Preserving Relations in Parallel Flow Data Processing

Tomáš Čejka[1(✉)] and Martin Žádník[2]

[1] FIT, CTU in Prague, Prague, Czech Republic
cejkato2@fit.cvut.cz
[2] CESNET, a.l.e., Prague, Czech Republic
zadnik@cesnet.cz

Abstract. Network monitoring produces high volume of data that must be analyzed ideally in near real-time to support network security operations. It is possible to process the data using Big Data frameworks, however, such approach requires adaptation or complete redesign of processing tools to get the same results. This paper elaborates on a parallel processing based on splitting a stream of flow records. The goal is to create subsets of traffic that contain enough information for parallel anomaly detection. The paper describes a methodology based on so called witnesses that helps to scale up without any need to modify existing algorithms.

1 Introduction

Common architecture of monitoring large networks contains multiple observation points measured by monitoring probes and a central collector with captured data. This approach creates a global view of the network traffic. In addition, it allows for analysis and detection of global events that are less visible from a local view.

This approach works well on small networks, however, since the network traffic grows, processing all data on one place reaches limits of resources such as memory capacity. In addition, various network events produce data that reach maximal performance of a single machine. Altogether, network monitoring becomes a Big Data processing and some scalable approach must be considered.

As it is described later in Sect. 2, Big Data principles are being studied for last years. However, a general methodology of splitting network data into subsets is missing. The aim of this work is to describe a principle how to split data with respect to internal relations and used processing algorithms in order to analyze balanced data subsets while the needed information still remains together.

The goal is to allow processing huge amounts of flow data using analysis tools that are not designed for a distributed environment. A correct selection of data subsets can improve current mechanisms of data distribution or sampling without loosing information needed by detection algorithms.

D. Tuncer et al. (Eds.): AIMS 2017, LNCS 10356, pp. 153–156, 2017.
DOI: 10.1007/978-3-319-60774-0_14

2 Related Work

MapReduce was used for network data processing e.g. in [6,7], however, the authors used a distributed database or a distributed filesystem to store data files. Paper [4] analyzes IP, TCP and HTTP traffic stored in offline files using Hadoop and MapReduce. Paper [1] analyzes campus network using several types of MapReduce jobs (e.g. measuring volume of traffic per subnet).

Authors of [5] try to use Apache Spark framework with Netmap to extract traffic features for a packet-based detection of different types of DDoS attacks in real-time. The detection uses machine learning methods. The authors rely on a distributed storage and an abstraction of objects called Resilient Distributed Dataset, however, no efficient data splitting is discussed. The paper notes that usage of sampled data produces many false-positives.

Semantic relations in data and possible negative effects of splitting data were mentioned in [3]. The authors present experiments with Hashdoop, an improved Hadoop, that splits data using CRC hashes of src and dst IPs. The authors chose a simple packet counting and ASTUTE algorithm for parallel processing. The splitting based just on IP addresses is a single case in our methodology.

3 Proposed Approach

The main requirement is to identify which parts of data must stay together to preserve data relations and which parts can be split into subsets.

A **detection algorithm** can be described as a function with data about network traffic as its input and alerts (detected events) as its output. Generally, the input data is a mixture of benign and malicious traffic. The goal of a detection algorithm is to identify the malicious traffic and to generate an alert that describes the detected event. The algorithm is successful if it observes at least a **minimal subset** of malicious traffic which triggers the alert. Lets call the instance of a minimal subset a **witness**. If a witness gets divided, the malicious traffic is not detectable with the same detection algorithm anymore because there is not enough information for decision. As a result, data can be divided for parallel processing in any way that does not break witnesses.

In practice, there are many different detection algorithms processing the same data to detect various types of malicious traffic. As a result, multiple different witnesses must be preserved at the same time which complicates data splitting.

Data can contain many witnesses that identify the same malicious traffic, while any of them is sufficient for a successful detection. In order to design a data splitter a particular type of witness should be characterized. This kind of characterization describes what data an algorithm analyzes, how the malicious traffic looks like and what is the configuration of an algorithm.

4 Evaluation

To evaluate the witness-based splitting, we analyze data distribution among computation nodes and overall detection results. We need to compare results

of a single instance and results of a distributed environment. As the distributed processing generates some alerts multiple times a deduplication based on timestamps, type of events and other information contained in the alerts is necessary.

For the evaluation, we use a NEMEA framework [2] which can be easily run in a single instance as well as in a distributed configuration. There are several detection modules in NEMEA and some of them were presented in our previous work. However, the presented principle can be used with any system that allows to modify an algorithm of data splitting.

The first experiments with splitting flow data with respect to potential witness showed that it is possible to distribute flow data almost uniformly and there is no significant difference between detection results of a single instance and the distributed environment. The measured difference was about 1% which is caused by inaccurate timing of stream-wise real-time analysis during our experiments.

5 Conclusion

This paper addressed a network traffic analysis in a distributed environment. There are many papers focusing on existing Big Data frameworks but, to our best knowledge, a general methodology of splitting a stream of flow data is missing. This research aims to describe data relations that must be preserved for the parallel analysis. The data relations, types of malicious traffic and used detection algorithms with their parameters define so called witnesses. Since this research is rather a work-in-progress, we have some preliminary results. However, the experiments with real data show that respecting witnesses allow for distributed processing without significant impact on detection results.

As a future work, we are going to explore the principle of witnesses in more detail. Moreover, based on witnesses, an algorithm of real-time reconfiguration of the splitter to scale up the distributed system would be useful.

Acknowledgments. This work was supported by the Technology Agency of the Czech Republic under No. TA04010062 *Technology for processing and analysis of network data in big data concept* and grant No. SGS17/212/OHK3/3T/18 funded by Ministry of Education, Youth and Sports of the Czech Republic.

References

1. Bumgardner, V.K., el al.: Scalable hybrid stream and Hadoop network analysis system. In: Proceedings of the 5th ACM/SPEC ICPE (2014). doi:10.1145/2568088.2568103
2. Cejka, T., et al.: NEMEA: a framework for network traffic analysis. In: Proceedings of CNSM (2016). doi:10.1109/CNSM.2016.7818417
3. Fontugne, R., et al.: Hashdoop: A MapReduce framework for network anomaly detection. In: Proceedings of INFOCOM (2014). doi:10.1109/INFCOMW.2014.6849281

4. Ibrahim, L.T., et al.: A study on improvement of internet traffic measurement and analysis using Hadoop system. In: Proceedings of ICEEI (2015). doi:10.1109/ICEEI.2015.7352545
5. Karimi, A.M., et al.: Distributed network traffic feature extraction for a real-time IDS. In: Proceedings of EIT (2016). doi:10.1109/EIT.2016.7535295
6. Lee, Y., et al.: Toward scalable internet traffic measurement and analysis with Hadoop. ACM SIGCOMM Comput. Commun. Rev. **43**(1), 5–13 (2013). doi:10.1145/2427036.2427038
7. Zhang, J., Zhang, Y., Liu, P., He, J.: A spark-based DDoS attack detection model in cloud services. In: Bao, F., Chen, L., Deng, R.H., Wang, G. (eds.) ISPEC 2016. LNCS, vol. 10060, pp. 48–64. Springer, Cham (2016). doi:10.1007/978-3-319-49151-6_4