Network Anomaly Detection: Comparison and Real-time Issues

Václav Bartoš, Martin Žádník

Faculty of Information Technology
Brno University of Technology
Božetěchova 2, Brno, Czech Republic
(ibartosv,izadnik)@fit.vutbr.cz

Abstract. This paper discusses several issues of evaluation and comparison of anomaly detection algorithms, namely lack of publicly available implementations and annotated data sets. Another problem of many methods is a detection delay caused by operating on data binned to a long time intervals. The paper presents a library under development which aims to tackle the comparison and evaluation issues. Further, the paper proposes a novel anomaly detection approach that can contribute to anomaly detection in real-time.

1 Introduction

Anomaly detection systems helps to keep track of security violations and disruptions to network and its services. Methods of anomaly detection have received great research attention and large number of methods have been proposed. This paper addresses some of the issues in the area of anomaly detection (AD).

There is a lack of comprehensive comparison of existing methods and a lack of annotated data sets for their evaluation. Therefore a library of AD algorithms is proposed and being implemented. Its goal is to provide reference implementations of AD methods and to annotate available data sets in a semi-automated manner. At the same time various methods may be compared in terms of their precision and types of detected anomalies.

Further, the attention is focused on real-time anomaly detection. A novel method of anomaly detection is proposed. The method is based on observing flow cache behavior, namely, detection of its hit ratio deviations.

2 Problematic comparison of anomaly detection methods

There are many methods proposed for anomaly detection systems. But they usually vary in their properties. Some of them are specialized for detection of particular category of intrusions, others are general. Some work upon short time scales and detect anomalies within seconds, some can detect only a long-term anomalies. Each method generates certain ratio of false alarms and misses certain

portion of existing anomalies. There are also great differences in computational complexity.

Although several papers [3, 4] categorize various methods according to their properties, there is a lack of experimental evaluation and direct comparison of their detection capabilities. Such evaluation is usually done in papers proposing new methods, where authors compare proposed method with one or two previous ones. But this may not be sufficient for a designer who wants to build an anomaly detection system and needs to choose methods to use. A thorough comparison with other methods is rendered hard due to missing library of AD algorithms which would be publicly available. Moreover, many papers do not provide specific details which may turn out to be crucial for correct implementation and setup of the algorithm. Therefore it would be very useful if authors of the methods publish also its reference implementations. But this is not usual.

Another problem of evaluation and comparison of methods is a lack of testing data. There are only few public sources of real network data (MAWI traffic archive [1] seems to be the most popular). Researchers often use their own data (e.g. from a university network) but these data cannot be usually published due to privacy or security issues. The data set should be annotated, i.e. all anomalies in the data set should be labeled. But available data sets are only a plain data[1], so researchers have to annotate them themselves. These problems may lead to a biased results of the same method implemented by various researchers.

Because of the reasons discussed above, we propose and currently implement a library containing reference implementations of several anomaly detection methods. The library will create a basis for a framework allowing intuitive evaluation and comparison with other methods. Besides the anomaly detectors, the framework will contain several supporting tools, for example software flow generator, modules for loading data (packet traces, flow records) in various formats, modules for plotting results, etc. There will also be several data sets with different levels of abstraction – packet traces, flow records and traffic volume/entropy timeseries. These data will be annotated automatically (by combining data from several detectors) or semi-automatically (with manual checking and labeling of anomalies found by detectors). Of course, details of all methods used for annotation and values of their parameters will be provided.

Such library allows for an objective comparison of methods by using the reference implementations and the reference data. It will also help authors to easily compare their new algorithm with others and designers to select a fitting set of algorithms for their particular system. Most of the library is being written in Python as it allows for fast prototyping, readable and well-documented code, although several time demanding parts are being written more effectively in C++. The library is being developed within research group ANT@FIT and its first version will be published in near future on group's web site[2].

[1] There exist annotated data sets from DARPA Intrusion Detection Evaluation 1998 and 1999, and KDD Cup 1999 data set, evidently these are very outdated now.

[2] `http://merlin.fit.vutbr.cz/ant/`

3 Flow-cache based anomaly detection

Most of the anomaly detection methods work upon traffic volumes such as number of bytes, packets, flows or upon some information extracted from flow records. The prevailing source of such data are flow exporters. The flow exporters (either a dedicated probe or router) report on observed flows via NetFlow or IPFIX protocol to a collector. The collector stores reported flows into files or database. Collected flow records are then subject to the anomaly detection.

The above measurement pipeline introduces several obstacles which renders anomaly detection delayed and unreliable. First, there are delays or even a reporting losses at the exporter side. The exporter contains a flow cache which stores records describing ongoing active flows. The record is removed from the flow cache if the corresponding flow becomes inactive or if the flow cache is full. A removed flow record is queued in the exporter and awaits assembly of a reporting packet which consists of multiple flow records.

In case of a sudden flood of new flows (anomaly), there may be necessary to remove large number of flows from the flow cache. This may result in a loss of reported flow records due to the lack of performance allocated for reporting removed flow records. There can also be a delay or loss at the collector side.

Another delay is introduced by the collecting engine or the analysis engine. Former or latter gathers flow records into fixed-size time bins (e.g. 1 or 5 minutes) which are convenient for the analysis [2] but renders real-time detection impossible.

We propose a fast detection mechanism which do not rely on this measurement pipeline. It is based directly on observing behavior of the flow cache. Several statistics can be measured upon the flow cache. One of them is flow cache hit ratio – number of packets whose flow record is found in the cache (cache hit) divided by total number of packets received.

According to preliminary experiments, the hit ratio is very stable under normal conditions. But it changes significantly during flood anomalies (DoS attacks, port scans, etc.), which makes it a suitable metric for anomaly detection.

Figure 1 shows the hit ratio of a flow cache sampled every second using 4 different packet traces from MAWI archive [1]. The graphs show that the hit ratio stays within 0.8 and 0.95 most of the time except for a few sharp spikes. Little investigation reveals that all these spikes are caused by network scans or similar malicious activity. This example shows that it is possible to detect at least some network anomalies only by measuring the hit ratio of a flow cache.

Much more experiments must be conducted to find out real capabilities of this approach. But even if it does not outperform other methods in terms of detection sensitivity, it has one important advantage – it can operate on very short time scales (1 second or even less). Inherently, it can detect anomalies much faster than methods processing flow records on a collector.

As a result it may raise alert immediately. Also, it should be possible to derive a filtering rule describing flood traffic. Then only an aggregated information about the flood may be sent to the collector (e.g. via IPFIX) in order to reduce performance and bandwidth usage.

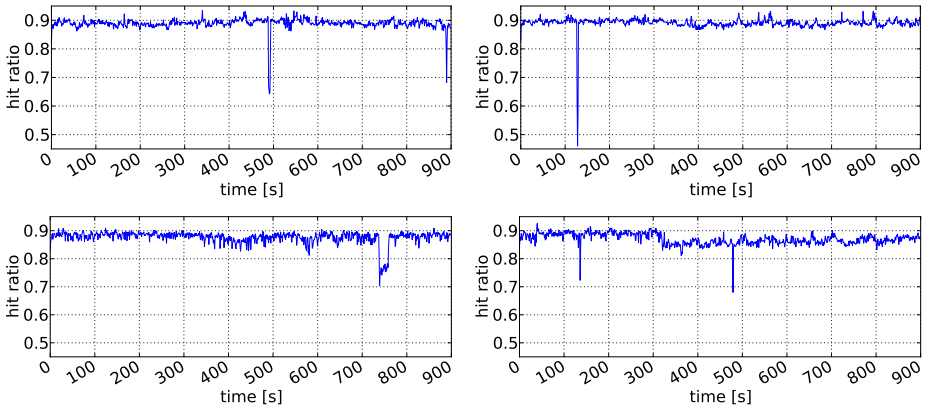

Fig. 1. Hit ratio of a flow cache for 4 different packet traces.

4 Conclusion

This paper describes author's work in the field of network anomaly detection. First, issues of evaluation and comparison of anomaly detection methods were discussed. These issues are addressed by a library which aims to cope these problems byroviding publicly available implementations of several anomaly detection methods, supporting tools and annotated data sets. Second part of the paper was devoted to description of novel method using a flow cache statistics for anomaly detection. The preliminary experiments shows promising results but more detailed investigation of flow cache properties and its usability for an anomaly detection is needed. It will be part of our future work.

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 and the grant BUT FIT-S-11-1.

References

1. Mawi traffic archive, `http://mawi.wide.ad.jp/mawi/`
2. Estan, C., Keys, K., Moore, D., Varghese, G.: Building a better netflow. SIGCOMM Computer Communication Review 34, 245–256 (August 2004)
3. Patcha, A., Park, J.M.: An overview of anomaly detection techniques: Existing solutions and latest technological trends. Computer Networks 51 (August 2007)
4. Zhang, W., Yang, Q., Geng, Y.: A survey of anomaly detection methods in networks. In: International Symposium on Computer Network and Multimedia Technology, CNMT 2009. (January 2009)