis from NetFlow data occurred at Ohio State University. These tools[1] are fairly special purpose and primitive compared to our approach.

4 The SiLKtools Suite and Its Extensions

According to the SiLK website[8]:

> SiLK, the System for Internet-Level Knowledge, is a collection of NetFlow tools developed by the CERT/AC to facilitate security analysis in large networks. SiLK consists of a suite of tools which collect and examine NetFlow data, allowing analysts to rapidly query large sets of data. SiLK was explicitly designed with a trade off in mind: while traffic summaries do not provide packet-by-packet (in particular, payload) information, they are also considerably more compact and consequently can be used to acquire a wider view of network traffic problems.
>
> SiLK consists of two sets of tools: a packing system[9] and analysis suite[10]. The packing system receives NetFlow V5 PDU's and converts them into a more space efficient format, recording the packed records into service-specific binary flat files. The analysis suite consists of tools which can read these flat files and then perform various query operations, ranging from per-record filtering to statistical analysis of groups of records. The analysis tools inter operate using pipes, allowing a user to develop a relatively sophisticated query from a simple beginning.
>
> The vast majority of the current code-base is implemented in C, Perl, or Python. This code has been tested on Linux, Solaris, Free/OpenBSD, AIX and Mac OS X, but should be usable with little or no change on other Unix platforms.
>
> The SiLK software components are released under the GPL.
>
> The project is the fruits of work done at the CERT Coordination Center (CERT/CC) that is part of the Software Engineering Institute at Carnegie Mellon University.

[7] See `http://www.endace.com/networkMCards.htm` for additional information

[8] `http://silktools.sourceforge.net/`

[9] The SiLK Packing System is a server application that receives NetFlow V5 PDU's and converts them into a more space efficient format, recording the packed records into service-specific binary flat files. Files are organized in a time-based directory hierarchy with files cover an hour at the leaves

[10] The SiLK Analysis Suite is a collection of command-line tools for querying packed NetFlow data. The most important tool is **rwfilter**, an application for querying the central NetFlow data repository for NetFlow records that satisfy a set of filtering options

The analysis suite includes a number of applications and utility programs. We discuss in some detail only those that are used in the examples below, however, manual pages for the entire suite are available from the web site. For convenience, we refer to the packed data files used by some of the programs as "rwdata" files. In most cases, input can come from stdin or from a rwdata file and it is possible to associate an output of most programs with stdout, allowing chains of programs.

5 Examples and Sample Analyses

In this section, we illustrate our analysis techniques with two examples. One is a brief data sample from a large cross section of networks that have been aggregated together. the other represents a detailed view of a weeks activity on a /16. In all cases, no real IPs are contained in the analyses. We note that the analyses presented are exploratory in nature. At the present time, we are just beginning to come to grips with the issues of analyzing and understanding network traffic on this scale. Every time we look at a new sample of the data, we find previously unexpected behaviors. Our customer's analysts use scripts to invoke the tools to perform routine inspections for a variety of malicious activities. Our goal is to attempt to understand what we see in the hopes that it will lead to improved detection in the long run. For now, we are not attempting to produce turnkey procedures for system administrators and analysts but rather to aid networking experts in understanding their domain. The examples are offered as an enquiry, much in the spirit of Tukey[2]. Programs from the suite used in producing the examples are printed in **bold**. Details of these programs appear in Appendix A.

5.1 A Brief Examination of a Large Sample

Using the **rwfilter** program to extract data from the incoming (Internet to customer network) and outgoing (customer network to Internet) archives, we obtained files of data from a cross section of monitored networks for a small interval of time. Outgoing data was monitored for an interval began slightly before interval used for incoming data and extended slightly beyond it. This insures that the internal hosts that responded to traffic from the outside are included in the sample even in the incoming and outgoing clocks are not perfectly synchronized[11] or internal hosts respond only after a small delay. The set of active hosts in the internal network is approximated by by passing the outgoing data file to the **rwsuperset** program asking it to create a set file for the source addresses contained in it's input. The incoming data file is then passed to **rwfilter** along with the active host set file to partition the incoming file into hit and miss files, based on the destination address of the incoming flows.

About 2/3 (65.4%) of the flow records are directed at inactive (presumed nonexistent) targets, the remaining 1/3 (34.6%) are directed at active hosts.

[11] The data is aggregated from a number of routers. In some cases, separate routers handle incoming and outgoing traffic

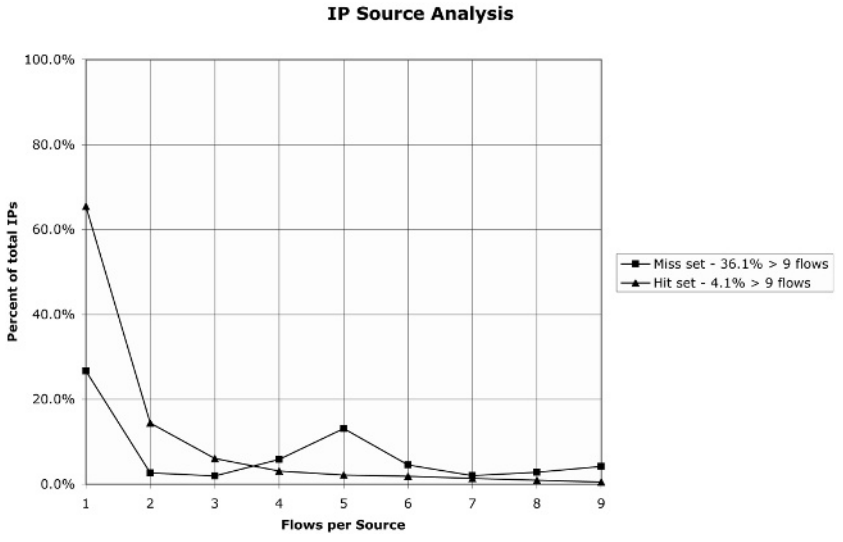

Fig. 1. Reduction in incoming IP Source Set Sizes as a Function of Number of Flows

Further partitioning by protocol, again using **rwfilter** simplifies subsequent analysis. As expected, the vast majority of the data is TCP. We further analyze the TCP data using **rwsuperset** to create bags for source and destination addresses for the hit and miss files. The fall off in set size is illustrated in Figures 1 and 2. Note that hit and miss set sizes follow similar, but distinct patterns. The big differences are between source and destination behaviors. About 36% of the source IPs in the miss partition generate ten or more flows while only 4% of those in the hit partition do so.

Using **readbag**, along with the system utilities `sort` and `head` we can identify the most commonly occurring addresses in the miss partition. Figure 3 shows some of the results of this analysis. A closer look at the top address is interesting. Using **rwfilter** to extract flows with this source address in the miss file extracts some 400K of records. The destination set for this contains 12994 hosts, all from XXX.YYY.0.0/16 The hit set also contains entries from this network, 7 in all. Using **rwcut** to extract the destination port, flag, and packet and byte count fields from the extracted files and clustering, using **cluster.sno** to determine and count the unique field combinations used, we find that all the flows sent to these addresses are 48 byte SYN packets addressed to port 4899 (listed as a "radmin" port by IANA, with other possible usages reported as ChiliASP and iMesh). An inspection of the outgoing traffic from this network indicates no responses to the connection attempts.

The second entry is somewhat different. The traffic from this address scans a different /16. looking for responses on port 7100[12] (X Font service according

[12] `http://www.cert.org/advisories/CA-2002-34.html` describes a vulnerability in the X font service on Solaris. It is likely that the scanner was looking for machines that could be attacked

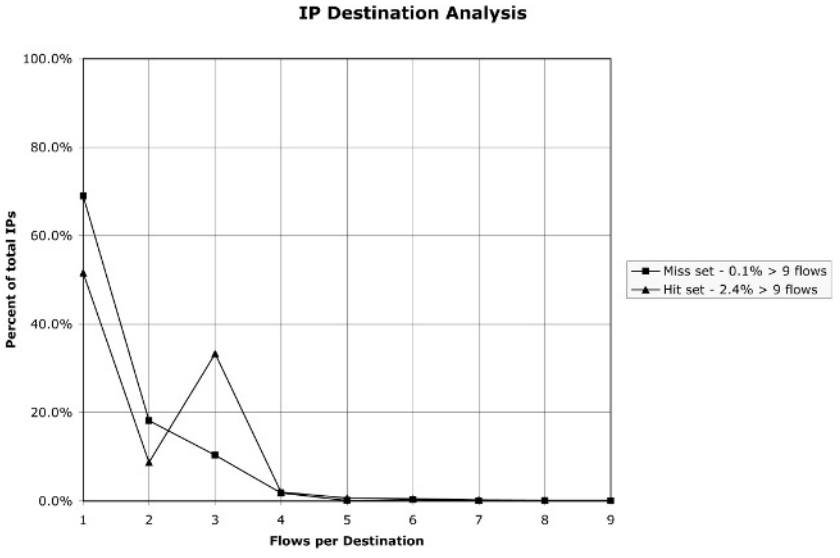

Fig. 2. Reduction in incoming IP Destination Set Sizes as a Function of Number of Flows

```
(39) lip $ readbag --count --print jcm-tcp-s-10+.bag| sort -r -n | head
     12994 AAA.BBB.068.218
      6598 CCC.DDD.209.215
      5944 EEE.FFF.125.117
      5465 GGG.HHH.114.052
      5303 III.JJJ.164.126
```

Fig. 3. The top five sources in the small sample

to IANA). Some 112 responses are seen from hosts SSS.RRR.QQQ.1-78,120-131,224-254. The contiguous ranges and the consistency of responses on a relatively obscure port may indicate the presence of a honey pot or a similar decoy. In all cases, the connection seems to have been broken after the probed host's SYN/ACK reply.

The third and fourth entries are scans of portions of other /16s, this time looking for service on port 20168. This appears to be associated with the "Lovegate" worm which binds a shell to this port on infected machines.

The fifth entry is a scan of a portion of yet another /16, this time looking for service on port 3127, listed by IANA as the CTX bridge port, but also in the range used by the ChiliASP module in Apache servers according to `www.portsdb.org`. This port is currently being used by the "MyDoom.C" worm on Linux[13].

[13] `http://www.linuxworld.com/story/43628.htm`

At the other end of the spectrum, there are 3335 external hosts that sent exactly one TCP flow into the monitored network during the analyzed time interval. Of these, only two port and flag combinations appear more than 100 times. SYN probes for port 8866[14] are seen 449 times. SYN probes for port 25 (SMTP - email) are seen 271 times. The vast majority of the remainder are SYNs to a variety of ports, mostly with high port numbers. There are a number of ACK/RST packets which are probably associated with responses to spoofed DDoS attacks.

5.2 A Week in the Life of a /16

We obtained hourly flow data from a /16 within the monitored network for the one week period from 11 - 17 January 2004. plus two additional days, 26 and 27 January. The data set consists of nearly 400Mb of data divided into hourly files for inside to outside traffic and for outside to inside traffic. The inside to outside traffic was analyzed and a set of IP addresses computed that represent all the hosts that were seen to be active during the initial week. The observed network structure is shown in Table 1. Note that only about 6% of the available /24 subnets are occupied at all and that the utilization of those that are ranges from less than 1% to about 25%. The information shown in Table 1 has been adapted from the output of **readset** using a specification that causes subnet summarization at the /24 level and for the internet as a whole.

Table 1. Network Structure for the selected /16

MMM.NNN.24.x	66 hosts	MMM.NNN.25.x	60 hosts
MMM.NNN.26.x	46 hosts	MMM.NNN.27.x	49 hosts
MMM.NNN.28.x	57 hosts	MMM.NNN.29.x	7 hosts
MMM.NNN.30.x	70 hosts	MMM.NNN.31.x	67 hosts
MMM.NNN.32.x	54 hosts	MMM.NNN.33.x	62 hosts
MMM.NNN.34.x	50 hosts	MMM.NNN.35.x	4 hosts
MMM.NNN.120.x	2 hosts	MMM.NNN.127.x	1 host
MMM.NNN.140.x	1 host	MMM.NNN.251.x	4 hosts

Network Summary
600 hosts ($1.4 * 10^{-5}$%) of 2^{32}
1 occupied class /8 (0.4%) of 256
1 occupied class /16 (0.002%) of 65536
16 occupied class /24s ($9.5 * 10^{-5}$%) of 2^{24}

The set of active addresses seen during the week is an optimistic estimate of the active host set for this network since activity on several subnets was not

[14] "W32.Beagle.B@mm is a mass-mailing worm that opens a back door on TCP port 8866. The worm uses its own SMTP engine for email propagation. It can also send to the attacker the port on which the back door listens, as well as a randomized ID number." according to `http://securityresponse.symantec.com/avcenter/venc/data/w32.beagle.b@mm.html`

observed until late in the week. The set includes all hosts seen to be active during the additional days, as well. This set was used to partition each outside to inside hourly data set into hit and miss portions as described earlier. Since these files consist of a small amount of header data and fixed length records, the file sizes are a good surrogate for the number of flows observed.

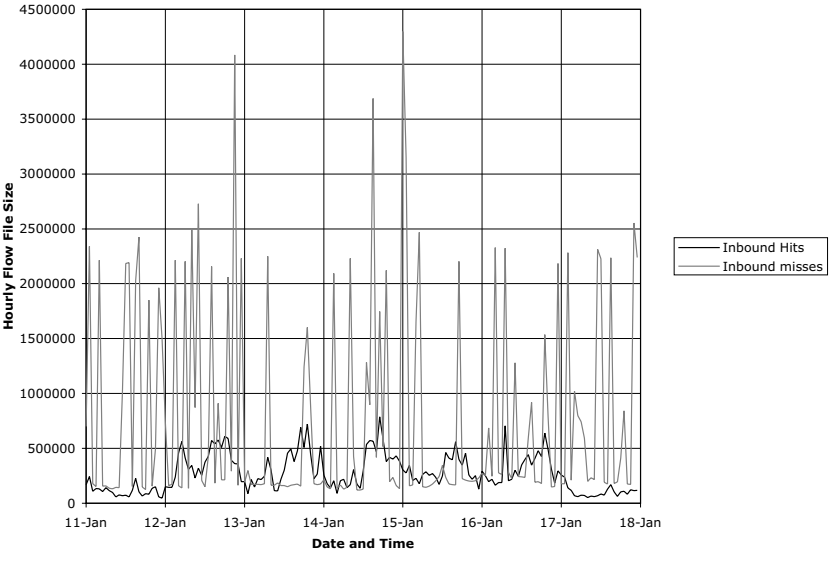

Fig. 4. Hourly flow file sizes for incoming hit and miss flows

Figure 4 shows the hit and miss file sizes for the base week of the data. It is interesting to note that many more flows miss than hit during most hours. We have looked at a few of the peaks in detail.

12 January 21:00 Two fairly complete scans of the entire /16, one for port 2000 TCP (61018 flows) and one for port 20168 TCP (60978 flows) with a total of 62616 unique IP addresses hit. Both scans came from a single IP address. As noted above, port 20168 is associated with the Lovegate worm. Port 2000 appears to be associated with a number of Windows remote administration tools (e.g. RemotelyAnywhere).

14 January 15:00 Two scans from different networks, each targeting port 80 TCP. One targets 58394 hosts, the other 53032.

15 January 00:00 Two fairly complete scans from two distinct source addresses, one for port 4000 TCP, the other for 5308 TCP, each from a separate source.

Port 4000 appears to be used by Remote-Anything from TWD Industries (`http://www.twd-industries.com/en/index.htm`).

Port 5308 is associated with CFengine, a configuration and administration mechanism[15].

Given the large numbers of peaks in the miss data, it was decided to run an IP frequency analysis on the entire week. This produced some interesting results both with respect to the the numbers of infrequent and frequent missers. The set of source IPs that attempt to contact hosts in the miss set contains 182632 distinct addresses. Of these, 145755 or almost 80% appear exactly once. 19751 or about 11% appear twice. 4382 (about 2%) appear 10 or more times, but these account for 3756029 of 4037208 or 93% of the total flows. The top scanners accounted for over 100000 probes each with 66 IP addresses registering more than 10000 flows (over 80/no obvious threshold or breakpoint in the distribution. In the 10000 flow range, adjacent entries in the table are separated by a few hundred flows. The set of the top 66 flow sources contains IP addresses from 65 distance /16 address ranges. In the one case where the addresses are in the same /16 they are also in the same /24 being only 16 addresses apart.

We extracted the miss data for the top 66 flow sources and clustered it based on TCP flags. All the flows are TCP (protocol 6) and the vast majority consist of flows having only a SYN flag. The presence of a substantial number of records containing other flags (especially ACK flags) is somewhat surprising.

Table 2. Flag clusters for the top 66 missed flow sources

Flags	Flows	Src IPs	Dst IPs	S	RA	SA	R	SRA	A
S	2955827	54	64936		22510	1734	8126	49	8
R A	197972	12	22510	1		843	2868	49	6
S A	66908	10	1734	0	9		193	49	6
R	33169	5	8212	1	5	4		9	2
SR A	725	8	49	0	7	8	4		6
A	8	3	8	1	2	2	1	2	

An analysis of the clustering is shown in Table 2. The numbers in the lower triangular section at the right hand side of the table represent counts of source addresses in common among the groups; the upper triangle represents the same analysis for destination addresses. We note that the sources of the SYN only flows are largely disjoint from the sources of the other flows. Since the source of the common RST/ACK and the RST flow are the same, only two IP addresses

[15] "CFengine, or the configuration engine is an autonomous agent and a middle to high level policy language for building expert systems which administrate and configure large computer networks. CFengine uses the idea of classes and a primitive intelligence to define and automate the configuration and maintenance of system state, for small to huge configurations. CFengine is designed to be a part of a computer immune system, and can be thought of as a gaming agent. It is ideal for cluster management and has been adopted for use all over the world in small and huge organizations alike." according to `http://www.iu.hio.no/cfengine/`

Table 3. Ports scanned by the top 54 SYNers

Rank	Count	Port	Service
1	1038778	80	HTTP
2	370404	4899	radmin
3	258538	4000	Remote-Anything
4	225665	21	ftp
5	191562	3389	Remote Desktop Protocol / ms-wbt-server
6	185400	20168	Lovegate worm
7	126142	5308	CFengine
8	64679	2277	
9	64679	554	Real time streaming protocol
10	64679	3410	NetworkLens SSL event
11	64675	1257	Shockwave 2
12	64312	1443	Integrated Engineering Software license manager
13	63166	5000	Cisco HIDS agent/console
14	63165	14232	ftp server as side effect of SMB exploit
15	61018	2000	RemotelyAnywhere / SCCP (Cisco phones)
16	31153	23	telnet

are in common with the SYN addresses. Based on this, we consider the SYN only case separately.

The remaining miss records contained some 25000 distinct ports, mostly high numbers. Analyzing the RST/ACK group separately, we find that the 12 IPs responsible for this behavior form a total of 69 groups when clustered by IP and source port. There is substantial overlap among the IPs in this group and the remaining groups leading us to conclude that it is highly likely that some or all of these hosts were undergoing denial of service attacks during the week of analyzed data. An examination of these flows indicates that it is likely that a number of distinct attacks were involved. One host emits single ACK/RST packets on a number of ports as well as from 1 to 20 or so RST packets on several thousand ports. For the other hosts, most of the activity is concentrated on a relatively few ports, but the patterns vary from host to host with port 80 being the most common. At this point, we do not know whether the activity is concentrated only in the monitored /16 or covers other monitored networks, as well.

Table 4. Clusters based on IP for significant flag combinations

Flags	Clusters	IPs
S	55	54
RA	6	12
SA	12	10
R	3816	5

Table 4 shows the number of unique port and IP combinations for the major flow types. The port used for the SYN flows was the destination port while

source ports were used for the others. This is consistent with our view that flows other than the SYN flows are likely to be responses to abuses of the IP sources by spoofed traffic.

We also briefly examined the very low rate traffic. As noted earlier, the vast majority of the IPs responsible for the miss traffic generate a single flow during the week of observation. Four protocols are represented. There are 14 flows for protocol 0 (IPv6 Hop by Hop?). It is not clear whether these are legitimate, but they originate from 14 distinct /16s and target a single host within the monitored network.

There were a total of 864 ICMP messages. 2 were echo replys, 612 destination unreachable, 3 source quenches,1 host redirect, 242 TTL exceeded in transit, and 1 timestamp reply. Of the destination unreachable messages, 24 were network unreachable, 139 host unreachable, 1 protocol unreachable, 160 port unreachable, 46 communication with host administratively prohibited, and 242 communication administratively prohibited.

There are 13161 TCP flows of which 12541 have only the SYN flag set. 12242 of these target port 80 (http), with port 4662 (EDonkey, a peer to peer service) next with 28 flows. 11 flows target port 0 (possibly an error), and 10 target port 55592, and 8 target port 23 (telnet). The remainder of the flows have fewer that 5 flows per port with most being 1 flow per port. Of the non SYN only flows, there are 80 SYN/ACKs and 80 RST/ACKs to port 80. There are 14 RST/ACKs target port 1025 (network blackjack?) and 11 targeting port 21 (ftp) Ignoring ports in the analysis, there are 226 RST/ACK combinations, 177 RST, 82 SYN/RST, 46 ACK, and a handfull of other combinations.

The largest component of the low rate data is UDP. There are 131716 flows of which 131519 target destination port 53 (dns). The next highest counts are less than 30 flows to a single port and singleton flows appear at rank 25 (of 74). Clustering on source ports produces a different picture. 25228 ports appear in the list, again, most associated with a single flow. Port 53 is again the most common source port (89450 flows) followed by 1025 (1042 flows). Ports 10000, 1024, 1044, 1064, 10001, 1084, 60000, and 1026 round out the top 10 (tie for 10th place at 60 flows). Clustering on input and output port pairs shows that the combination (53,53) is most common with 89531 flows. Port 53 is the destination port in the 27 top flows and is either the source or destination port in the top 39 flows. (500,500) is the 40th most common combination with 14 flows and port 53 appears as the destination port for ranks from 41 through 197. Clustering by destination host and port is unproductive. The most frequent clusters all target a single host with 15 flows to port 6144, and 14 each to ports 53 and 16896 on the same host.

6 Enhancements and Extensions

The SiLKtools set has shown itself to be useful for a variety of analyses of large data sets of network flow data. For future work, there are a number of extensions that we would like to make. Extensions to the set and bag mechanism to allow

adding of bags, conversion of a bag to its basis set, and to allow a set to be used to select portions of a bag would be particularly useful. While stand alone programs such as the ones we have used here are useful for exploratory work, we envision routine analyses that would benefit from being performed in an integrated program. To this end, we are proposing to the SiLKtools developers that they consider implementing the set and bag code as a subroutine library, along with a library of data structures and I/O routines that can be used to access and manipulate the raw data.

The underlying data representation also has some deficiencies that will be fixed in the near future. Time is currently represented only in whole seconds which makes it difficult to compare, for example flow start times. We are working on matching inbound and outbound flows for TCP sessions. Better timing information would help here, as would a flag that indicated that the first packet of the flow contained a SYN and not a SYN/ACK combination. As mentioned earlier, we are looking at creating flows from other data sources.

Extending the set and bag concept to other data types would be useful. We have need for efficient representations of active ports associated with individual IPs, for example. The bags currently provide 32 bit counters for the IPv4 space, but depend on the sparse non-zero counts to fit in memory. We have recently analyzed the several alternative representations and feel that we can build a set and bag framework that will allow much more compact representations than we have at present by taking advantage of observed address distributions. The current structures use one (sets) or 2 (bags) levels of indirection and then allocate blocks of storage at the /16 (sets) or /24 (bags) level. Because of the low occupancy of most of our customer's subnets combined with a tendency to allocate addresses at the low end of the subnets, it appears that /27 allocations of storage could reduce bag sizes by a factor of 4 to 8. Allocating 8 bit counters to the block initially and expanding to 16 bits or 32 bits as needed would reduce bag sizes by another factor of 2 or more.

We also have a need for set and bag like structures for things like connections (2 IP addresses and 2 port addresses, reversed for response) and connection attempts (Source IP, Destination IP and port). We have experimented with the use of Bloom filters[3] for determining connection sets and believe that these could play a major role in analyzing more complex behaviors such as determining the number of distinct connection attempts a given source IP makes over time. We are currently developing a plugin for **rwfilter** that will allow collection for continuation flows, i.e. groups of flows with the same IP, protocol, and port information, using a two stage Bloom filter. All flows will be entered into the first stage of the filter, but only flows that match the first stage are entered into the second filter which can then be used to separate singleton flows (the vast majority) from those that might be consolidated. Similarly, transposing source and destination inputs to a Bloom filter can simplify matching the two sides of a conversation.

And then there is the question of extending the concept to IPv6 . . . .

7 Conclusions

We have introduced the concept of using sets and bags to facilitate the analysis of large sets of network flow data, provided some motivating examples and suggested further lines of inquiry. The tools we use are in a state of flux, but have already produced interesting and useful results in analyzing both aggregated data and data from single subnets.

Acknowledgments

Special thanks go to Mike Collins who has been the chief instigator of the SiLKtools effort since the death of Suresh Konda. Carrie Gates and Damon Becknel have provided many useful contributions to the effort. Other members of the CERT analysis team have been most gracious in tolerating my code modifications.

References

1. Fullmer, M., Romig, S.: The OSU flow-tools package and Cisco NetFlow logs. In: LISA XIV, New Orleans (2000) 291–303
2. Tukey, J.W.: Exploratory Data Analysis. Addison-Wesley, Reading, MA. (1977)
3. Bloom, B.: Space/time trade-offs in hash coding with allowable errors. Communications of ACM **13** (1970) 422–426

A Brief Descriptions of Selected SiLKtools Programs

rwfilter is the front end for the system. It extracts a subset of the centrally stored rwdata based on a number of aggregation and filtering criteria or applies filtering data to a previously extracted rwdata file. Although it is much more general than the usages we show, our typical use starts by selecting data based on some time interval and possibly some range of network addresses. This data will be stored in a single file and subsequently refiltered based on other criteria. The program can be used to partition data based on the filtering criteria. Thus, TCP data could be extracted to a file in one pass with other protocols going to different file (or piped to another invocation of rwfilter to extract UDP, etc. for as many levels as desired.) Since sets of IP addresses can be used as filter criteria, traffic whose source (and / or destination) addresses appear in given sets can be extracted or partitioned.

rwcut lists selected fields from an rwdata file, one line per flow, for subsequent analysis. A dynamic library facility allows derived quantities, e.g. bytes per packet, to be listed.

rwsuperset is my enhanced version of **rwset**. It can create multiple sets and bags from rwdata files based on source and / or destination addresses. An additional feature allows a diminished set of the input data (flows with IP

addresses already in a set under construction) to be passed for source, destination or both. This allows us to cascade invocations using sets for the majority of flows with small counts and bags for the residue as seen in the following hack (to be obviated by a more efficient bag representation).

```
#!/bin/bash
# raw data file $1.rwf creates 9 levels of set files and a bag file.
# $2 is either s or d, for bagging on source or destination IPs
rwsuperset --p --$2-s=$1-$2-1+.set --$2-d=stdout $1.rwf |\
rwsuperset --p --$2-s=$1-$2-2+.set --$2-d=stdout |\
...
rwsuperset --p --$2-s=$1-$2-10+.set --$2-d=$1-$2-10+.rwf \
           --$2-i=10 --$2-b=$1-$2-10+.bag
```

readset reads in a set file and lists its contents in various ways including set size, a list of members, or an analysis of the network structure of the set.

readbag is similar to readset, but will only enumerate the bag,

buildset builds a set file from a list of IP addresses. Wildcards are permitted as are CIDR notations, useful in building sets for masking operations.

buildbag is like buildset except that it counts occurrences of each IP address.

setintersect performs intersection operations on set files or their complements.

rwsetunion performs the union of two or more set files.

rwcat concatenates multiple rwdata files into a single file or stdout stream.

rwsort provides the ability to sort rwdata files based on address, size or start time fields.

rwstats can provide a variety of statistics about the contents of an rwdata file.

cluster.sno is written in SNOBOL4[16] and counts the number of unique lines in its input. This is a simplistic view of clustering, but is adequate for now.

[16] Available at `http://www.snobol4.org/`