Aggregation of Network Protocol Data Near its Source

Marcel Fourné, Kevin Stegemann, Dominique Petersen, and Norbert Pohlmann

Institute for Internet Security, Westfälische Hochschule, University of Applied Sciences, Gelsenkirchen, Germany
[fourne,stegemann,petersen,pohlmann]@internet-sicherheit.de

Abstract. In Network Anomaly and Botnet Detection the main source of input for analysis is the network traffic, which has to be transmitted from its capture source to the analysis system. High-volume data sources often generate traffic volumes prohibiting direct pass-through of bulk data into researchers hands.
In this paper we achieve a reduction in volume of transmitted data from network flow captures by aggregating raw data using extraction of protocol semantics, orthogonal to classic bulk compression algorithms. We propose a formalization for this concept called Descriptors and extend its use to network flow data. Using this approach a preliminary selection of protocol information can be deferred into detail analysis.
A comparison with common bulk data file compression formats will be given for full Packet Capture (PCAP) files. Our approach aims to be compatible with Internet Protocol Flow Information Export (IPFIX) and other standardized network flow data formats as possible inputs.

Keywords: Network Anomaly Detection, Botnet Detection, Network Flow Data Formats, Data Compression

1 Introduction and Background

For Network Anomaly Detection as well as network based Botnet and Malware Detection (common umbrella term: NAD) the network traffic is their defining input. Any findings rely on the availability of large volumes of network data. The standard approach to get network data from Internet Service Providers (ISPs) is to simply request them for research or security purposes under a contract. A common problem emerges from the difference in bandwidth of big Internet exchanges compared to the bandwidth of common research laboratories.

The most common format for network flow data is called NetFlow, which was the basis for the more modern and standardized IPFIX. Since IPFIX is the future replacement for NetFlow, it can be expected to get widespread implementation in network routing equipment. The main target for this protocol is the transfer of data from an exporter to a collector.

When transmitting data for analysis, the most common formats in use today are NetFlow version 5 and raw PCAP files. Using NetFlow or IPFIX, the set

of data to analyze is equally dependent on the supplier — ISPs etc. — so both have different problems. The first has — in its minimal form — only limited information for reliable detection of malicious activity and the second is not reasonable to transmit from live, high bandwidth data sources. A wider selection of records in NetFlow or IPFIX leads to more useful data, but in return raises the requirements for larger transmission bandwidth.

A possible workaround is compression of the bulk data using the DEFLATE or similar compression algorithms, but this only achieves modest improvements.

2 Related Work

There are projects using different approaches to NAD, including protocol headers [1], statistical detection [2] or totally different approaches. We will only name a few as a primer for the interested [3, 4].

Most other research is done using already available NetFlow data using a minimal dataset [5–7] and sometimes also direct packet captures [8], but other flow data formats are also contestants for statistic approaches in finding anomalies. The main benefit of sampling flow data formats, by only transferring every N-th network packet is to be able to get information on Internet links which would otherwise saturate NAD systems is also their main disadvantage: Since not every packet gets reported, finding anomalies becomes a work of chance.

NetFlow [9] in itself is widely available in commercial routing hard- and software, so NetFlow data are easy to collect and many data are extractable from NetFlow records. The data available from NetFlow is variable, so the least common denominator is the only guaranteed information set, all other reportable records are optional. This may hamper NAD research if probabilistic analysis has to be inserted to find common higher level protocol usage without having detailed records available [10]. Standardization of the NetFlow format has led to IPFIX [11], which will see further adoption first in industrial use and therefore later in scientific literature using data in this format.

The general idea of Descriptors has been mentioned in [12], but their approach was based on time intervals alone, thus not allowing to detect parties connected to anomalous activity in network flows. This is not needed for simple alarms, but for precise detection of attack sources.

3 Approach

We propose a security flow data transfer format with aggregate network protocol data to reduce the need for bandwidth compared to other formats containing the same information without aggregation. The aim of this format is to enable analysis of high bandwidth network data sources at smaller bandwidth sites. The security information content has to be maximized, of course, else a NAD system would be hampered in its workings. We define the security information (SI) of

the data using P (Information at the Producer), D (Data at a sensor) and Y (Result of aggregation) as

$$SI(Y) \leq SI(D) \leq SI(P) \tag{1}$$

and

$$Y <<< P,\ SI(Y) \leq SI(P)\ (\text{ideally } SI(Y) = SI(P)) \tag{2}$$

The transport format Version 5 is structurally similar to common flow formats with the addition of so-called Descriptors, carrying the aggregate protocol data. Additional flow information can be added, e.g. labeling information, full packet size lists or the time distance of packets — called time difference list.

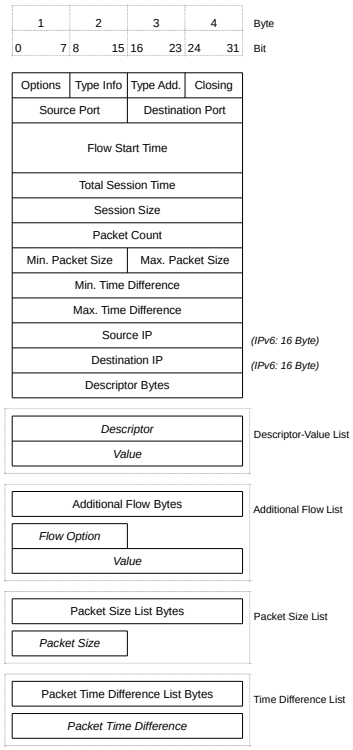


Fig. 1. Network Flow aware packet format v5 for Descriptor-Value pairs

Of special note is the semantic approach in reducing the transferable network protocol data, since it relies on expert knowledge to implement aggregation plugins. They are as stable in their implementation details as the network protocol standards, so it would be combinatorically complex to implement all network protocols. It is in our best interest to implement a subset of most common

network protocols instead of taking an "every last bit" approach, since very uncommon protocols only falsify the network view by very small and easily quantifiable percentages, so the Descriptors compress large amounts of data very well.

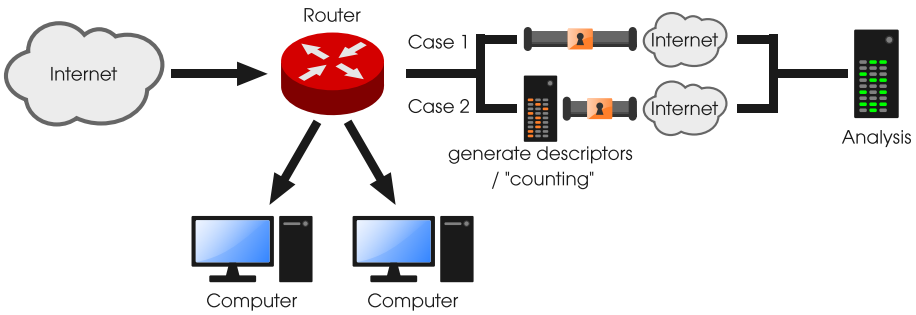


Fig. 2. Network setup comparing with and without Descriptor counting

We implemented the data aggregation not at a research laboratory, but on-site of the ISP to be able to extract network protocol data for detection right before transmitting the data to researchers. This can be seen as case 2 of Fig. 2, case 1 being the standard case, e.g. usage with plain IPFIX.

As seen on Fig. 2, the computer aggregating the network protocol data before the — ideally cryptographically secured — transmission is non-invasive to the core business of an ISP and can be fed with possibly filtered data coming from internal processes which are intended for data forwarding in other formats.

Our main input at the moment are raw PCAP files as well as direct network links, but structurally any well documented format (see Fig. 1) containing as much protocol data as possible is well suited for conversion.

3.1 Descriptors

The Descriptors are a subset of the transferable information, resulting in a big reduction of data by summation of occurrences of protocol data fields. The definition of a descriptor can be presented in a more exact manner over a larger set of formal languages each describing a unique data field in a network protocol. Our set of values we want to transmit for further analysis is the data describing the network flow and a set of Descriptors defined as

$$\left\{ Descriptor(i) = \sum_{j=1}^{Y} V_j \right\}_{i=1}^{X} \tag{3}$$

where

$$V \in \Sigma_F, \Sigma_F = \text{ values of protocol field } i, \forall j \in \text{network packets} \tag{4}$$

Most protocol fields are single bit flags, so $\Sigma_F = \{0, 1\}$, but byte counters are also supported in the same fashion.

The main focus of Eq. 3 is the difference of the running indices, building each sum over the network packets with index j and the whole set over index i for each value of one network protocol field for all fields under aggregation. X is bounded by the number of network protocols which can be aggregated into Descriptors.

The operations on descriptors are the arithmetic addition $+$, so they are commutative by construction. It is practical to compute the descriptors in parallel, but special fields have to be excluded for this. Addition trees are a common optimization possible by this choice as well as heterogeneous computation by distribution to different computational units. This allows offloading onto General Purpose Graphics Processing Units (GPGPUs) to speed up computation.

For practical reasons the summation intervals are bounded. Common boundaries are network flows and time intervals. If a Descriptor is 0, it is omitted from the set of transferable data, so the set will be sparsely populated. Still, the sparse set of Descriptors is a map with many small values, resulting in a high number of 0x00-bytes in the data structure. This property allows for further compression of the data, as we will see.

3.2 Implementation

Our prototype was implemented during the course of a masters thesis and is in use locally and on-site at industry partners.

We define a network flow as unidirectional, since each bidirectional network flow can easily be recombined from the two complementary unidirectional flows. Using this approach, it is possible to identify malicious parties even using only aggregate protocol data.

Our implementation is a prototype based on a cost-benefit analysis. There were several ways to implement the count system for flows. The decision was given by the main requirements for a modular, maintainable, scalable system and a meaningful data caching. The system is in use locally and on-site at industry partners. There are different possibilities for recording and handling the flow data. The most important variant for testing is writing data into binary files. A tool for editing and implementation of privacy policy was also created, containing a buffered parser. It can print out datasets and a content summary, anonymize IP addresses and reset timestamps or label datasets.

The network protocol aware parts of the code-base have been split out into plug-ins for simple extensibility. These are also the most computationally critical parts of the implementation, limiting the aggregatable bandwidth. Currently over 50 plug-ins are implemented. Most of them represent a whole protocol aggregation and other ones search for defined incidents. If a new plug-in is needed it can be easily added to the system. It is also possible to define different aggregation processes that use only selected plug-ins.

A time memory trade-off in aggregation of data over a certain time interval in each flow results in better compression at the cost of more time in computation

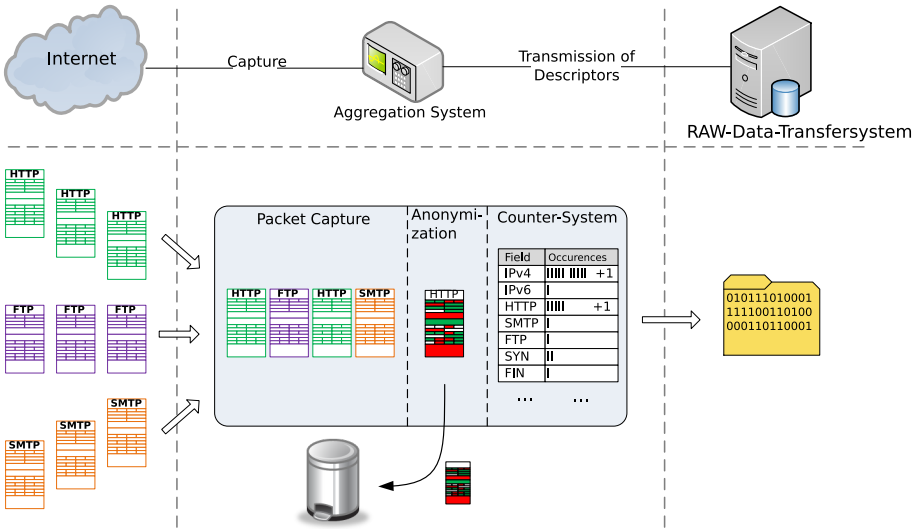


Fig. 3. Implementation of the aggregation system and process

as well as necessary memory. To offset this trade-off, we use the independence of different network flows as well as the commutative quality of Descriptors to offload their generation onto different threads. Offloading onto GPGPUs would most certainly be beneficial due to the embarrassingly parallel nature of the computations, but the bandwidth of the memory bus is a limiting factor for high data rates, since the data is moved from a source — e.g. a network interface card — into the main memory, switching at least once to a GPGPU and again at least once back into the main memory, where the results reside ultimately. Another practical limit is thus the main memory bus transfer rate.

Computationally the emergence of cheaper parallel processing units makes a strong case for some form of compression before the transmission of captured network flow data. This is due to the fact that available processing power grew faster than the ratio between available bandwidth at research laboratories compared to that at ISPs.

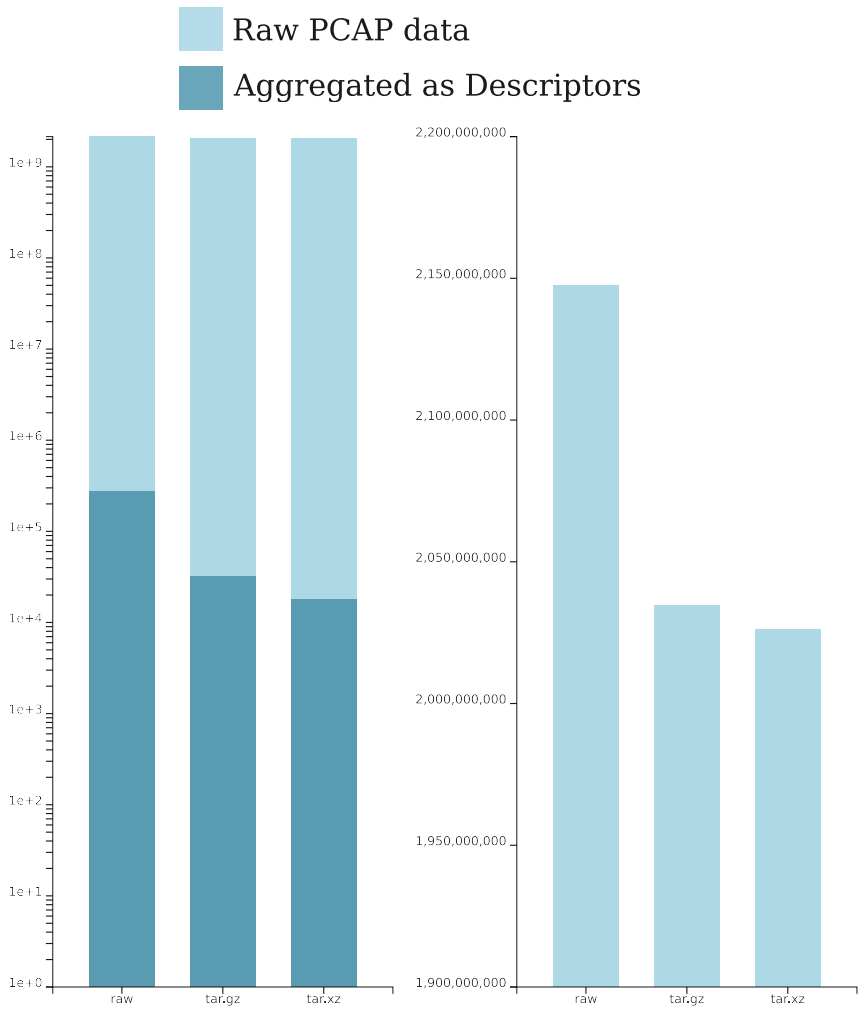
4 Analysis and Results

For an effective measurement of the data reduction ability of Descriptors for network protocol data transfer we took an exemplary PCAP file with some very large network flows and a large number of very small ones.

Intuitively the larger network flows should be easier to reduce in size, since most protocol header data should be recurring and thus easy to aggregate in one Descriptor. In contrast to this many small network flows make the overheads of our format more visible, as Descriptors will tend to have only small number of packets for aggregation.

Format	Raw data size	.tar.gz	.tar.xz
PCAP-testfile	2147483647 B	2034619265 B	2026200592 B
relative size	100 %	94.74 %	94.35 %
Flow-subversion5	274056 B	32067 B	17832 B
relative size	100 %	11.70 %	6.51 %
relative to full raw data	0.0128 %	0.00165 %	0.0009 %
compression factor 1/	7835.93	63449.01	113627.22

Table 1. Data aggregation and compression



(a) logarithmic view

(b) absolute view, crop

Fig. 4. Compression results compared logarithmically, to the right are the small details of the top in absolute values

Numerically, our results are some orders of magnitude better than bulk compression of the raw data, but of course we omit network packet data which carry no protocol information, so the network flows get trimmed of their big payloads. This is not the only size reduction, since our format allows for better compression factors and can deliver interesting data during protocol handshakes even if the later payloads will be encrypted, which we expect to happen more routinely in the future.

Nonetheless, if we set the magnitudes in data size reduction into relation (again see Fig. 4) to the practicability of having the network protocol data or missing them entirely because of bandwidth saturation, we see huge benefits in our approach.

Please note the omission of large parts in the scale numbering in Fig. 4(a) to make the small differences from bulk file data compression more visible, since in Fig. 4(b) their differences are hardly visible at all on the logarithmic scale.

5 Problems encountered

Our prototype does not honor packet ordering constraints and detects new flows by transport layer properties: If TCP or UDP, group flows by IP-addresses, network ports used and network interface source ID. The ordering of network packets in each flow is not important due to the nature of Descriptors. More agrregation protocols can be implemented via further protocol plug-ins and the percentage of network traffic with no implementation of Descriptor aggregating plug-ins can be measured as the difference between the aggregated packet count and the unaggregated one.

The bottleneck in our implementation is the copying of network data during aggregation into Descriptors, so the compression on multi gigabit per second Internet links is not in real-time.

Since our prototype only works on unidirectional network flows and not directly on bidirectional ones, connection tracking is harder to implement.

6 Conclusion

We presented an efficient method to effectively reduce the bandwidth needed for transferring network flow protocol data for later analysis in NAD. Our approach can be used as an addition to standard ISP operating procedures producing flow data in different formats as long as these formats also contain higher protocol level data. Our aggregation of network protocol data is efficiently parallelizable and can be computed in a data pipeline fashion at high data rates on standard computing hardware. Since the aggregated data are grouped by network flow membership, single flows containing malicious activity can be detected by higher level analysis. Providing rich protocol data for researchers on slower Internet links allows analyzing network activity on the protocol level at larger ISPs for malicious activity.

Our approach compares favorably to common bulk data file compression formats and is itself a good pre-filter before their application, resulting in further compression at higher factors than on raw PCAP data. Using our approach it is much more practicable to pass full network protocol data through to researcher laboratories for NAD.

6.1 Lessons Learned

Parallelization for the aggregation of data is not effective if not planned from the beginning on, so the bottleneck for getting data from large ISPs takes place in this phase. Our next development will be optimized in its performance for bandwidth scalability from the ground up, so our solution can be used to make data acquisition from most sources viable even for smaller research organizations.

A minor unwieldiness is the implementation of aggregation plug-ins for different network protocols, since each Descriptor has to be selected by a developer with semantic knowledge of the protocol on implementation. This is mostly offset by targeting most major protocols in use on the Internet and measuring the difference between those implemented by Descriptors and the small percentage left over.

7 Future Work

Our main focus is re-engineering and optimizing the parallel aggregation pipeline to achieve optimal throughput in the order of 10 gigabit per second utilizing GPGPU if available.

More network protocol plug-ins will be developed to generate more Descriptors, giving an even more complete view of network protocol activity.

We will implement an IPFIX connector to get our input from IPFIX records. After pre-filtering of bulk data, our aim is to build efficient and therefore scalable network malware detection systems.

References

1. Ahmad, R., Ghani, M., S.H.C.Haris, Waleed, G.M.: Anomaly detection of ip header threats (2012)
2. Marina Bykova, Shawn Ostermann, B.T.: Detecting network intrusions via a statistical analysis of network packet characteristics (2001)
3. Deri, L., Maselli, G., Suin, S.: Design and implementation of an anomaly detection system: an empirical approach (2003)
4. Rossow, C., Dietrich, C.J., Bos, H., Cavallaro, L., van Steen, M., Freiling, F.C., Pohlmann, N.: Sandnet: Network traffic analysis of malicious software. In: Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, ACM (2011) 78–88
5. Lakhina, A., Crovella, M., Diot, C.: Characterization of network-wide anomalies in traffic flows. In: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, ACM (2004) 201–206

6. Bilge, L., Kirda, E., Kruegel, C., Balduzzi, M.: Exposure: Finding malicious domains using passive dns analysis. In: Proceedings of NDSS. (2011)
7. Bilge, L., Balzarotti, D., Robertson, W., Kirda, E., Kruegel, C.: Disclosure: detecting botnet command and control servers through large-scale netflow analysis. In: Proceedings of the 28th Annual Computer Security Applications Conference, ACM (2012) 129–138
8. Tegeler, F., Fu, X., Vigna, G., Krügel, C.: Finding bots in network traffic without deep packet inspection. In: The 8th ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT 2012), Nice, France (December 2012)
9. Claise, E.B.: Cisco systems netflow services export version 9 (2004)
10. Sebastian Abt, Sascha Wener, H.B.: Performance evaluation of classification and feature selection algorithms for netflow-based protocol recognition. In: INFORMATIK 2013, Gesellschaft für Informatik (2013) 2184–2197
11. Network Working Group, Trammell, B.: Bidirectional Flow Export Using IP Flow Information Export (IPFIX). `http://www.ietf.org/rfc/rfc5103.txt`
12. Petersen, D., Himmelsbach, K., Bastke, S., Pohlmann, N.: Measuring and warning (2008)