Hardware precomputation of entropy for anomaly detection

Václav Bartoš, Martin Žádník
Brno University of Technology
{ibartosv,izadnik}@fit.vutbr.cz

Categories and Subject Descriptors

E.4 [**CODING AND INFORMATION THEORY**]: Data compaction and compression

General Terms

Algorithms, Design, Performance

Keywords

entropy, acceleration, network, anomaly

1. INTRODUCTION

With constantly increasing network speeds, the volume of flow data exported from measuring points is also increasing rapidly. Even a collecting and storing this data into a database causes a heavy load on a collector which may become overloaded and unable to perform advanced data analysis such as anomaly detection.

The straight-forward approach is to use sampling but while sampled data may be sufficient for basic network management, it is not advisable to use it for anomaly detection [1].

We propose a system in which statistic values needed for anomaly detection are precomputed in exporters (using unsampled data) and sent to collector together with sampled flow data. We propose to precompute an entropy of various packet features as it is popular and common metric in modern anomaly detection algorithms [2, 3, 4].

Our previous experiments with method by Lakhina et al. [2] show that it is very useful to compute entropy of not only source and destination addresses and ports, but also of other features and mainly their combinations (e.g., entropy of unique combinations of destination address and number of packets in flow). According to our experience more than ten features and their combinations might be needed. This implies more computational and memory requirements which renders effective hardware precomputation useful.

2. DESIGN

To compute an entropy of some packet header field (dimension) in some time interval, we must first count number of occurrences of all values which appeared in the interval, i.e., to create a histogram.

One of the problems that needs to be solved is how to store such histograms. Simple array of integers can be used for dimensions with only few possible values (e.g., protocol field). For other dimensions, we need extremely large arrays (e.g., 2^{32} items for IPv4 addresses) although most of its fields

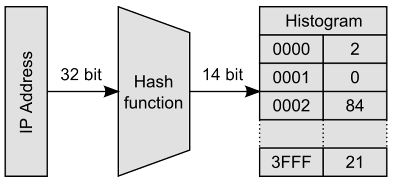

Figure 1: Hash function used to reduce the size of histogram.

are empty. Another approach is to use associative array but it is slower and more complex, especially in hardware.

We propose to use a hash function to convert values to lower data width and use small array as histogram of such thinner values (see Figure 1).During such conversion a collision may occur. Our previous research shows that if sufficient width of hash function output is used the impact on resulting entropy is negligible. The number of hash-bits needed depends on network characteristic and precision required. For example, for data from our university network, hashing IP addresses to 14 bits suffices for good precision. This implies an array of 16384 integers for histogram.

Once we have a histogram of data sample X, $hist(X) = \{n_1, n_2, ..., n_N\}$, where N is number of distinct values of X and n_i is number of occurrences of value x_i, the entropy $H(X)$ is computed as

$$H(X) = -\frac{1}{N}\sum_{i=1}^{N} n_i(\log n_i - \log N)$$

which is an equivalent formula to the well known one but more suitable for hardware computation. In this form there is only one division outside of the sum. To avoid implementation of the division in hardware it is better to send entropies as pairs $N \cdot H(X)$ and N and finalize the computation in software.

Approximation of logarithm can be implemented using table of precomputed values and linear interpolation. For example, if we want 3 decimal digits precision and maximal input value will be 2^{20}, we need a table with 11264 11-bit values (121 Kbit ROM).

A suitable platform for implementation of this system is FlowMon [5] – a hardware probe based on COMBOv2 card with powerful FPGA chip. Its purpose is to monitor network traffic and export flow records but since it is based on FPGA, it can be easily extended to perform entropy precomputation as well.

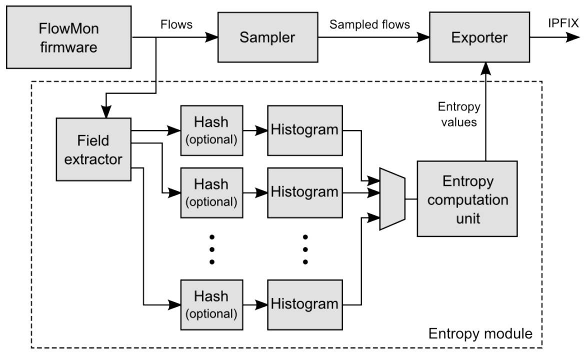

Figure 2: Extension of FlowMon probe with module for entropy computation.

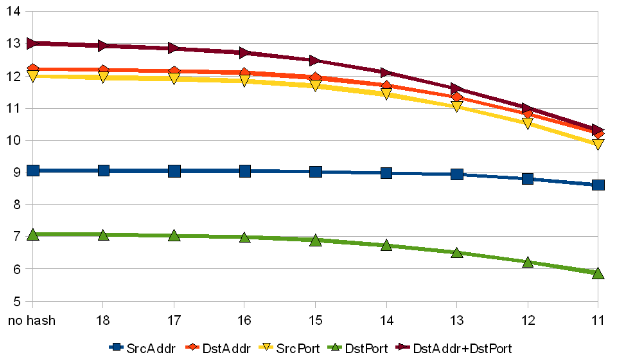

Figure 3: Example of entropy values for different width of hash function output.

3. ARCHITECTURE

An architecture of the proposed system is depicted in Figure 2.Firmware of FlowMon probe generates flow records which can be sampled and then exported in NetFlow or IPFIX format to a collector. But all flow records go also to an entropy unit.

In the entropy unit, relevant fields are extracted and combination of fields are formed. Resulting vectors are sent to units implementing a hash function which convert them to lower data width. Results are then used as addresses to memories (histograms) where addressed fields are incremented by one.

After a predefined interval, histogram values are regularly sent to the entropy unit. Once the entropy is computed for all dimensions it is sent to the exporter which adds it to the flow records transferred to a collector.

Please note that collecting and erasing values from histograms may take some time but it may run concurrently with the accounting of new values in the histograms. This may cause a small bias in obtained histogram but that is negligible. The time interval during which the histogram is built is usually one or more minutes while interval for collecting these values is under 10 ms.

4. EVALUATION

For each measured dimension, we must first determine optimal width of hash function output. It is a trade-off between memory requirements and precision – each bit doubles the size of corresponding histogram but too few bits cause large error. On Figure 3, there is an example of entropy values for different hash widths for selected dimensions. The leftmost values are real entropy computed without hash functions.

Decrease of absolute values of entropies by few percents is not a problem because relative changes in entropy caused by anomalies still remains significant. Most dimensions start to decrease quickly at 15 bits therefore we select this value as a good choice for these dimensions. For source address only 13 bits are sufficient.

As an example, let us suppose that 10 dimensions (features and their combinations) are measured. The network traffic exhibits such characteristics, that five of these dimensions can be hashed into 15 bits, three need 14 bits and two suffice with 13 bits. Let maximal number of flows in the time interval is 2^{20}. Size of memory required to store histograms in this case is 4.375 Mbit ($5 \cdot 2^{15} + 3 \cdot 2^{14} + 2 \cdot 2^{13}$ 20-bit integers). Together with table of precomputed values of logarithm, total memory requirements are 4.5 Mbit which fits into moderately sized Virtex 5 FPGA with total BlockRAM capacity 6912 Kbit (155LX). According to preliminary results of our implementation, the hash functions (Jenkins hash is used), entropy computation unit and control logic requires approximately 3500 LUT Flip flop pairs of Virtex5 FPGA.

5. CONCLUSION AND FUTURE WORK

Proposed architecture allows to use flow sampling to reduce collector load, while entropy for anomaly detection is computed from unsampled data. Even for high-speed networks, the amount of required memory and precision of results are acceptable.

Memory requirements of histograms can be further reduced by technique called counter braids [6] which can compress arrays of counters significantly. This will be subject of our future research.

Acknowledgment

This work has been partially supported by the Research Plan MSM 0021630528, IT4Innovations Centre of Excellence project CZ.1.05/1.1.00/02.0070, the grant BUT FIT-S-11-1.

6. REFERENCES

[1] J. Mai et al. Is sampled data sufficient for anomaly detection? In *Proc. of the 6th ACM SIGCOMM conference on Internet measurement*, pages 165–176, New York, NY, USA, 2006.

[2] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. *ACM SIGCOMM Comp. Com. Rev.*, 35(4), Oct. 2005.

[3] C.-K. Han and H.-K. Choi. Effective discovery of attacks using entropy of packet dynamics. *IEEE Network*, 23(5):4–12, 2009.

[4] K. Xu, Z.-L. Zhang, S. Bhattacharyya. Profiling internet backbone traffic: Behavior models and applications. In *ACM SIGCOMM*, Aug. 2005.

[5] M. Žádník et al. Flowmon for network monitoring. Technical report, CESNET, Praha, 2010.

[6] Y. Lu et al. Counter braids: a novel counter architecture for per-flow measurement. *SIGMETRICS Perform. Eval. Rev.*, 36:121–132, June 2008.