Stream-Based IP Flow Analysis

Milan Cermak*†, Pavel Celeda*†
*Institute of Computer Science, †Faculty of Informatics
Masaryk University, Brno, Czech Republic
{cermak,celeda}@ics.muni.cz

***Abstract*—As the complexity of Internet services, transmission speed, and data volume increases, current IP flow monitoring and analysis approaches cease to be sufficient, especially within high-speed and large-scale networks. Although IP flows consist only of selected network traffic features, their processing faces high computational demands, analysis delays, and large storage requirements. To address these challenges, we propose to improve the IP flow monitoring workflow by stream-based collection and analysis of IP flows utilizing a distributed data stream processing. This approach requires changing the paradigm of IP flow data monitoring and analysis, which is the main goal of our research. We analyze distributed stream processing systems, for which we design a novel performance benchmark to determine their suitability for stream-based processing of IP flow data. We define a stream-based workflow of IP flow collection and analysis based on the benchmark results, which we also implement as a publicly available and open-source framework Stream4Flow. Furthermore, we propose new analytical methods that leverage the stream-based IP flow data processing approach and extend network monitoring and threat detection capabilities.**

***Index Terms*—Stream Processing, IP Flow, Stream4Flow**

I. Introduction

Computer networks are a fundamental part of modern IT services and play an important role in our daily work and personal life. Their failure directly affects us and has consequences to our lives, economy, and cybersecurity. Therefore, ensuring network reliability and cybersecurity is one of the key tasks of the network, services, and cloud operators. To provide network reliability and cybersecurity, operators need to have an overview of the network and status of provided services. In high-speed and large-scale networks such as backbone, enterprise, or cloud networks, IP flow monitoring and analysis is typically used to provide this overview. IP flow monitoring is based on an aggregation of packets to connection-like records, which significantly reduce the amount of data that needs to be analyzed [22]. Despite the data reduction, this approach provides sufficient insight into network traffic and can be used for anomaly and intrusion detection as well as network monitoring to ensure the quality of provided services. Nowadays, IP flow monitoring is a de facto standard of the modern network infrastructure, and the majority of network operators use it on a daily basis [27].

Recent years have brought not only improvement in network monitoring but also in data storage and processing – big data architectures. The improvement has started with the MapReduce computation concept [17] that enabled the development of scalable distributed data storage and processing systems with Hadoop [29] at the forefront. Shortly afterward, other systems and frameworks based on this concept have appeared, which also lead to the development of new network traffic data analysis methods [23, 19, 25]. In addition to the evolution of the batch-based data processing approach, the MapReduce computation concept has enabled the creation of novel distributed data stream processing systems. They emerged in response to the limited performance of conventional persistent databases, which were not designed for the rapid and continuous updates of individual data items arriving at high velocities [11]. These systems introduce an entirely new approach to processing large volumes of data and offer great potential for network traffic analysis.

The major challenges of contemporary approaches to IP flow monitoring arise from the fact that network traffic processing has become a big data problem, as the volume and speed of transmitted data continuously increase [15]. Besides, network monitoring faces demands for advanced analytical methods able to give results in the shortest possible time while providing a detailed overview of the network. Our research [1] addresses these challenges by utilization of the recent development of distributed stream processing methods and systems to enhance the capabilities of the collection and analysis part of the IP flow monitoring workflow. This enhancement requires a change not only in the workflow architecture but also in our perception of IP flow data processing.

II. Problem Statement and Research Goals

The IP flow monitoring workflow consists of several monitoring probes and one or more collectors, where all IP flows are collected and further analyzed. Most of today's collectors are still based on the design pattern proposed by Peter Haag in nfdump collector architecture [21]. Therefore, IP flow data are processed in batches, whereas their analysis takes place only after the data is stored. Due to the increasing volume and velocity of measured IP flow data, it has become computationally expensive and impractical to store and then process IP flow records [11]. Moreover, this architecture does not allow efficient analysis of IP flows in near real time, which plays an important role in automated anomaly response mechanisms [16]. Developers of IP flow monitoring systems face these challenges by increasing the hardware performance or using simple master-slave architectures. Nevertheless, the IP flow collection and analysis stays centralized, scalability is limited, and analysis time remains relatively high.

978-3-903176-32-4

We have identified three main open issues of IP flow monitoring workflow, which directly impact its capabilities of efficient network monitoring. First, the workflow must change to handle the continuously growing volume and velocity of measured IP flows. Second, service outages or intrusions need to be detected in the shortest possible time in order to avoid significant damage and financial loss [20]. Last, the IP flow analysis methods' capabilities need to reflect new network monitoring trends and face challenges raised by the rapid evolution of the cyber threat landscape.

A. Increasing Volume and Velocity of IP Flow Data

Along with the increase in volume and speed of transmitted data in today's networks, the number of IP flows that need to be analyzed grows. Therefore, the IP flow collection and analysis workflow must be capable of processing up to millions of IP flows per second to handle regular traffic as well as peaks that emerge during attacks and other anomalies. Such data processing puts high demands on computational resources and storage capacity of used systems and represents the main challenge of IP flow collection and analysis.

A possible solution to efficient IP flow data processing is the use of distributed stream processing methods and systems that, thanks to the scalable distribution of data streams within a computing cluster, allows us to process large amounts of IP flow data and respond to their future growth. There are currently several distributed stream processing systems differing in architecture, method of data processing, provided analysis methods, and programming languages. However, none of these systems are directly designed for IP flow data processing [7]. Therefore, it is necessary to assess and compare these systems' key features to determine their suitability for the IP flow collection and analysis.

B. Delays in IP Flow Collection and Analysis

Technologies and data processing approaches currently used within the IP flow monitoring workflow introduce significant delays in network traffic processing. The first delay occurs during the observation and export of IP flows, which is affected by the setting of used IP flow expiration timeouts. Another delay occurs during the collection and analysis of IP flow data. The analysis is performed in batches, and it is, therefore, necessary to wait until a batch of data has become available. This approach may cause a network traffic anomaly to be detected with a delay of up to several minutes that may be fatal when we try to reduce the harm caused by an attack or network service disruption.

To overcome the delay caused by batch-based IP flow data analysis, it is necessary to either process IP flow data in much smaller batches, which complicate the analysis and correlation of data, or process them continuously as they arrive at the collector. A redesign of the IP flow collection and analysis workflow from batch-based data processing to a new approach is needed to make this possible. In addition to the deployment of new technologies, it is necessary to adapt the way IP flow data are processed and revise current analytical approaches.

C. Limited Methods of IP Flow Data Analysis

When we started our research in 2014, the IP flow monitoring was very limited both in the visibility to the application layer information and in the capabilities of used technologies to efficiently compute and store large volumes of IP flow data. Analytical methods were often based on overall network state monitoring and in-depth overview (such as collecting information about individual network hosts and their actions) were provided only in a limited way. Other issues of IP flow data analysis were caused by the very nature of batch data processing, where, for example, aggregation over broader time windows can hide important information such as anomalies in the form of short bursts of network traffic. Therefore, it was necessary to design a new approach to IP flow data processing and analysis that will overcome these issues and reflect current analysis trends and network monitoring requirements.

D. Research Goals

The above-described problems of IP flow collection and analysis can be summarized into the following main objective of our research:

> *Research how the IP flow collection and analysis can be improved by distributed stream data processing and investigate its capabilities for advanced network monitoring and threat detection.*

Given the main objective, we divide our research into the following three consecutive research goals (RG) that are motivated by challenges of stream-based IP flow data analysis and improvements of the IP flow collection and analysis:

- *RG1:* Evaluate the suitability of distributed stream processing for analysis of IP flow data in high-speed and large-scale networks.
- *RG2:* Define an improved IP flow collection and analysis workflow based on distributed stream processing of IP flow data.
- *RG3:* Propose advanced IP flow analysis methods for high-speed and large-scale network monitoring and threat detection.

The main results of our research on these goals are presented in the next sections.

III. Distributed Data Stream Processing

Stream processing systems (historically referred to as data stream management systems [13]) emerged in response to the poor performance of traditional persistent databases. These databases were not designed for the rapid and continuous updates of individual data items continuously arriving at high velocities. Instead of storage of data in secondary memory, stream processing systems process data immediately after their arrival. During processing, the data are stored in primary memory, and only the results of individual analytical operations or the original filtered data, which are essential for further processing, are stored on persistent storage. This approach significantly eliminates the requirements for both the analytical system's storage capacity and computational resources as smaller blocks of data are processed.

A. Performance Benchmark Definition

Distributed stream processing frameworks differ in system architecture, data processing approaches, provided analysis methods, and programming languages. As a result, each of these frameworks is suited to a different type of data and analysis purpose. However, none of them was designed for IP flow data analysis, which means processing a large volume of small structured messages. To determine which system has the best performance, we proposed a novel benchmark [7] for measuring the performance of these frameworks.

Contrary to universal benchmark StreamBench, proposed by Lu et al. [24], our proposed benchmark is primarily inspired by common cybersecurity analysis methods of IP flow data [26]. Based on the exploration of these methods, we identified four basic operations that are included in the majority of them: *Filtering*, *Count*, *Aggregation* and *Top N*. We transformed these operations into standalone programs publicly available in the benchmark archive [7] to compare the tested systems' performance over the basic processing of IP flow data. These four operations are preceded by an *Identity* operation, used to determine the system's performance baseline. We have also added a *SYN DoS* (Denial of Service) operation representing an example of a real network attack detection method.

The benchmark utilizes a dataset based on real network traffic to simulate realistic computations. The dataset's basis is formed by a network traffic sample from the CAIDA dataset [14] transformed to IP flows represented in JSON format. The architecture of the benchmark corresponds to a typical pipeline of the distributed data stream processing. The input stream of data is provided by the Apache Kafka messaging system, fed by a dataset using multiple writing threads such that its processing speed exceeds the speed of the tested system. The dataset is obtained from Kafka through multiple partitions corresponding to the number of cores available in the testing environment or their multiples, which optimally utilizes the tested system. Another Kafka instance is connected to the tested system's output, but a different instance is used, so it does not affect input Kafka's speed.

B. Benchmark of Distributed Stream Processing Systems

For our research's, we have selected the three frameworks that were widespread at that time (2016) and represented the most significant potential for further development – Apache Samza, Storm, and Spark. To identify the most suitable framework, we compared the throughput of these frameworks using the proposed benchmark that we deployed on a cluster of 7 nodes. The configuration of this cluster corresponds to commonly used setup, making the benchmark more realistic. A detailed specification of the used hardware and software can be found in the thesis [1].

We performed the benchmark using four different settings of the nodes used. Figure 1 shows the benchmark results when four nodes with 4 virtual CPUs were used. The benchmark results and corresponding discussion of all testing environments (4 nodes with 8 vCPUs and 4 vCPUs, 1 node with 32 and 16 vCPUs) can be found in the thesis.

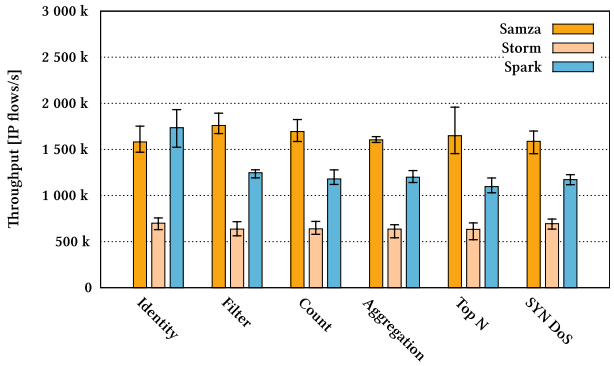

Fig. 1. Performance benchmark using 4 *vm_small* nodes (16 vCPUs in total).

The benchmark results show that each tested distributed stream processing framework can process at least 500 k IP flows/second. This result fulfills the minimal requirement of 300 k IP flows/second throughput derived from our observations to handle normal traffic and peaks, which emerge during attacks and other anomalies. Although Samza has the best throughput results, we cannot say that it is optimal for IP flow data processing and analysis due to its strict requirement for a number of data partitions corresponding to available processor cores. If partitioning cannot be performed on data before, the analysis utilizes multiple partitions, which requires a shared state, causing a throughput decrease. Thus, in selecting the best working system for IP flow analysis, the decision needs to consider the advantages and disadvantages of the system and the deployment environment.

IV. Stream-Based IP Flow Analysis

The transformation of the traditional workflow of network IP flow monitoring into a stream-based one raises new challenges and requirements that must be addressed. The stream-based IP flow data analysis approach must enable the IP flow data to be processed in a similar way as batch-based approaches. This means that it should provide at least the same basic set of data processing operations. The stream-based approach should also enable applying these operations to larger data units; thus, the window functionality is necessary to supply batch-based approaches. In addition to the supported operations, stream-based data processing must also ensure that each IP flow is processed just once to avoid skewed results. Thus, the recoverability and durability options of the data processing system should be considered too.

A. Workflow Design

A generic interconnection of the stream-based IP flow collection and analysis and common IP flow monitoring workflow is shown in Figure 2. To allow such interconnection, the collector must provide the functionality to transform IP flow records into a suitable data serialization format (DSF). Alternatively, the collector can be omitted from the workflow if the IP flow exporter can provide IP flow records in such a format. The typical format for distributed stream processing

frameworks is the JavaScript Object Notation (JSON) format, enabling to represent any data records suitably. It is also possible to utilize a more space-efficient data serialization format, such as binary JSON (BSON) or MessagePack, to avoid network overload if many IP flows are processed.

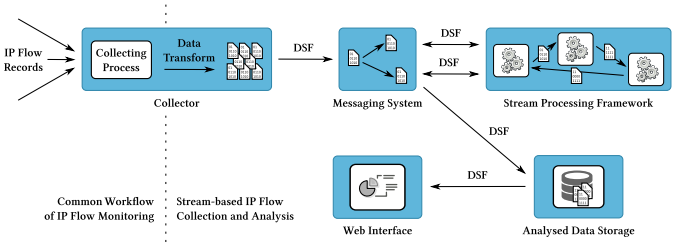

Fig. 2. Stream-based IP flow collection and analysis workflow.

B. Stream4Flow Analysis Framework

Following the definition of stream-based IP flow collection and analysis workflow, we have developed an open-source framework Stream4Flow [11]. This framework, among others, interconnects contemporary systems for IP flow data processing, provides simple administration and deployment of all components, and enables fast prototyping of stream-based IP flow analysis applications. Developed framework Stream4Flow allows to explore the advantages and disadvantages of IP flow analysis using distributed stream processing systems when deployed in a real network environment. The framework facilitates the development of new stream-based IP flow analysis algorithms, their verification using both simulated and real-world data, and experimental evaluation.

The architecture of the Stream4Flow framework reflects the workflow defined in Figure 2. Based on the systems evaluation, our experience, and consideration of requirements on IP flow analysis, we have implemented the Stream4Flow framework using the IPFIXcol collector, Kafka messaging system, Apache Spark, and Elastic Stack. The collector enables incoming IP flow records to be transformed into the JSON format provided to the Kafka messaging system. The framework's core consists of Apache Spark distributed stream processing system with custom applications for near real-time IP flow analysis. Analysis results are sent back to Kafka and stored in the Elastic Stack that also offers basic visualizations using the Kibana framework. The last part of the framework is an additional web interface capable of customized results visualization.

C. Lessons Learned

We experimentally deployed the Stream4Flow framework within the Masaryk University campus network with 24 000 active IP addresses and 12 000 IP flows/second observed on average. Thanks to this experimental deployment, we evaluated the stream-based data processing's specifics and compared existing analytical methods with new methods utilizing stream-based analysis. One of the key features that we observed during the evaluation is that we could transform the most commonly used methods for the analysis of IP flows into a stream-based approach. However, it is necessary to consider stream data processing's specific properties, which partially limits this transformation. Our experience shows that stream-based IP flows analysis cannot fully replace the common batch-based methods for network traffic analysis, but it can suitably complement and expand their capabilities.

It is important to stress that stream data processing changes the nature of the data analysis itself since the data are processed on the fly, and the analysis must be performed in a certain fashion. The batch-based approach allows to query historical data or search back through raw data for additional information after anomaly or attack detection. In stream-based IP flow analysis, the data cannot be analyzed retrospectively (i.e., perform ex-post analysis). For example, in collecting statistics on individual hosts in the network and creating their profiles, it is not possible to consider any data other than the one defined initially. Since such an ex-post analysis may be needed, we recommend combining the stream-based workflow with a suitable primary data retention store to make the optional ex-post analysis possible.

V. Network Monitoring and Threat Detection

We have evaluated the concept of stream-based IP flow analysis and Stream4Flow framework using two common use-cases of network traffic analysis: *cyber situational awareness* and *intrusion detection*. Evaluation results are provided in the thesis [1], where we introduce challenges related to use-cases and discuss the benefits of the stream-based analysis approach. The following subsections introduce new methods for large-scale and high-speed network monitoring and threat detection that follow the initial evaluation and utilize the stream-based IP flow analysis benefits.

A. Detection of DNS Traffic Anomalies

Since both the DNS query and response transferred via UDP are represented by one IP flow record, it is possible to extend the standard IP flow record by DNS application data. This information does not disrupt the IP flow record and does not excessively increase the IP flow record size. As a result, we could analyze DNS traffic with other IP flows that can reveal traffic anomalies which otherwise would only have been detectable by a deep packet inspection. Based on the manual analysis of the DNS data and common analysis methods, we identified that only four DNS packet fields are useful for the most of methods: *queried domain name*, *queried record type*, *response return code*, *response* itself, and *time to live* of the response. The other fields may unnecessarily increase the size of the IP flow record without providing additional value. Because the DNS response may contain more than one answer, we recommend storing only the first answer with the same record type as a query or authoritative name server.

One of the easiest ways to monitor and detect network traffic anomalies is a computation and analysis of Top N statistics of DNS traffic. These statistics can reveal misconfigured servers or hosts, and indicate a cybersecurity threat based on a significant statistics change. Among the most important statistics are

DNS Record Types, *DNS Response Codes*, *Queried Domains*, *Queried External DNS Resolvers*, and *Hosts With the Most Queries or Responses*. These statistics, along with the anomalies they may detect, are discussed in the thesis. In addition to the network overview, these statistics can also be incorporated into network hosts' behavior profiles, thanks to which it is possible to expand the possibilities of anomaly detection.

In addition to the statistics computation, we proposed novel detection methods focusing on malware domain queries, open resolvers, and non-local DNS resolver usage to show the advantages of combining DNS traffic information with other IP flow data. These methods are independent of the version of the IP protocol, and thus it is possible to deploy them also in IPv6 networks. In malware domain queries detection, we utilize the data combination feature of the stream-based analysis allowing us to combine IP flows data with predefined blacklist and detect malicious queries in near real time.

The main challenge of open DNS resolver detection is to distinguish an open DNS resolver from a regular one responding to local domain queries. For this purpose, the proposed method analyzes all DNS responses observed at the network edge and checks if the domain is assigned to the monitored network. This check is done by whitelisting the local domain and network address. If the result does not contain at least one record with an IP address from the monitored network or network domain, the DNS server is reported as an open DNS resolver. Our evaluation shows that this approach allows to detect an open DNS resolver as soon as it answers the first query and overcome the limitations of open scanning projects.

The crucial part of the external DNS resolver usage detection is distinguishing between a host and a local DNS resolver. Our approach utilizes the fact that the DNS resolver performs only queries, whereas the host visits the queried domain. The visit is checked by finding IP flows with communication between the host and the queried domain, which starts within approximately five seconds of the query. If the host did not visit queried domains, then it is marked as a local DNS server. This check can be performed very efficiently using stream-based analysis, allowing to combine DNS data with other IP flows within a sliding time window. Using the proposed approach, we found several hosts infected by malware that update the host settings to use malicious DNS resolvers, which returned forged IP addresses of popular web pages.

B. Near Real-Time Patterns Detection

A significant portion of deployed network cybersecurity mechanisms is based on pattern (signature) matching, where malicious traffic is identified based on an exact match with a predefined attack pattern [28]. Pattern matching detection methods ensure high accuracy but lower coverage as they can be easily evaded. A minor modification of an attack, e.g., an attack frequency, generates a new attack pattern that does not match the detection anymore. Further, the network patterns outdate quickly as network communication and attack tools evolve. Another challenge is detection delay that may cause irreversible damage, especially in critical network services.

We face these requirements by proposing a novel approach for near real-time pattern detection in IP flow data. Our pattern matching approach allows specifying various distance functions [18] and pattern definitions to enable detection of previously unknown variations of network attacks. The pattern matching approach is proposed in the context of the stream-based IP flow analysis framework Stream4Flow [11] using Apache Spark. We prototyped this approach as *PatternFinder* application that enables flexible patterns definition utilizing a set of distance functions, weights, and thresholds. Its functionality is demonstrated in the thesis on SSH dictionary attacks detection to facilitate understanding of the approach and its capabilities. We create and provide a dataset for the SSH attack pattern identification, define detection patterns for well-known attack tools, and describe the results of experimental deployment in the real-world network.

The most exciting feature of *PatternFinder* is its ability to distinguish some tools used for the attack based on provided patterns. During the concept evaluation, we were able to identify attacking tools for the third of all detected attacks, as shown in Figure 3. Besides, the evaluation showed that the application could detect not only known variants of network attacks but also their unknown variations, such as stealthy or long-lasting attacks.

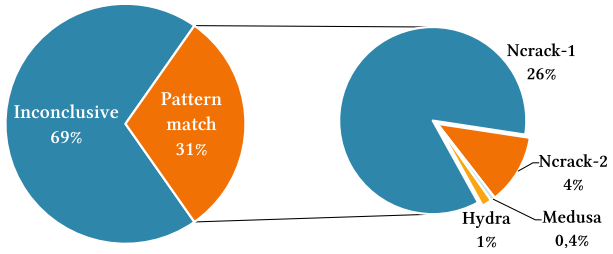

Fig. 3. Distribution of observed SSH dictionary attack tools.

VI. Conclusions

We defined the thesis's main objective to *research how the IP flow collection and analysis can be improved by distributed stream data processing and investigate its capabilities for advanced network monitoring and threat detection*. To the best of our knowledge, we were one of the first who verified the suitability of distributed data stream processing systems for IP flow data analysis and proposed integrating this data processing approach into the monitoring and analytical workflow. We also proposed new analytical methods and investigated how it is possible to use IP flows extended by application data from DNS traffic to enhance network monitoring and threat detection capabilities. Next, we have shown that these methods can utilize the benefits of stream-based processing. This approach facilitates their implementation and overcomes the limitations of batch data processing, and allows these detections to be performed in high-speed and large-scale networks in near real time.

Selected Publications

[1] Milan Cermak. "Stream-Based IP Flow Analysis". Doctoral thesis, Dissertation. Masaryk University, Faculty of Informatics, Brno, 2020. URL: https://is.muni.cz/th/tgxb6/.

[2] Milan Cermak, Martin Laštovička, and Tomáš Jirsík. "Real-time Pattern Detection in IP Flow Data using Apache Spark". In: *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 521–526. ISBN: 978-3-903176-15-7.

[3] Milan Cermak et al. "Towards Provable Network Traffic Measurement and Analysis via Semi-Labeled Trace Datasets". In: *2018 Network Traffic Measurement and Analysis Conference (TMA)*. IEEE, 2018. ISBN: 978-3-903176-09-6. DOI: 10.23919/TMA.2018.8506498.

[4] Milan Čermák and Pavel Čeleda. "Detecting Advanced Network Threats Using a Similarity Search". In: *Management and Security in the Age of Hyperconnectivity*. Munich, Germany: Springer International Publishing, 2016, pp. 137–141. ISBN: 978-3-319-39813-6. DOI: 10.1007/978-3-319-39814-3_14.

[5] Milan Čermák, Pavel Čeleda, and Jan Vykopal. "Detection of DNS Traffic Anomalies in Large Networks". In: *Advances in Communication Networking*. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 215–226. ISBN: 978-3-319-13488-8. DOI: 10.1007/978-3-319-13488-8_20.

[6] Milan Čermák, Tomáš Jirsík, and Martin Laštovička. "Real-time analysis of NetFlow data for generating network traffic statistics using Apache Spark". In: *Proceedings of the NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2016, pp. 1019–1020. ISBN: 978-1-5090-0223-8. DOI: 10.1109/NOMS.2016.7502952.

[7] Milan Čermák et al. "A Performance Benchmark for NetFlow Data Analysis on Distributed Stream Processing Systems". In: *Proceedings of the NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2016, pp. 919–924. ISBN: 978-1-5090-0223-8. DOI: 10.1109/NOMS.2016.7502926.

[8] Martin Husák et al. "Exchanging Security Events: Which and How Many Alerts Can We Aggregate?" In: *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2017, pp. 604–607. ISBN: 978-3-901882-89-0. DOI: 10.23919/INM.2017.7987340.

[9] Martin Husák et al. "HTTPS traffic analysis and client identification using passive SSL/TLS fingerprinting". In: *EURASIP Journal on Information Security* 2016.1 (2016), p. 6. ISSN: 1687-417X. DOI: 10.1186/s13635-016-0030-7.

[10] Martin Husák et al. "Network-based HTTPS Client Identification Using SSL/TLS Fingerprinting". In: *2015 10th International Conference on Availability, Reliability and Security*. Toulouse: IEEE, 2015, pp. 389–396. ISBN: 978-1-4673-6590-1. DOI: 10.1109/ARES.2015.35.

[11] Tomas Jirsik et al. "Toward Stream-Based IP Flow Analysis". In: *IEEE Communications Magazine* 55.7 (2017), pp. 70–76. ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1600972.

[12] Petr Velan et al. "A Survey of Methods for Encrypted Traffic Classification and Analysis". In: *International Journal of Network Management* 25.5 (2015), pp. 355–374. DOI: 10.1002/nem.1901.

Acknowledgment

This research was supported by ERDF "CyberSecurity, CyberCrime and Critical Information Infrastructures Center of Excellence" (No. CZ.02.1.01/0.0/0.0/16_019/0000822).

References

[13] Brian Babcock et al. "Models and Issues in Data Stream Systems". In: *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. New York, USA: Association for Computing Machinery, 2002. ISBN: 1581135076. DOI: 10.1145/543613.543615.

[14] CAIDA. *The CAIDA UCSD Anonymized Internet Traces 2015 - 20150219-130000*. 2015. URL: http://www.caida.org/data/passive/passive_2015_dataset.xml (visited on August 6, 2015).

[15] CISCO. *Cisco Annual Internet Report (2018–2023)*. Tech. rep. 2020. URL: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf.

[16] *Cyber Defense and Situational Awareness*. Vol. 62. Advances in Information Security. Springer, 2014. ISBN: 978-3-319-11390-6. DOI: 10.1007/978-3-319-11391-3.

[17] Jeffrey Dean and Sanjay Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters". In: *Commun. ACM* 51.1 (January 2008), pp. 107–113. ISSN: 0001-0782. DOI: 10.1145/1327452.1327492.

[18] Michel Marie Deza and Elena Deza. *Encyclopedia of Distances*. 3rd ed. Springer Berlin Heidelberg, 2014. DOI: 10.1007/978-3-662-44342-2.

[19] Romain Fontugne, Johan Mazel, and Kensuke Fukuda. "Hashdoop: A MapReduce framework for network anomaly detection". In: *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. April 2014, pp. 494–499. DOI: 10.1109/INFCOMW.2014.6849281.

[20] Ulrik Franke et al. "Availability of enterprise IT systems: an expert-based Bayesian framework". In: *Software Quality Journal* 20.2 (2012), pp. 369–394. DOI: 10.1007/s11219-011-9141-z.

[21] Peter Haag. *nfdump*. 2020. URL: https://github.com/phaag/nfdump (visited on January 5, 2020).

[22] Rick Hofstede et al. "Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX". In: *IEEE Communications Surveys & Tutorials* 16.4 (2014), pp. 2037–2064. ISSN: 1553-877X. DOI: 10.1109/COMST.2014.2321898.

[23] Yeonhee Lee and Youngseok Lee. "Toward Scalable Internet Traffic Measurement and Analysis with Hadoop". In: *SIGCOMM Comput. Commun. Rev.* 43.1 (January 2013), pp. 5–13. DOI: 10.1145/2427036.2427038.

[24] Ruirui Lu et al. "StreamBench: Towards Benchmarking Modern Distributed Stream Computing Frameworks". In: *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC)*. December 2014, pp. 69–78. DOI: 10.1109/UCC.2014.15.

[25] Samuel Marchal et al. "A Big Data Architecture for Large Scale Security Monitoring". In: *2014 IEEE International Congress on Big Data (BigData Congress)*. June 2014. DOI: 10.1109/BigData.Congress.2014.18.

[26] Anna Sperotto et al. "An Overview of IP Flow-Based Intrusion Detection". In: *IEEE Communications Surveys Tutorials* 12.3 (2010), pp. 343–356. DOI: 10.1109/SURV.2010.032210.00054.

[27] Jessica Steinberger et al. "Anomaly Detection and Mitigation at Internet Scale: A Survey". In: *Emerging Management Mechanisms for the Future Internet*. Vol. 7943. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 49–60. ISBN: 978-3-642-38997-9. DOI: 10.1007/978-3-642-38998-6_7.

[28] Muhammad Fahad Umer, Muhammad Sher, and Yaxin Bi. "Flow-based intrusion detection: Techniques and challenges". In: *Computers & Security* 70 (2017), pp. 238–254. ISSN: 0167-4048. DOI: 10.1016/j.cose.2017.05.009.

[29] Tom White. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., May 2009. ISBN: 0596521979.