CANINE: A Combined Conversion and Anonymization Tool for Processing NetFlows for Security

Yifan Li, Adam Slagell, Katherine Luo, William Yurcik
National Center for Supercomputing Applications (NCSA)
University of Illinois at Urbana-Champaign
{*yifan,slagell,xluo1,byurcik*}*@ncsa.uiuc.edu*

Abstract

Those creating NetFlow tools struggle with two problems: (1) NetFlows come in many different, incompatible formats, and (2) the sensitivity of NetFlow logs can hinder the sharing of these logs and thus make it difficult for developers—particularly student research assistants—to get real data to use. Our solution is a new tool we created that converts and anonymizes NetFlow logs. In this paper we discuss our tool in detail and demonstrate that it is extremely scalable.

Keywords: NetFlows, computer network security monitoring, intrusion detection, network management

1 Introduction

The monitoring of a network's security situational awareness through visualizations has been demonstrated to be an effective and efficient aid to intrusion detection. The quality of the source data used in a visualization tool is directly related to this effectiveness. Due to their unique level of abstraction over raw packets, NetFlows are increasingly being used by security engineers to monitor security events. The most commonly used NetFlows formats are Cisco [3] and Argus [1] NetFlows, though there are many others as well as subcategories of Cisco versions. One of the data management issues of using NetFlows is that they exist in so many formats. In this paper, we introduce CANINE (Converter and ANonymizer for Investigating Netflow Events), a tool that augments existing flow tools. It augments them as it enables tools working exclusively with one type of NetFlows [6, 11] to operate on data from NetFlows in other formats. This is very beneficial given the fact that different types of NetFlows can come from complementary sources as the format is often tied to the routing hardware or computer collecting the data.

In addition to issues about formats, people often have concerns about information disclosure when publishing results or performing demonstrations that utilize sensitive NetFlow logs. In order to address this second problem, we have also integrated anonymization capabilities with the converter. For instance, we implement a prefix-preserving encryption algorithm [10] to anonymize the IP address field. In this way, the concrete IP address information is concealed while the subnet structures are preserved. This is critical for some NetFlow visualization tools [6, 11]. We also support the anonymization of several other fields: time stamps, port numbers, protocols, and byte counts. This converter/anonymizer has been vital to the development of visualization tools developed at NCSA[6, 11] as it allows students to work with sensitive log data. We expect this work to likewise promote better insight into the use of NetFlows for security and network performance monitoring at other institutions.

The rest of the paper is organized as follows. Section 2 discusses the file types CANINE can currently handle. The supported anonymization methods for different fields are presented in Section 3. Section 4 shows a screen shot of CANINE and discusses its interface. We report experimental performance results in Section 5 and conclude this paper in Section 6.

2 NetFlow Conversion

A *network flow* is defined as a sequence of packets that are transferred between two endpoints within a certain time interval (typically a half hour at most). The endpoints are uniquely identified by IP address as well as the transport layer port number (for TCP and UDP traffic). NetFlows are a useful data source at a granularity that is scalable for network management or security analysis. Accordingly, NetFlows find broad applications, including network monitoring, network planning and analysis, accounting/billing, application/user monitoring and profiling, and data warehousing/mining.

With the increased use of NetFlows for security monitoring [9], more and more tools based on NetFlows are being built and deployed. However, the

differences between the formats of different NetFlows impede the progress of network security monitoring, since many, if not all, tools that are based on NetFlows support only one format, while organizations often use multiple formats.

Specifically, we were motivated to develop our NetFlow converter to augment our existing flow tools [6, 11] by enabling them use NetFlows from the multiple, heterogeneous sources here at the NCSA. The different NetFlow sources, as well as collectors deployed, often log in different ways, creating different incompatible versions of NetFlows.

In this section, we briefly introduce some of the NetFlow formats currently supported by CANINE.

2.1 Cisco NetFlows

As defined in [4], a *Cisco NetFlow* record is a *unidirectional* flow that is identified by the following unique keys: source IP address, destination IP address, source port, destination port and protocol type. As described in [3], Cisco NetFlows are generated with a set of sophisticated flow cache management algorithms. These algorithms determine whether a packet should be included in an existing flow or whether it should generate a new flow cache entry. They also perform flow statistic updates and handle flow aging/expiration.

The expired flows are clustered together to form *NetFlow Export* UDP datagrams which are transferred from the NetFlow-enabled devices to flow collectors (e.g., dedicated workstations). The NetFlow Export datagrams contain approximately 1500 bytes, which amount to about 20–50 flow records. Currently there are multiple versions of Cisco NetFlows (e.g., V1, V5, V7, V8 and V9), and newer Cisco equipment tends to be backwards compatible in generating older formats. In all versions, the datagram consists of a header and one or more flow records (Figure 1). The header contains the version number of the export datagram, which is necessary to properly interpret the datagram. The second field of the header contains the number of records in the datagram and is used to index the individual records. Thus, datagrams are generally variable in length due to the variation in the number of records, even if all the datagrams belong to a particular version. The most frequently used versions are Cisco NetFlow version 5 and version 7, whose important header and flow record fields are described in Table 1 and Table 2, respectively. For more details about the formats of each version, readers are referred to [3].

Cisco NetFlow collectors provide fast, scalable, and efficient data collection from multiple NetFlow-enabled devices—typically these are routers. Foremost, a NetFlow collector consumes the NetFlow datagrams from multiple sources, performs data reduction through filtering and aggregation, and stores flow information in flat files that are ready for further processing (e.g., visualization and analysis).

Currently, CANINE supports both Cisco V5 and V7 NetFlows, which are the most commonly used.

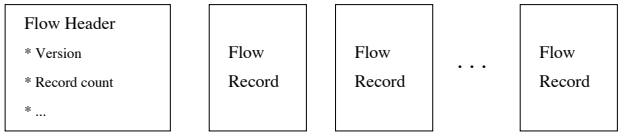

Figure 1: The Cisco NetFlow export datagram structure.

Item	**Description**	**Length**(B)
version	NetFlow format version number	2
count	number of flow records in this packet	2
SysUptime	current time since the device booted	4
unix_secs	current time since 0000 UTC 1970	4

Table 1: Cisco NetFlow version 5/7 header, primary contents

Item	**Description**	**Length**(B)
src_addr	source IP address	4
dst_addr	destination IP address	4
dPkts	number of packets	4
dOctets	number of bytes	4
first	SysUptime at start of flow	4
last	SysUptime at end of flow	4

Table 2: Cisco NetFlow version 5/7 flow record, primary contents

2.2 Argus NetFlows

Argus is a real time flow monitor that is able to track and report network transactions it observes from packets collected on a network interface in promiscuous mode [1]. In contrast to Cisco NetFlows, Argus views each network flow as a *bidirectional* sequence of packets that typically contain two sub-flows, one for each direction. Similar to Cisco flows, each flow record contains the attributes of source IP, source port, destination IP, destination port, protocol type, etc (Note that the source and destination are interchangeable here since a network flow is bidirectional). Similar to a Cisco NetFlow, an Argus flow is a set of packets that share a common set of attributes, including addresses, protocol, ports used, session IDs, etc. According to [5], a new flow is created when a packet is encountered that does not match the attributes of an existing flow. Argus records the time when the new flow is created. Some other flow attributes (IP addresses, ports, protocols, etc.) are determined at the same time. A *LastTime* value is associated with each flow and indicates the time when Argus captured the last packet of the flow.

There are two types of Argus records: the *Management Audit Record* (Table 3) and the *Flow Activity Record* (Table 4), where the former provides information about Argus itself, and the latter provides information about specific network flows that Argus tracked. Each record shares a common header (Table 3). Note that the tables shown here are simplifications of the original formats. To be inundated with details of the format, readers are referred to [2].

The Management Audit Record (MAR) provides the meta-data about the Argus version, and a *Start MAR* is always the first record in the stream. *Status MAR*s are also generated by Argus periodically so that the client reader can make sure the Argus file is correct. An optional *Stop MAR* should be the last record in a well formed Argus stream. Thus, a typical Argus stream will look like the following:

```
Start MAR record
FAR record
...
Status MAR record (optional)
FAR record
...
Stop MAR record (optional)
```

We use the *RA* client to read Argus streams to generate NetFlows in ASCII format.

Item	**Description**	**Length**(B)
starttime	start time of the record	8
pktsRcvd	number of packets received	8
bytesRcvd	number of bytes received	8

Table 3: Argus MAR record, primary contents

Item	**Description**	**Length**(B)
ip_src	source IP address	4
ip_dst	destination IP address	4
start	start time	8
last	end time	8

Table 4: Argus FAR record, primary contents

Item	**Description**	**Length**(B)
type	type of the Argus record	1
length	length of the record	2
status	a bit flag to identify general conditions	4

Table 5: Argus header, primary contents

2.3 NCSA Uniform Format

Since different versions of Cisco NetFlow Export datagrams are generated by the diverse routing equipment at the NCSA and because Cisco datagrams are of variable length, we have created an *NCSA Uniform format* for use by our visualization tools. This not only enables easier access control and data manipulation, but the fixed length records are necessary for our visualization tools ([6, 11]) that depend upon random access to NetFlow records. Furthermore, NCSA uniform format serves as an internal format into which multiple versions of NetFlows can be transformed. Each record (44 bytes) contains the principle information about a network flow, including IP addresses, ports, protocol used, bytes transferred, etc (Table 6).

2.4 Nfdump NetFlows

As part of project NfSen [8], nfdump tools collect and process network flow data. The data is stored in the nfdump format [7], which is a variation of the Cisco NetFlow format (version 5). Due to the increased use of these tools in network monitoring and administration, we provide support of the nfdump format in CANINE.

3 NetFlow Anonymizer

The anonymization engine of CANINE supports the anonymization of several fields, often in multiple ways. Below we describe the different anonymization algorithms supported by CANINE at this time.

3.1 IP Anonymization

3.1.1 Truncation

Truncation is the most basic type of IP address a-nonymization. Here the user chooses the number of least significant bits to truncate from an IP address. For example, truncating 8 bits would simply replace an IP address with the corresponding class C network address. Truncating all 32 bits would replace every IP with the constant address of *0.0.0.0*. Truncation is probably the most common type of log anonymization currently employed.

Item	**Description**	**Length**(B)
version	version of Cisco NetFlow	1
padding	set to 0	1
router IP	router's IP address	4
src_ip	source IP address	4
dst_ip	destination IP address	4
src_port	source port number	2
dst_port	destination port number	2
flow_bytes	number of bytes	4
flow_packets	number of packets	4
prot	protocol	1
flags	TCP flags	1
start_time	start time (seconds since epoch)	4
start_time_offset	milliseconds offset of start time	2
end_time	end time (seconds since epoch)	4
end_time_offset	milliseconds offset of end time	2
padding	set to 0	4

Table 6: NCSA uniform record format

3.1.2 Random Permutations

With this method, a random permutation on the set of possible IP addresses is applied to translate each IP address. A 32-bit block cipher would represent a subset of permutations on the IP space. We implement a truly random permutation through use of random hash tables. In this way, it is possible to generate any permutation, not just one from some subset of the possible permutations.

3.1.3 Prefix-preserving Pseudonymization

Prefix-preserving pseudonymization is a special class of permutations that have a unique structure preserving property. The property is that two anonymized IP addresses match on a prefix of n bits if and only if the unanonymized addresses match on n bits. We implement this algorithm in such a way that a user supplied passphrase generates an AES key that in turn determines the permutation. This is fast and allows anonymization to be done in parallel with a consistent mapping between anonymizers. This is difficult to do when shared tables are used as in the previous method.

3.2 Timestamp Anonymization

3.2.1 Time Unit Annihilation

Timestamps can be broken down into the units of Year, Month, Day, Hour, Minute and Second. We support the annihilation of any subset of those units. If one wishes to remove the hour, minute and second information, they can do so. Likewise, if someone wishes to obfuscate the date, they can remove the year, month and day information. If they want to completely eliminate time information, i.e. perform *black marker* anonymization of the entire field, they can select all of the time units for annihilation. Ending times are adjusted so that the duration of the flow is kept the same.

3.2.2 Random Time Shifts

In some cases it may be important to know how far apart two events are temporally without knowing exactly when they happened. For this reason a log or set of logs can be anonymized at once such that all timestamps are shifted by the same random number. If one does this to two different sets of logs at different times, then this random number will be different between the data sets. This means that data-mining cannot be done by indexing the time field between the data sets. The solution requires the ability to choose the number by which to shift. However, it seems cumbersome and impractical for data owners to save and keep track of shifting amounts used on different logs. That is why we do not support that ability in CANINE but instead warn users to be aware of the troubles with data-mining (by indexing the timestamp) between sets anonymized at different times when using this specific method.

3.2.3 Enumeration

All time information could be removed except the order in which the events occurred. In this method, the algorithm simply chooses a random time for the earliest record. All other starting times are equidistant from each other and in chronological order. Corresponding ending times are calculated from the original flow duration. Implementation of this method can be troublesome, especially when dealing with streamed data. The problem arises because entries in the logs are not presorted by starting or ending time. They are close to being in order by ending time, but they are not in perfect order. Sorting cannot work perfectly on streamed data, and it would be extremely slow on large log files. A good solution is to buffer events to sort locally. Since events are never terribly disordered, this can sort things with great accuracy. If data is from multiple routers, there will likely be small errors in this regard anyway, due to time skews between routers. We support this faster, local sorting method. Starting times are adjusted so that the duration of the flow is kept the same.

3.3 Other Fields to Anonymize

3.3.1 Port Number

Bilateral Classification Usually the port number is useless unless one knows exactly what it is. However, there is one important piece of information that does not require one to know the actual port number: whether or not the port is ephemeral. In this way we can classify ports as being below 1024 or above 1023. To make the output look the same as the input, a representative of the set, such as port 0, can replace all non-ephemeral ports, and 65535 can replace all ephemeral port numbers. While it should be easy to recognize when this type of anonymization is performed, it is theoretically possible—though impossible for all practical purposes—for a log to contain only entries with ports 0 and 65535 being used. In this case it would be ambiguous as to

whether the port number has been anonymized. Meta-data may be necessary to make it clear, though CANINE does not generate any meta-data. This method of anonymization is very similar to truncation of IP addresses in that we are collapsing a subset of ports down to a single representative within that set.

Black Marker Anonymization This is the same from an information theoretic view as printing the logs and blacking out all port information. In a digital form, we just replace all ports with a constant. Port 0 is a good candidate for the constant. We need to use a 16 bit representation for 0 so that programs that process unanonymized logs can still process anonymized logs. We have been careful to ensure that anonymized logs do not break current tools by changing the format.

3.3.2 Protocol

While we can conceive of no reason to anonymize this field, protocol information can simply be removed. We do this by replacing the protocol number with the unused, but IANA reserved, number of 255. This is the maximal number for that 8 bit field.

3.3.3 Byte Count

It is conceivable that one may wish, for privacy reasons, to anonymize byte counts. Users may not want others to know whether or not they are using a lot of bandwidth. Thus we support black marker anonymization of this field where all byte counts are replaced with the constant of 0, an impossible byte count in reality because headers do account for some of those bytes.

4 CANINE GUI

Figure 2 displays a screen-shot of CANINE. The underlying window is the main menu for CANINE where one can select source/destination files, conversion settings and the fields to anonymize. Additionally, a progress bar (out of view in this snapshot) is present to indicate progress when processing a log file.

The window in focus presents the options for IP anonymization. In Figure 2, the truncation method has been selected with it set to zero-out the 10 least significant bits of the IP addresses via the sliding bar. The other options (e.g., random permutation and prefix-preserving pseudonymization) are grayed out since it is impossible to arbitrarily mix anonymization methods on the same field, though multiple fields can be anonymized each in a different way in the same log.

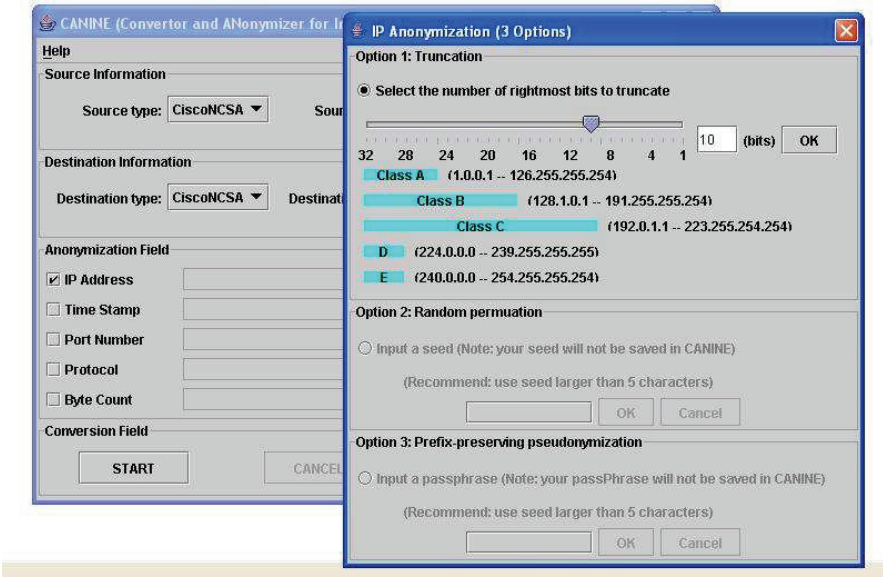

Figure 2: Screen shot of our CANINE tool.

5 Empirical Evaluation

All experiments were run on a 2.4 GHz Xeon (Dell 650 N) with 1 GB of memory, running a Linux 2.4.20 kernel. Each data point in the following figures represents the mean of 20 runs of the exact same experiment.

5.1 Experiments with the Conversion Engine

Since each conversion of a record can be bounded in $O(1)$ time, we expected the empirical data to to corroborate this. As you can see from Figure 4, this is indeed the case. Since each record is simply processed sequentially, this means that the converter should scale linearly with respect to the number of

records. As the number of records is directly proportional to the file size, we would expect CANINE's converter to scale linearly with respect to file size as well. This is what happens as can be seen from Figure 4.

In both figures, we are showing conversion times from all the supported formats to the fixed length binary format that we call CiscoNCSA. This is a format we use internally for tools we developed to visualize NetFlows (The Cisco 5/7 format is just a log that contains a mix of Cisco 5 and 7 records gathered at a single collector). Immediately, one notices that the conversion from Argus to CiscoNCSA is much slower—up to 5.5 times slower. This is because the Argus file is the only type in ASCII, rather than binary. It is slower because it is more difficult to parse, many string operations must be performed, and individual fields must be converted to binary. For example, IP addresses must be transformed from a variable length dotted decimal representation to a 32 bit binary field. This conversion must only be done for the Argus logs since the others already represent the IP address fields in binary form.

Still, none of the conversion methods are very slow on an absolute scale. For example, a 100 MB Cisco 5 log can be converted in only 30 seconds. This is many times faster than necessary to convert in real-time. Even in a large grid environment that generates 2GB of NetFlows a day, as we do at the NCSA, real-time works out to about 1.3 MB per minute.

5.2 Experiments with the Anonymization Engine

We conducted tests of every anonymization option for each of the log formats. However, due to space limitations, we only show the results of anonymizing files in CiscoNCSA format. Results for other log formats are extremely similar.

Figure 5 presents the performance results for each of the IP address anonymization algorithms as well as a baseline of *no anonymization* (For the IP truncation method, the 13 least significant bits were truncated). From the figure, we observe that IP truncation and IP permutation entails very little overhead—the execution time is almost the same using no anonymization and just copying the log. On the other hand, the prefix-preserving method takes much longer. This is likely due to the fact each record requires 64 calls to an AES encryption function, essentially the same as encrypting 1 KB of data. While not slow in absolute terms, it is much slower than the other methods. As expected, linear growth with respect to file size is observed.

Figure 6 displays the results for the 3 different types of timestamp

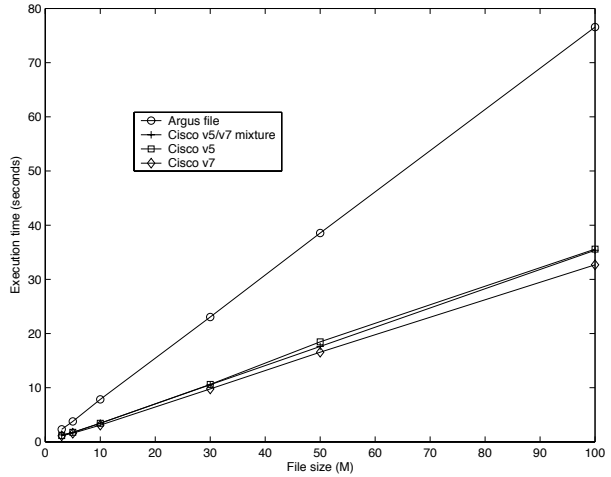

Figure 3: Execution time for different file type and size being converted into CiscoNCSA.

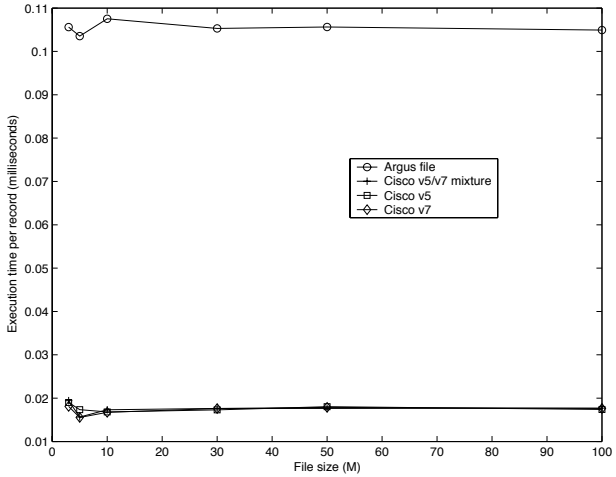

Figure 4: Execution time per record for different file type and size being converted into CiscoNCSA.

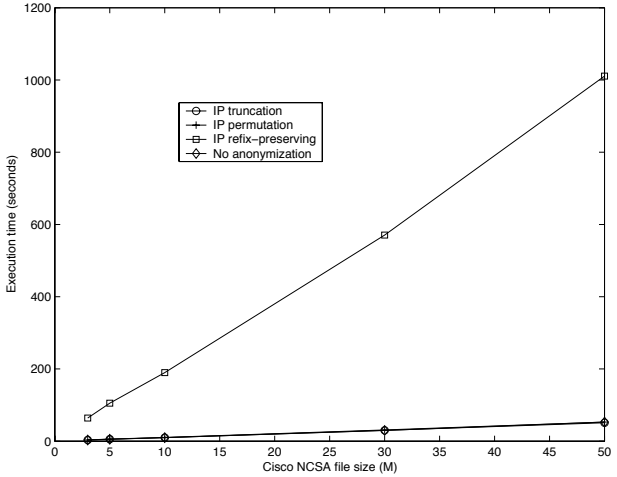

Figure 5: IP anonymization.

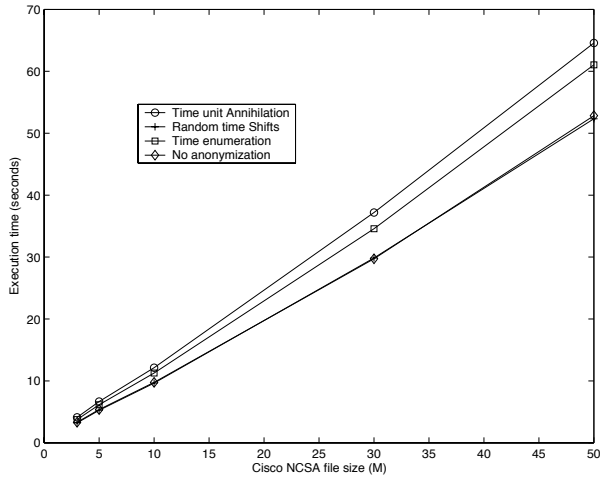

Figure 6: Time anonymization.

anonymization we support as well as the baseline of no anonymization. In particular, the time unit annihilation method was set to annihilate all time units except the month, day and minute information. The window in the time enumeration method, which is used to set a size for a buffer to sort records, was set to 100 records during the experiments. Unlike the IP address anonymization, there is no greatly pronounced difference in performance between different methods. Though, time unit annihilation does take a bit longer due to the transformation between date representations. As expected, we see the anonymization scales linearly with respect to file size.

The results of anonymizing the remaining fields (port number, protocol, and byte count) are shown in Figure 7. Generally speaking, minimal extra cost is involved in these methods since the anonymization algorithms are extremely simple. Thus most of the time is spent just doing file I/O. As with every other anonymization method, these algorithms scale linearly with respect to log file size.

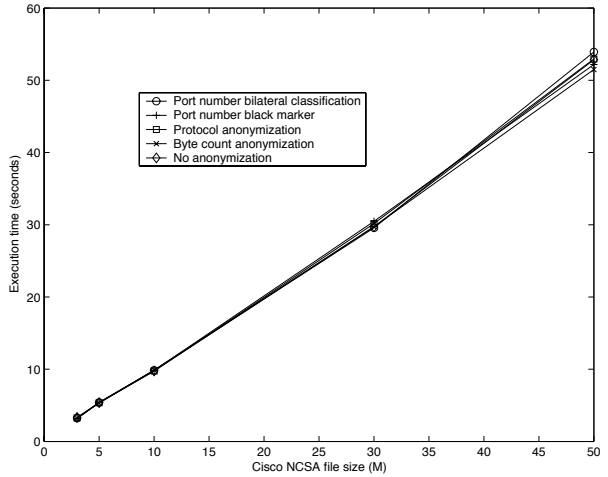

Figure 7: Port number, protocol, and byte count anonymization.

6 Summary

In this paper we have have presented two important problems facing developers of NetFlow tools: (1) NetFlows come in many different, incompatible formats, and (2) the sensitivity of NetFlow logs can hinder the sharing of these logs and the development of tools to operate on them. We presented our solution to these problems: CANINE (Converter and ANonymizer for Investigating Netflow Events). We discussed, in depth, the types of conversion and anonymization supported by CANINE. Furthermore, we demonstrated CANINE's scalability which is linear with respect to log size for all operations.

While there are many options to anonymize a NetFlow with CANINE, it can still be difficult to choose the correct options for a particular organization's needs. Thus, future work should focus on creating multiple, useful levels of anonymization that trade-off between the utility of the anonymized log and the security of the anonymization scheme. This work should also strive to help organizations map levels of trust shared with would-be receivers to these different levels of anonymization.

References

[1] C. Bullard, *Argus, the network Audit Record Generation and Utilization System*, website: <http://www.qosient.com/argus/>.

[2] C. Bullard, *Argus record format*, website: <http://www.qosient.com/argus/argus.5.htm/>.

[3] *Cisco NetFlow Services and Applications White Paper*, <http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm>.

[4] K. Claffy, G. C. Polyzos, and H.-W. Braun. *Internet traffic flow profiling*. UCSD TR-CS93-328, SDSC GA-A21526, 1993.

[5] S. Handelman, S. Stibler, N. Brownlee, G. Ruth. *RTFM: New Attributes for Traffic Flow Measurement*. <http://www.rfc-editor.org/rfc/rfc2724.txt>

[6] K. Lakkaraju, W. Yurcik, A. Lee, R. Bearavolu, Y. Li, and X. Yin, *NVisionIP: NetFlow Visualizations of System State for Security Situational Awareness*. VizSEC/DMSEC, 2004.

[7] *NFDUMP*, <http://nfdump.sourceforge.net/>.

[8] *NFSEN*, <http://sourceforge.net/projects/nfsen/>.

[9] Yiming Gong, *Detecting Worms and Abnormal Activities with NetFlows* Security Focus Article, August 2004.

[10] A. Slagell, J. Wang, and W. Yurcik, *Network Log Anonymization:Application of Crypto-PAn to Cisco Netflows* SKM Workshop, 2004.

[11] X. Yin, W. Yurcik, M. Treaster, Y. Li, and K. Lakkaraju *VisFlowConnect: NetFlow Visualizations of Link Relationships for Security Situational Awareness*. VizSEC/DMSEC, 2004.