Towards Stream-based IP Flow Analysis

Tomas Jirsik, Milan Cermak, Daniel Tovarnak, and Pavel Celeda[1]

Abstract

Analyzing IP flows is an essential part of traffic measurement for cyber security. Based on information from IP flows, it is possible to discover the majority of concurrent cyber threats in high-speed, large-scale networks. Some major prevailing challenges for IP flow analysis include, but are not limited to, analysis over a large volume of IP flows, scalability issues, and detecting cyber threats in real time. In this article, we discuss the transformation of present IP flow analysis into a stream-based approach to face the above-mentioned challenges. We examine the possible positive and negative impacts of the transformation and present examples of real-world applications, along with our recommendations. Our ongoing results show that stream-based IP flow analysis successfully meets the above-mentioned challenges and is suitable for achieving real-time network security analysis and situational awareness.

Introduction

Monitoring IP flows and their analysis play a vital role in network traffic measurements for cyber security. Currently, IP flows are broadly used for traffic measurement in large-scale, high-speed networks, cloud environments, and various enterprise networks [1]. IP flow analysis is used for detecting the majority of severe contemporary cyber threats, such as Denial-of-Service (DoS), botnets, and Advanced Persistent Threats (APT). Moreover, the analysis can be done both on unencrypted and encrypted traffic, as IP flows gather information only from packet headers. Such advantages have made IP flow monitoring a fundamental part of network traffic measurement for cyber security.

Nevertheless, IP flow analysis still faces several challenges raised by the rapid evolution of the threat landscape. First, network traffic measurement has become a Big Data problem. Due to the increasing volume and velocity of network traffic, it has become expensive and impractical to first store and then read again all IP flows from large networks for analysis. Second, it has become impossible to analyze a large volume of IP flows from networks in real time. This plays an important role for automated defense mechanisms that need to take an action as soon as possible [2]. Current approaches try to face this challenge by increasing the hardware performance of analytical machines or simple master-slave architectures. Nevertheless, the IP analysis itself stays centralized, scalability of these solutions is limited, and analysis time still remains relatively long. Last, but not least, the time needed to detect a cyber attack is a challenge for IP flow analysis. Current IP flow-based

[1] *Tomas Jirsik, Milan Cermak, Daniel Tovarnak, and Pavel Celeda are with Masaryk University*

cyber security solutions exhibit a detection delay in the order of minutes. Such a delay may be fatal when we try to reduce the harm caused by an attack [3]. Therefore, demands for near real-time attack detection have risen recently.

To address the above-mentioned challenges, we present a transformation of current IP flow monitoring and analysis into a scalable stream-based approach. In the stream-based approach, the IP flows are processed and analyzed in data streams immediately after an IP flow is observed. The analysis of IP flows in data streams reduces the volume of data that needs to be stored. This is because data is kept in primary memory for the time necessary for processing and only results are stored in the secondary memory. This represents the greatest advantage of the stream-based concept. It allows the user to perform an immediate data analysis, which makes real-time attack detection possible. Moreover, thanks to the distribution of data streams within a computing cluster, this is possible even in large-scale, high-speed networks.

In this article, we will describe the transformation of current IP flow traffic measurement and analysis towards a scalable stream-based solution. We will present a workflow of stream-based IP flow analysis, along with its prototype implementation. Capabilities of the approach will be demonstrated on cyber security use-cases followed by practical implications for its usage.

Basic Concepts

To cover the basic concepts used in this article, we provide an overview of IP flow-based network monitoring and data stream processing. This overview can be considered as a high-level abstraction of the main ideas of both areas, for a detailed description consult [1, 4, 5].

Network IP Flow Monitoring

IP flow monitoring was principally designed to monitor high-speed network traffic in large-scale networks. Since the performance limitations do not allow to process, store, and analyze all information from each packet in such networks (Deep Packet Inspection), an abstraction of single direction communication, called an IP flow, was introduced. An IP flow is defined as a set of packets passing through a point in the network during a certain time interval. All packets belonging to a particular IP flow have a set of common properties called flow keys (RFC 7011). The traditional 5-tuple of flow keys is comprised of source and destination IP address, source and destination port, and transport protocol. Apart from the traditional 5-tuple, the IP flow contains statistics about the connection (such as the number of packets in an IP flow), and may be enriched by information from the application layer of network traffic. IP flow information is stored in flow records.

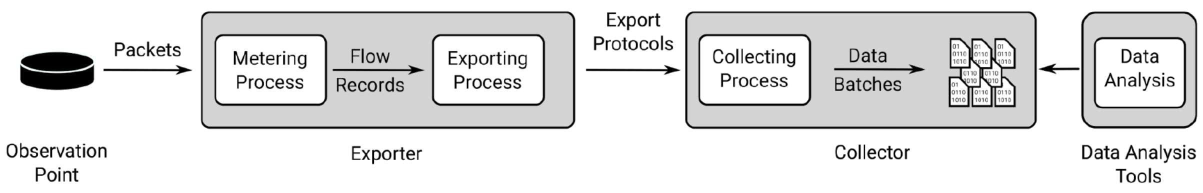

Figure 1. Traditional workflow of network IP flow monitoring and analysis.

The traditional workflow of IP flow monitoring consists of several different interconnected systems, as depicted in Fig. 1. A flow record is generated at an observation point in a network by a flow exporter. The exporter captures information from packet headers and creates flow records during the metering process. The created flow records are submitted to the exporting process to be sent to a flow collector via export protocols, such as NetFlow or IPFIX. The collector receives flow records from one or more exporters, processes them, and stores them for further analysis. The collecting process manages flow records and stores them, usually in one- to five-minutes batches into binary flat files (such as nfdump[2], SiLK[3]) or column-oriented databases (FastBit[4], Vertica[5], and so on). Row-oriented databases (such as MySQL, PostgreSQL) are not suitable for flow storage and querying due to their insufficient performance. Individual batches are then available for further data analysis.

There are three main application areas of IP flow analysis: Flow Analysis and Reporting, Threat Detection, and Performance Monitoring [1]. Flow Analysis and Reporting covers querying and filtering flow data for relevant information (network visibility), statistics overview of the data (Top N statistics, and so on), traffic accounting, reporting, and alerting (such as exceeding transfer data quota). The Threat Detection area focuses on the analysis of specific traffic events, most often scans, DoS, worms, and botnets [4]. Performance Monitoring reports the status of running services on the network by observing application metrics, such as delay, jitter, and round-trip-time.

All three application areas for IP flow analysis have one thing in common – a time aspect. As the network is monitored in time, the majority of statistics, detection methods, and performance characteristics are aggregated over a given time window (such as Top talkers in the last hour, the number of transferred bytes in the last minute). The time window is strongly influenced by the settings of the network monitoring process, namely the size of the batches in the collecting process. Data analysis can be performed only when a new batch occurs. The batch is set to five minutes in the majority of IP flow analysis tools. The analysis is then run every five minutes. This means that, for example an attack or service outage may be detected with a five-minute delay. Such delay cases that automatic defense mechanisms take action too late and protected systems are more affected. However, the demand for real-time analysis has risen recently in order to achieve shorter detection and reaction times. The analysis delay can be shortened by replacing the batch-based analysis with stream-based analysis, where each flow record is analyzed immediately as it arrives.

[2] http://nfdump.sourceforge.net/
[3] https://tools.netsa.cert.org/silk/
[4] https://sdm.lbl.gov/fastbit/
[5] http://www8.hp.com/us/en/software-solutions/advanced-sql-big-data-analytics/

Data Stream Processing

Stream processing systems (historically referred to as Data Stream Management Systems [5]) emerged in response to the poor performance of traditional persistent databases, which were not designed for the rapid and continuous updates of individual data items continuously arriving at high velocities. The key differences between traditional data processing and stream processing are summarized in Table 1.

Data stream is a possibly infinite, discrete, and ordered sequence of data elements with a given schema and assigned timestamp. Stream processing systems are designed to evaluate continuous queries over many data streams in real time, whilst predominantly using only primary memory for storage. The existing implementations of stream processing systems differ in several aspects including, but not limited to, query language capabilities, nature of processed data, time model, and so on. For example, Esper[6] is a full-fledged stream processing engine focused on evaluating continuous SQL-like queries over streams of events. For an extensive survey of stream processing systems see [6].

Nowadays, a new generation of stream processing systems is emerging that is generally referred to as distributed stream processing frameworks. These systems are used to process generic data streams and provide capabilities for distributed processing. In many cases, users must implement their own processing logic, yet they are provided with powerful abstractions that allow them to transparently execute the implemented logic in a parallel-distributed way. The most notable examples of distributed stream processing frameworks include Samza, Spark, Storm, and Flink (all maintained by Apache Software Foundation)[7].

Traditional processing	vs.	**Stream processing**
Data stored as persistent sets	**Data**	Infinite streams of individual data tuples
Large secondary memory	**Storage**	Small primary memory
Ad-hoc	**Queries**	Continuous
No real-time capabilities	**Real-time**	Real-time processing
Single-query	**Optimization**	Multi-query
Mature tools and technologies	**Maturity**	New tools and technologies

Table 1. Differences between traditional and stream data processing.

Stream-based Workflow of IP Flow Analysis

The transformation of the traditional workflow of network IP flow monitoring into the stream-based raises new challenges and requirements that must be addressed. We summarize the

[6] http://www.espertech.com/esper/
[7] http://{samza|spark|storm|flink}.apache.org/

functional and nonfunctional requirements for this transformation and describe the resulting workflow together with relevant systems. To demonstrate the possibilities of this approach, we present the Stream4Flow framework that is based on modern systems for large data processing.

Design Considerations

To successfully transform the batch-based workflow of IP flow analysis into stream-based, it is necessary to meet the same requirements as the original approach and in real time. The data processing speed plays an especially important role, but so do other requirements must be met, such as a set of available data processing operations, fault tolerance, and system durability. As regards to the minimal data processing speed of the approach, it must at least correspond to the average number of flows generated by observation points inside the monitored network. For example, in a medium sized network of 24,000 active IP addresses, we observed an average of 12,000 flows/second and 110,000 flows/second in the national wide research and education network. It can be expected that these numbers will grow in the future and, for that reason, the scalability possibilities of the stream-based processing should be also considered so that it will not be necessary to significantly change the data processing algorithms.

The stream-based approach of IP flow data analysis must enable to process the IP flow data in a similar way as traditional batch-based approaches. This means that it should provide at least the same basic set of data processing operations. Based on the common IP flow analysis algorithms, we identified the following minimal set of operations that should be provided: *filter*, *count*, *aggregation*, *combination*, *sort,* and *Top N*. The stream-based approach should also enable to apply these operations to larger units of data and, thus, the window functionality is necessary to supply traditional batch-based approaches. In addition to the available operations, stream-based data processing must also ensure that each flow was processed just once to avoid skewed results. Thus, the recoverability and durability options of data processing system should be considered too.

Workflow Design

Analyzing IP flows in real time was almost impossible in previously due to the poor performance of data processing systems. In recent years, however, a change has occurred and a number of scalable systems for fast batch-based and stream-based processing of large volumes of data were progressively introduced. In the paper [7], the authors demonstrated that distributed stream processing frameworks, such as Spark, Samza, and Storm, are able to process at least 500,000 flows/s using 16 or 32 processor cores, which is sufficient for common networks. Thanks to this, it is possible to utilize these frameworks and extend traditional workflow of IP flow monitoring and analysis. This enables to analyze IP flows in real time and provides other analytical methods that are not possible, or difficult to achieve, in common batch-based systems.

A generic interconnection of typical workflow of stream-based data processing and traditional IP flow monitoring workflow is shown in the Fig. 2. To allow such interconnection it is necessary to enable the collector 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 is able to provide flow records in such a format. The typical format for distributed stream processing frameworks is the JavaScript Object Notation (JSON) format, which enables it to suitably represent any data records. However, the JSON format is not space efficient and can cause an overloading of the network if a lot of IP flows are processed. In the case of a large amount of transmitted data, it is better to utilize a more space efficient data serialization format, such as binary JSON (BSON) or MessagePack.

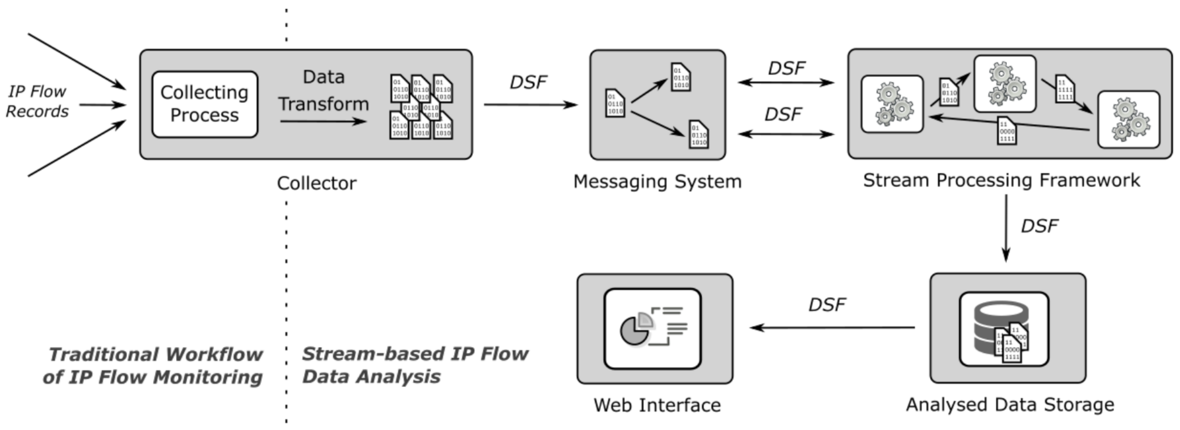

Figure 2. Stream-based workflow of network IP flow monitoring and analysis.

The collector's ability to transform IP flow records into a suitable data serialization format is currently not widespread for common IP flow collectors, but several solutions, such as IPFIXCol[8] or Logstash[9], exist, and it can be assumed that new solutions will emerge in the near future. To effectively distribute the transformed IP flows, it is advisable to utilize a messaging system that serves as an input interface for the stream processing framework. There are many such systems, such as ActiveMQ, RabbitMQ, or Apache Kafka (for the full list see [8]), however, to process IP flows it is necessary to select one providing sufficient throughput. Currently, the most suitable system is Apache Kafka, which offers sufficient message throughput and is compatible with most data stream processing frameworks.

As mentioned earlier, modern distributed stream processing frameworks, such as Samza, Storm, Spark, and Flink, provide sufficient data processing throughput. Thus, in the case of selecting an appropriate system, the decision needs to consider the deployment environment and functional requirements, such as data reliability, scalability, and operators that suit to the considered use well [7]. The intended use must be also considered during the selection of an appropriate data storage for analysis results that can be stored in a common relational database, as well as in a Next Generation Database. The storage should support advanced queries over stored data and provide an optimal interface for a web interface, so IP flow analysis results can be visualized to a user. Currently, the most common approach for connecting distributed data processing systems and data storage is the deployment of

[8] https://github.com/CESNET/ipfixcol
[9] https://www.elastic.co/products/logstash

Elastic Stack[10], composed of Logstash, Elasticsearch, and Kibana.

As discussed above, the workflow of stream-based IP flow analysis combines several interconnected components. The choice of systems for each of the components should reflect the proposed use, deployment environment, and other mentioned requirements. A list of the most suitable systems for a stream-based IP flow analysis is listed in the Table 2.

Workflow component	**Suggested systems**
Collector	IPFIXCol, Logstash
Messaging system	Apache Kafka, NATS, RabbitMQ (The full list available at [8])
Stream processing framework	Spark Streaming, Flink, Samza, Storm, Trident (All maintained by Apache Software Foundation)
Data storage	Elasticsearch, Druid, OrientDB (Next Generation Databases)
User interface	Kibana, Grafana, Tableau

Table 2. Suggested systems for the workflow of stream-based IP flow analysis.

Workflow Prototype

To demonstrate possibilities of the presented workflow for real-time analysis of IP flows, the Stream4Flow framework was introduced (available at https://github.com/CSIRT-MU/Stream4Flow). This framework, among others, interconnects modern systems for fast IP flow data processing, provides simple administration, enables fast application development and demonstrates the possibilities of stream-based IP flow analysis. The basis of the framework is formed by the IPFIXCol collector, which enables incoming IP flow records to be transformed into the JSON format provided to the Kafka messaging system. The selection of Kafka was based on its scalability and partitioning possibilities, which provide sufficient data throughput. Apache Spark was selected as the data stream processing framework for its quick IP flow data throughput [7], available programming languages (Scala, Java, or Python) and MapReduce programming model [9]. The analysis results are stored in Elastic Stack containing Kibana, which enables browsing and visualizing the results. The Stream4Flow framework also contains the additional web interface (Fig. 3), in order to make administration easier and visualize complex results of the analysis.

[10] https://www.elastic.co/v5

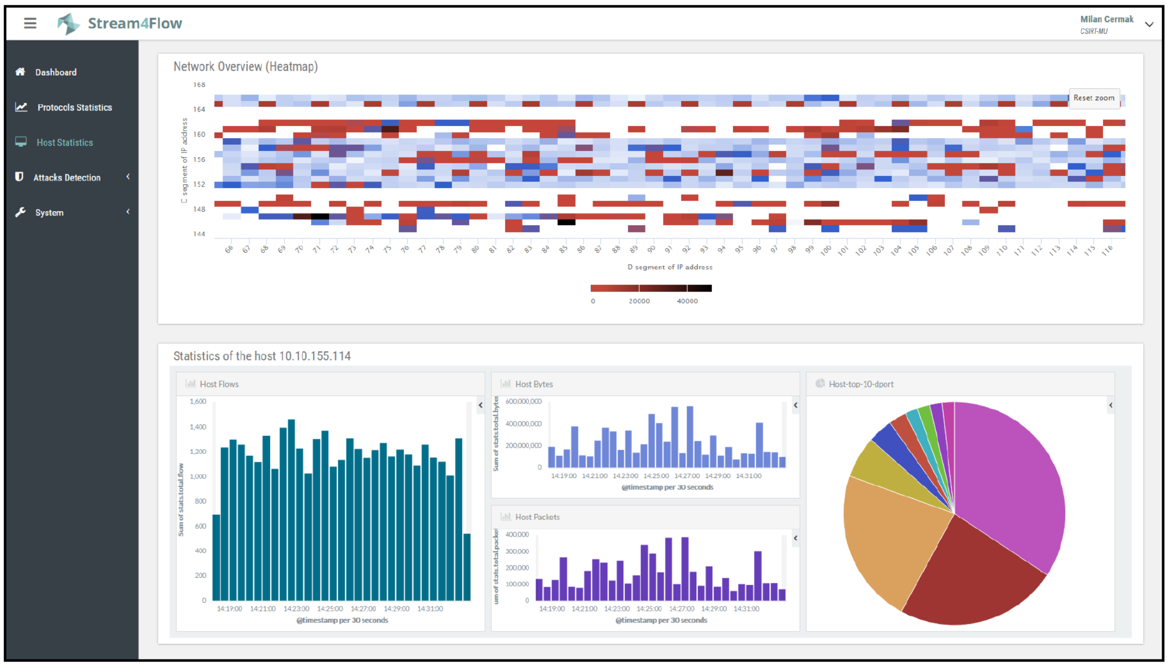

Figure 3. Stream4Flow web interface with network overview and detailed characteristics of a selected host.

Implications for Cyber Security

In the following paragraphs, we will discuss two use-cases of stream-based, IP flow analysis applications. The use-cases are chosen to reflect real-world issues of network traffic measurement for cyber security. These issues were identified by others in literature [2, 10], and confirmed by our own experience, gained during day-to-day operations of the Computer Security Incident Response Team (CSIRT). The team operates a large-scale network of 22,500 hosts with average traffic rates of 6,000 flows/second.

For each use-case, we will introduce the specific problem at hand, and discuss the possible benefits of stream-based IP flow analysis. In addition, we will share our experience with the experimental deployment of the Stream4Flow framework prototype and its applications in our network. Last, but not least, some of the possible pitfalls stemming from the deployment will be discussed. All of the deployed applications are publicly available within the Stream4Flow repository.

In-Depth Situational Awareness

The goal of cyber situational awareness is to provide in-depth comprehension of events in monitored environments, which is essential for the effective defense of computer networks [2]. A holistic view of the network (a macro view) is typically provided by traditional network monitoring applications. However, to develop a comprehensible overview of the network, more in-depth information (a micro view) is needed, for example information about individual hosts and their actions. The combination of macro and micro views gives security analysts the ability to observe the overall status, as well as the status of any specific elements in a network and so to develop an in-depth comprehension of the network.

Stream-based IP flow analysis represents a suitable approach for creating a current micro view of the network. It allows us to compute a number of detailed characteristics simultaneously due to the support of scalable and distributed computing. Distributed stream processing systems are able to create a micro view of the network as they provide enough computational power needed to compute a number of live and detailed statistics. First, these systems are designed with scalability in mind. Thus the computational resources can be instantly increased by adding additional computational nodes to the system. Second, they provide the means to distribute the IP flow analysis over multiple computational nodes, which allows analyses at a scale that is impossible on a single machine. The distribution is achieved via the MapReduce programming model [9], traditionally used for distributed batch-based processing, yet now adapted to be utilized also in a streaming fashion. Third, all the data are processed on-the-fly in primary memory, which increases throughput as no disk I/O operations are necessary during the analysis. The unique combination of scalability, distributed processing, and on-the-fly data processing makes the creation of a micro view for situational awareness possible in real time.

Our experimental deployment provides a demonstration of an in-depth situational awareness application in a stream-based workflow prototype. We use the MapReduce programming model [9] for creating the micro view by computing host statistics for all devices in our network. A host's IP address is set as a map key for data distribution. The choice of the key and MapReduce model enables us to compute detailed characteristics that represent a host's activity in the network (the number of transferred packets, bytes, communication partners, and so on), and a host's communication profile (such as frequently visited IP addresses, web pages, or active hours in a network). The micro view enables us to detect malicious hosts activities or abrupt changes in host behavior caused by the attacker. Our production instance of the prototype is able to maintain these statistics and detections for each of the 22,500 hosts in real time.

The ability of stream-based approach to process large volumes of IP flow data has also been shown in the paper [7]. We benchmarked current distributed stream processing systems and their suitability for IP flow data analysis on large volumes. The benchmark measured the throughput of the systems on a set of typical analysis queries. The systems were able to process up to 2 million flows/s on a cluster with 32 vCPU in total. This result was also confirmed by an experimental deployment of the prototype in our network, where it was able to successfully process all the provided IP flows.

It is important to point out here that stream 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. In batch-based IP flow data analysis, it is possible to perform a query over historical data, or search back through raw data for additional information after detecting a successful attack. In stream-based IP flow analysis, the data cannot be analyzed retrospectively (ex-post analysis). The start of the stream-based data analysis is marked with the creation of a particular analytical continuous query. Since ex-post analysis is as vital as real-time analysis in the context of complex network security, we recommend extending the stream-based workflow with a suitable primary data retention store to make the optional ex-post analysis possible.

Real-time Attack Detection

A cyber attack can happen in a fraction of a second and cause serious harm [2]. Distributed denial of service attack (DDoS) represents a convenient illustration of such a case. A purpose of the attack is to make a service unavailable and consequently to cause financial loss to a service provider. The cost of unavailability can reach millions of dollars per minute [3]. To reduce the costs, we need to be able to detect the cyber attack as soon as possible. The detection time can be reduced by using real-time attack detection methods instead of traditional batch-based methods with detection delays in the order of minutes. Reducing the detection delay from minutes to seconds enables us to take relevant precautions instantly and considerably reduce financial damages caused by an attack.

The stream-based approach enables immediate attack detection as it is capable of analyzing network traffic in real time. In stream-based analysis, an IP flow is analyzed on-the-fly as soon as it is received by the system or in micro-batches (for example one second batches). The detection method then reports an attack immediately as it receives the triggering IP flow. There is no detection delay as in the case of batch-based approach, where the data is analyzed in several-minute batches. Besides real-time detection, the stream-based approach offers additional benefits for detecting cyber attacks. An analysis of IP flows can be done over particularly short time windows, which can reveal information such as bursts of network traffic, which would be lost in aggregation when using the usual five-minute batches (see Fig. 4). Stream-based analysis also naturally implements sliding windows with slides smaller than the window size. This approach allows us to analyze the data and detect network attacks which would be split into two batches in non-stream approaches.

The operational deployment of real-time detection methods in our network highlights the advantages of the stream-based approach. We analyzed a sample of network traffic that contained 29 attacks captured from the daily operations of our CSIRT team. We compared the detection times of both stream-based approach and traditional batch-based approach with five-minute batches. Stream-based detection identifies an attack immediately after a triggering flow is observed whereas the traditional approach executes the detection per batch, that is once every five minutes. Thanks to this immediate attack detection, the attacks were reported 181.79 seconds earlier on average than in the case of batch-based approach. Due to this fact, we are able to use automated attack mitigation techniques more promptly and eventually mitigate even ongoing attacks.

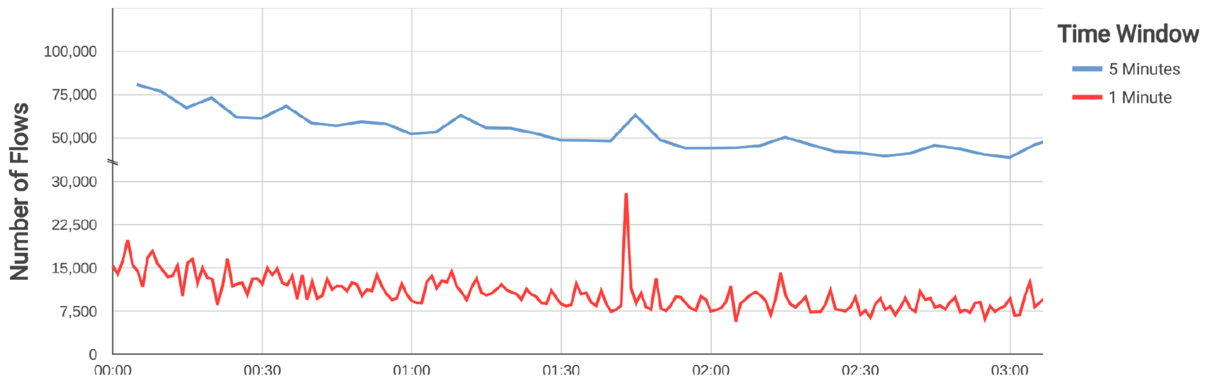

Figure 4. The discovery of a burst of traffic using short analysis windows.

Our experience shows that the majority of batch-based detection methods can be transformed into a stream-based approach. The above-mentioned reduction of the window in the stream-based approach, however, influences the IP flow analysis in several ways and it is necessary to adapt detection methods appropriately. First, the computed statistics become more volatile and detection techniques may report higher errors with the original settings. Therefore, the detection method settings (for example thresholds) need to be adapted accordingly to provide correct results. Second, the reduction of the window size raises the issue of ordering the IP flows coherently, since their misplacement to an inaccurate window may bias detection results. The IP flows may get misordered due to link latencies or data loss during their collection from the probes. Traditionally, this is managed by considering the arrival time as a baseline for ordering, but the detection methods must be adapted accordingly to reflect this necessary alleviation.

Summary and Outlook

Stream-based IP flow analysis represents a natural complement to current batch-based approaches to cyber security. It aids traditional monitoring with the ability to run analytical queries that are evaluated in real time with high throughput, low latency and good scalability, all at the same time. This allows security analysts to perform real-time analyses on network data, detect network attacks instantly, and provides them with a deep understanding of the network via in-depth situational awareness. The stream-based analysis workflow benefits from compatibility with current monitoring systems and excels in real-time attack detection, monitoring both network and individual hosts, and providing a context to network security. The presented distributed stream-based framework for IP flow analysis is able to handle streams of large volume of data at high speeds, and keeps up with latest network monitoring trends.

Since the volume, speed, and diversity of network traffic will continue to increase in the following years, network monitoring tools should follow this trend [11]. The tools of the future should be able to process traffic at speeds of over 100 Gb/s, gather more information from network traffic (such as service-specific or IoT information), and should natively support a wider variety of formats for exporting IP flows (such as DSF). In a similar manner to the probes, stream processing systems will have to process more data at higher speeds. This challenge is partially solved by the above-described scalability of current systems. Nevertheless, we expect optimizations for managing resources more efficiently in memory allocation, or query response time, for example via the use of sketches and other probabilistic data structures, which are likely to emerge. Moreover, advanced data mining and machine learning methods for intrusion detection are expected to be adapted and natively supported by future data stream systems.

We anticipate increased utilization of network IP flow monitoring for both network and host security. With the emergence of new visionary paradigms, such as the Internet of Things, host-based security will become obsolete as it will be impossible to guarantee the proper setup of host security systems for all connected devices. Network traffic measurement, namely IP flow analysis, will become essential for network defense. We believe that stream-based IP flow analysis is a suitable approach to reach the next-generation of network security.

Acknowledgement

This research 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.

References

[1] R. Hofstede *et al.*, "Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX," *Commun. Surveys Tuts.*, vol. 16, no. 4, 2014, pp. 2037-2064.

[2] A. Kott *et al.*, *Cyber Defense and Situational Awareness*, 2014.

[3] U. Franke *et al.,* "Availability of enterprise IT systems: an expert-based Bayesian framework," *Software Quality Journal*, vol. 20, no. 2, 2012, pp. 369-394.

[4] A. Sperotto *et al.*, "An Overview of IP Flow-Based Intrusion Detection," *Commun. Surveys Tuts.*, vol. 12, no. 3, 2010, pp. 343-356.

[5] B. Babcock *et al.*, "Models and Issues in Data Stream Systems," *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 2002.

[6] G. Cugola, and A. Margara, "Processing Flows of Information: From Data Stream to Complex Event Processing," *ACM Computing Surveys*, vol. 44, no. 3, 2012, pp. 1-62.

[7] M. Cermak *et al.*, "A Performance Benchmark for NetFlow Data Analysis on Distributed Stream Processing Systems," *Proceedings of the 2016 IEEE/IFIP Network Operations and Management Symposium*, 2016.

[8] L. Strzalkowski, "Queues," August 2016; http://queues.io

[9] J. Dean, and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Communications of the ACM*, vol. 51, no. 1, 2008, pp. 107-113.

[10] E. Cole, *Network Security Bible*, 2011.

[11] Cisco, "The Zettabyte Era: Trends and Analysis", August 2016; http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html

Biographies

Tomas Jirsik (jirsik@ics.muni.cz) obtained his M.S. in applied mathematics from Faculty of Science, Masaryk University, Czech Republic. He is currently a Ph.D. candidate at Faculty of Informatics and a member of Computer Security Incident Response Team of Masaryk University (CSIRT-MU). Besides his main research activities in network data analysis, anomaly detection, and host monitoring, he participates in several projects concerning network security monitoring, cyber education, and big data analytics.

Milan Cermak (cermak@ics.muni.cz) is a security researcher at the Computer Security Incident Response Team of Masaryk University (CSIRT-MU) and a Ph.D. candidate in Computer Systems and Technologies at the Faculty of Informatics, Masaryk University. His main research interests include large-volume network data analysis and advanced network threat detection based on similarity search. He is currently focusing on characterization of anomalies patterns and their utilization for real-time classification of ongoing network traffic.

Daniel Tovarnak (tovarnak@ics.muni.cz) obtained his M.S. in applied informatics from the Faculty of Informatics, Masaryk University, Czech Republic. Currently, he is a Ph.D. student at the Faculty of Informatics and a researcher in Computer Security Incident Response Team at Institute of Computer Science of Masaryk University. His focus lies on the applications of Complex Event Processing paradigm in the context of security monitoring. He specializes in security data processing and data normalization in distributed environments.

Pavel Celeda (celeda@ics.muni.cz) is an associate professor affiliated with the Institute of Computer Science at the Masaryk University in Brno. He received a Ph.D. degree in Informatics from University of Defence, Brno. His main research interests include cyber security, flow monitoring, situational awareness and research and development of network security devices. He has been participating in a number of academia, industry and defense projects. He is the head of the CSIRT-MU.