Detecting Network Anomalies in Backbone Networks

Christian Callegari, Stefano Giordano, Michele Pagano, and Teresa Pepe
Dept. of Information Engineering, University of Pisa, ITALY
E-mail: {firstname.lastname}@iet.unipi.it

1 Introduction

Uncovering anomalies in large ISPs and enterprise networks is challenging because of the wide variety of such anomalies. They can come from activity with malicious intentions (e.g., scanning, DDoS, prefix hijacking), or from misconfigurations and failures of network components (e.g., link failures, routing problems, outages in measurement equipment). In the literature, the problem of detecting anomalies in the network traffic has often been seen as equivalent to the problem of detecting heavy changes (HCs) in some traffic descriptors. In this context a wide variety of approaches has been proposed.

Nevertheless most of them analyzes the single traffic flows, resulting to be unscalable and thus not applicable in modern backbone networks.

For this reason, in this work we have decided to analyze traffic aggregates, so as to obtain a more scalable system, and in more detail we have designed our system to work on the top of probabilistic structures, namely the sketches, that allow us to obtain a scalable real-time system (that analyzes the traffic flows after having randomly aggregated them), while simultaneously improving the detection rate of "classical" systems [1].

Give this *substrate*, our method is based on a statistical analysis of the distribution of the Heavy Hitters (HHs) [2] in the network traffic. The idea behind this approach is that the distribution of the *big* flows should change between normal and attacks period, especially in the case of DoS/DDoS attacks, network scans, and so on.

Hence, in this work we present a novel method for network anomaly detection, based on the idea of discovering HC in the distribution of the HHs in the network traffic. To assess the validity of the proposed method, we have performed an extensive experimental evaluation phase, during which our system performance have been compared to a more "classical" HC-based approach.

2 System architecture

In this section we detail the system we have implemented to detect anomalies in the network traffic. The following subsections describe the significant blocks of the proposed system.

2.1 System Input

First of all the input data are processed by a module called Data Formatting. Indeed, this module is responsible of reading the Netflow [3] traces and of transforming them in ASCII data files, by means of the Flow-Tools. The output of this first block is given by text files containing on each line an IP address and the number of bytes sent by that IP in the last time bin.

In more detail, in our implementation we have in input Netflow data, measuring the traffic gone through a given router over five minutes time-bins. Thus, this module will output a distinct file for each considered time bin; let us assume we have N distinct time bins.

Note that the modularity of the system allows great flexibility. Indeed, instead of considering the number of bytes sent by a given IP, the system administrator can easily choose of using another traffic descriptor that better allows her to detect the different attacks.

2.2 Sketch module

After the data have been correctly formatted, they are passed as inputs to the hash functions responsible for the construction of the sketch tables. In more detail, for each line of each file, the IP address is considered as the key i_t, as described in Section II.A, while the number of bytes is considered as the weight c_t. Each file, corresponding to a time bin, is thus used to build a distinct sketch table.

Note that in our implementation we have used $d = 16$ distinct hash functions, which give output in the interval $[0; w-1]$, that means that the resulting sketch tables will be $\in \mathbb{N}_{d\times w}$, where w can be varied. As far as the the hash functions are concerned, we have used 4-universal hashes, obtained as:

$$h(x) = \sum_{i=0}^{3} a_i \cdot x^i \; mod \; p \; mod \; w \tag{1}$$

where the coefficients a_i are randomly chosen in the set $[0, p-1]$ and p is a random prime number (we have considered the Mersenne numbers).

At this point, given that we had N distinct time bins, we have obtained N distinct sketch tables $T^n_{d\times w}$, where $n \in [1,N]$ is the time bin.

2.3 Detection Phase

Basically, the method tracks the variations in the HH distribution of the network traffic.

In more detail, the sketch table T is given in input to two distinct modules, namely a forecast module and a HH matrix construction module.

The forecast module takes in input the sketch table T^{n-1} and its own output at the previous step, and uses these two elements for forecasting the value of the next sketch table. This prediction is performed by using an Exponentially Weighted Moving Average (EWMA) forecast algorithm, described by the following equation:

$$\widehat{T^n} = \alpha T^{n-1} + (1-\alpha)\widehat{T^{n-1}} \tag{2}$$

where $\alpha \in [0,1]$ is a tunable parameter of the algorithm.

Given this step, the system has two distinct values for the sketch table at time bin n, the real value T^n and the predicted value $\widehat{T^n}$. Both these tables are feed to a module, responsible for computing an *empirical* distribution of the HHs.

This "distribution" is computed by evaluating the HHs present in the traffic, that is the traffic aggregates (namely the sketch buckets) that exceed a given threshold, given by a percentage of the total traffic. The related buckets are then updating by inserting the quantity of traffic for which that aggregate exceed the threshold, while all the other buckets are set to one byte.

This matrix is named M^n_{HH} if computed starting from T^n and $\widehat{M^n_{HH}}$ if calculated starting from $\widehat{T^n}$.

Given these two matrices, the system compares the actual HH *distribution* in M_{HH} with the forecasted one in $\widehat{M}_{HH}$. To perform such task the system computes the Jane-Shannon Divergence (JSD) between each line of the two matrices, where the JSD between two generic vectors P and Q is defined as:

$$JSD(P,Q) = \frac{1}{2} \cdot KL(P,M) + \frac{1}{2} \cdot KL(M,Q) \tag{3}$$

where KL is the Kullback-Leibler divergence, defined as

$$KL(P,Q) = \sum_i p_i \cdot log(p_i/q_i) \tag{4}$$

and M is the "average" array of P and Q, that is $m_i = (p_i + q_i)/2$.

If such distance exceeds a given threshold for more than H lines of the matrix, the system reveals an anomalous time bin and the anomaly is thus identified.

3 Experimental results

The proposed system has been tested using a publicly available data-set, composed of traffic traces collected in the Abilene/Internet2 Network [4], a hybrid optical and packet network used by the U.S. research and education community.

The used traces consist of the traffic related to nine distinct routers, collected in one week, and are organized into 2016 files, each one containing data about five minutes of traffic (netflow data). To be noted that the last 11 bits of the IP addresses are anonymized for privacy reasons; nevertheless we have more than 220000 distinct IP addresses.

Moreover we have synthetically added some anomalies in the data, so as to be able to correctly interpret the offered results.

Tables 1 and 2 respectively report the results achieved by our proposed system and by the "classical" HC-based system [5]. Note that the tables have been obtained

Threshold	**Total Anomalies**	**Synthetic Anomalies**
th_1	310	154
th_2	256	152
th_3	199	148
th_4	179	144
th_5	172	142
th_6	167	137
th_7	163	133
th_8	151	123
th_9	135	111
th_{10}	115	94
th_{11}	99	80
th_{12}	80	31

Table 1: Experimental Results: our method

Threshold	**Total Anomalies**	**Synthetic Anomalies**
th_a	1969	154
th_b	1920	48
th_c	1381	28
th_d	1269	23

Table 2: Experimental Results: HC-based method

varying the values of the threshold. The real values of such threshold are not reported since are not significant in themselves, just consider that the first values (namely th_1 and th_a) correspond to the highest threshold value for which the two systems detect all the 154 synthetic anomalies.

From the tables we can see that both the systems are able to obtain a 100% detection rate (revealing all the 154 synthetic anomalies). Nevertheless, the two systems present very different performance, indeed our system, when detecting all the synthetic anomalies, only detects 156 more anomalies (see table 1). In this case, after a manual verification of the data set, we can conclude that most of them are real anomalies already present in the traces. Note that, in any case, event though all of these would not be "real" anomalies they'd correspond to a maximum false alarm rate of 8.3% that could be considered as "acceptable".

Regarding the HC-based method, instead, we can easily see (Table 2) that for detecting all the synthetic anomalies, we have to accept a total number of detection equal to 1969, which is not acceptable.

Moreover we can easily notice, by comparing the two tables, that our system is much easier to tune, indeed the number of total/synthetic anomalies varies quite uniformly when changing the value of the threshold. On the other hand, in the HC-based method the number of detected synthetic anomalies suddenly decreases when increasing the threshold, while the number of total detected anomalies remains quite stable.

References

[1] C. Callegari, L. Gazzarrini, S. Giordano, M. Pagano, and T. Pepe, “When randomness improves the anomaly detection performance,” in *Proceedings of 3rd International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL)*, 2010.

[2] M. Charikar, K. Chen, and M. Farach-colton, “Finding frequent items in data streams,” in *Proc. VLDB Endow.*, pp. 693–703, 2002.

[3] B. Claise, “Cisco Systems NetFlow Services Export Version 9.” RFC 3954 (Informational), Oct. 2004.

[4] “The Internet2 Network.” http://www.internet2.edu/network/.

[5] B. K. Subhabrata, E. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, “Sketch-based change detection: Methods, evaluation, and applications,” in *In Internet Measurement Conference*, pp. 234–247, 2003.