Real-time Pattern Detection in IP Flow Data using Apache Spark

Milan Cermak*, Martin Laštovička*†, Tomas Jirsik*
* Masaryk University, Institute of Computer Science, Brno, Czech Republic
† Masaryk University, Faculty of Informatics, Brno, Czech Republic
E-mail: {cermak, lastovicka, jirsik}@ics.muni.cz

***Abstract*—Detection of network attacks is a challenging task, especially concerning detection coverage and timeliness. The defenders need to be able to detect advanced types of attacks and minimize the time gap between the attack detection and its mitigation. To meet these requirements, we present a stream-based IP flow data processing application for real-time attack detection using similarity search techniques. Our approach extends capabilities of traditional detection systems and allows to detect not only anomalies and attacks that match exactly to predefined patterns but also their variations. The approach is demonstrated on detection of SSH authentication attacks. We describe a process of patterns definition and illustrate their usage in a real-world deployment. We show that our approach provides sufficient performance of IP flow data processing for real-time detection while maintaining versatility and ability to detect network attacks that have not been recognized by traditional approaches.**

I. Introduction

Although a taxonomy of network attacks has not changed significantly in recent years, the network attack landscape is still highly dynamic. New variations of network attacks are introduced on an everyday basis as attackers try to evade deployed security mechanisms. A significant portion of deployed network security mechanisms is based on pattern (signature) matching, where malicious traffic is identified based on an exact match with a predefined attack pattern [19]. Pattern matching detection methods ensure high accuracy and, however, 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 outdates quickly as a network communication and attack tools evolve. For example, attack patterns presented by Vykopal et al. [21] are not valid in current networks according to our measurements. A pattern matching mechanism that would be resistant to the attack variation is still a challenge for network traffic security monitoring.

Moreover, the contemporary network attacks can cause a severe harm every second they remain undetected. This represents a significant challenge especially in high-speed and large-scale networks that are typically analyzed with long detection delay. Analysis of traffic in such networks is usually based on IP flow measurement which is characterized by a delay of up to ten minutes [14]. Such delays may cause irreversible damage especially in the case of critical network services. Therefore, apart from the detection coverage, an analysis speed needs to be enhanced as well to identify an attack as soon as possible.

978-3-903176-15-7

A modern pattern matching application needs to meet the following requirements to face contemporary network threats successfully. First, the application should be able to detect various modification and versions of network attacks. Second, the pattern for attack detection should be easily comprehensible and definable to allow for a new pattern definition and easy adoption by network security operators. Third, the system should enable a real-time pattern matching and attack detection even in the high-speed, large-scale networks.

We face these requirements by proposing a novel approach for real-time pattern detection in IP flow data. Our pattern matching approach allows specifying various distance functions [7] and pattern definitions to enable detection of previously unknown variations of network attacks. The pattern matching approach is proposed in the context of stream-based analysis framework Stream4Flow [14] using Apache Spark analytic engine. We implement this approach in *PatternFinder* application that enables highly flexible and universal patterns definition utilizing a set of distance functions, weights, and thresholds. Its functionality is demonstrated on SSH dictionary attacks detection to facilitate understanding of the approach and its capabilities. We create a dataset for the SSH attack pattern identification, provide detection pattern for well-known attack tools, and describe the results of experimental deployment in the real-world network.

II. Related Work

Over the past years, many different ways of network traffic anomaly and attack detection approaches have been introduced from simple statistical analysis to machine learning methods [2], [9] event for IP flow [19]. However, the exact match approach prevails in the majority of commercially available analysis tools due to its low false positive rate and simple patterns definition (e.g., in Snort, Suricata, or commercially available Flowmon ADS system). Our goal is to extend this approach by using an analysis of IP flows based on similarity search principles [23]. This detection approach consists of two follow-up phases: extraction of network traffic features that form the basis of the pattern and comparison of created patterns using an appropriate distance function. Various data mining techniques can be used for this purpose as shown for

example in [13], [18], [22], [24]. Nevertheless, our observations show that these techniques are usually not designed for real-time, large-scale analysis of IP flow records and require specific data sources, preprocessing, formats and also uses complex analysis systems.

The majority of IP flow data analysis systems is based on batch data processing that significantly increases the time needed to detect attack or anomaly in network traffic. This approach is used mainly due to the performance requirements that the network traffic analysis brings. However, this issue has been overcome with the current advent of distributed stream data processing engines such as Apache Spark, Storm, Samza or Flink. Our previous results [5] showed that these systems could process a large amount of IP flow data in real time with sufficient data throughput. This approach of IP flow data processing was verified in [3] that used Apache Spark for generation of network traffic statistics. These systems allow not only real-time network traffic analysis but also provide advanced data processing methods.

III. Stream4Flow Analysis Framework

Real-time processing of IP flow data puts high demands on computation resources and requires the use of systems that scale well and allow for parallel computing. To meet this requirement, we introduced a Stream4Flow analysis framework [14] based on a stream-based IP flow analysis workflow. The framework enriches traditional IP flow monitoring architecture allowing to process data in real time and providing new analysis capabilities. It is composed of several separate systems for data normalization, analysis, and presentation. Mutual interconnection of those systems is illustrated in Fig. 1. Receiving and initial processing of IP flows is provided by IPFIXcol collector [20] capable of collecting IP flow data in various transport formats and transforming them into the JSON format. Such transformed IP flows are provided to the Kafka messaging systems [15] ensuring their scalable and fast distribution within the framework. The core of the framework consists of Apache Spark distributed stream processing engine [25] with custom applications for real-time IP flow analysis. Analysis results provided by these applications are sent back to the Kafka and stored in the Elastic Stack [8] that also offers basic visualizations using Kibana framework. The last part of the Stream4Flow framework is an additional custom web interface capable of advanced results visualization.

Data in the Stream4Flow framework are distributed in JSON data serialization format which is commonly used by modern data processing engines and provides good readability and variability. Transformation of IP flow records within the Stream4Flow is performed using IPFIXcol that offers broad configuration options including the ability to specify names for extended IP flow elements. Therefore, any information exported by monitoring probes, including information from the application layer of a network packet, can be used by analysis applications. The following sample gives a shortened example (reduced to 8 fields down from original 29) of a flow record in JSON format representing DNS response, as it is received by the analysis application.

```
{
  "@type":"ipfix.entry",
  "ipfix.octetDeltaCount": 96,
  "ipfix.packetDeltaCount": 1,
  "ipfix.sourceTransportPort": 53,
  "ipfix.sourceIPv4Address": "240.0.0.2",
  "ipfix.destinationTransportPort": 50498 ,
  "ipfix.destinationIPv4Address": "240.0.1.2",
  "ipfix.DNSName": "iuIAAAAPCAECAwIB.test.com"
}
```

The analytical part of the Stream4Flow framework is composed of Apache Spark distributed stream processing engine providing sufficient IP flow data throughput [5] and MapReduce programming model [6]. It enables to obtain in-depth information about network traffic and create advanced analysis applications. The core of the MapReduce-based analysis is specification of a key, which can be an IP address, communicating pair of devices, or arbitrary IP flow element. For such a key, it is possible to compute specific aggregations and provide detailed statistics of network traffic. These detailed aggregations and their fast distributed processing enable to create applications capable of analyzing network traffic in high-speed and large-scale networks and even in real time.

IV. PatternFinder

In addition to common applications for an IP flow analysis, the MapReduce programming model in the Stream4Flow framework allows implementing analyzes, which are difficult to perform and computationally intensive in other systems. One such analysis task is a computation of traffic aggregations for each pair of connection peers, or each observed IP address. Such aggregation can be continuously compared with predefined patterns of known network attacks or anomalies. To demonstrate this approach, we created an application *PatternFinder* that is publicly available in the Stream4Flow project repository. It is a highly flexible, easily extensible and modular application, capable of analyzing IP flow data, and comparing known patterns with real measurements in real time. In addition to a comparison of traffic patterns based on an exact match, it allows comparisons based on similarity using different distance functions [7].

The application uses a predefined data input and output interfaces of the Stream4Flow framework. At the beginning of the data processing, IP flows are filtered using a wide range of filtering options such as required fields, values, or network IP ranges. The application can handle multiple filtering options at once, whereas the IP flow has to comply with every one of these to be processed. Such cleaned data are handled by a *vector creation module*, which represents one of the crucial parts of the application. The module allows creating bi-flow or simple vectors. The simple configuration treats each combination of source and destination IP addresses and their ports independently. The bi-flow configuration attempts to group both directions of communication for their mutual analysis.

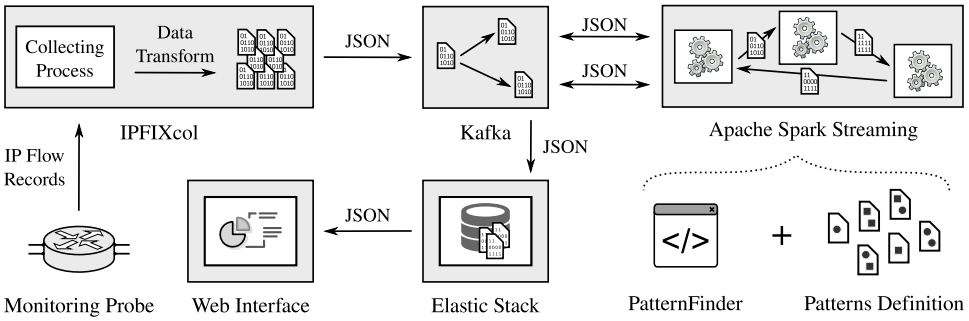

Fig. 1. The architecture of Stream4Flow framework for the real-time analysis of IP flows.

Values of bi-flow or simple vectors represent characteristics of analyzed connection that are handled by *pattern comparison module*. Each connection vector is compared in real time with predefined patterns using a defined distance function. The result of the processing is a number representing a distance of analyzed connection vector and compared patterns. The format of the pattern can be arbitrary, whereas an appropriate distance function must be correctly implemented. The following example shows a definition of patterns where the distance is computed by a Quadratic form distance function.

```
distance_function: biflow_quadratic_form
patterns:
  - name: anomaly
    request: [23, 8983, 9098]
    response: [24, 1125, 9101]
```

The last part of the analysis application is an *aggregation module* that aggregates computed distances into distribution vector. This aggregation allows considering the distance of each analyzed connection and tune detection sensitivity. The distribution vector is represented as an array containing a sum of weights in the position corresponding to the computed distance. The position is specified in the application configuration as an increasing range of values determining the range position for computed distance. A recommended approach is to set the higher weight to connections with smaller computed distance as well as the largest. This approach reduces the number of false positives caused by a small number of connections closed to the given pattern whereas the rest is very distant. The following example shows a specification of intervals and weights.

```
distribution:
  anomaly:
    intervals: [0, 3, 5, 6, 7, 11]
    weights: [3, 2, 1, 1, 2, 3]
```

The application marks the aggregation as anomaly or attack in a moment when a sum of the left half of the distribution vector is greater than or equal to the predefined limit and to the right half of the distribution at the same time. Results of the detection are sent to the Stream4Flow framework and can be explored within the Elastic Stack. Besides, we developed the web interface to represent results as an extension to the Stream4Flow web page. The interface provides a simple overview of the detection using Top K graphs of source and destination addresses with the summarizing table containing all aggregations that were identified as closest to the given patterns. Figure 2 shows an example of the table whereas the results are extended by information about confidence derived from the distance distribution.

Timestamp	Source host	Destination host	Closest patterns	Confidence
2018-08-15 08:18:32.583	240.113.0.9	240.113.0.148	ncrack-2	Medium
2018-08-15 08:13:32.413	241.128.0.58	240.30.0.112	hydra	Very high
2018-08-15 08:19:58.211	241.128.0.58	240.30.0.113	ncrack-1	Very high
2018-08-15 08:17:13.015	242.177.0.63	240.251.0.3	hydra; ncrack-2	Low
2018-08-15 08:04:23.217	240.251.0.57	240.251.0.193	ncrack-1; ncrack-2	Low
2018-08-15 07:14:28.210	242.123.0.180	240.113.0.23	ncrack-1	High
2018-08-15 07:14:28.210	241.123.0.180	240.113.0.39	medusa	Very high

Showing 51 to 60 of 239 rows 7 rows per page ‹ 1 ... 5 6 7 ... 24 ›

Fig. 2. Summarizing table of *PatternFinder* detection results.

V. Patterns Definition

Definition of an appropriate pattern is an essential process for the detection of severe events in network traffic. A weak pattern that is not specific or unique enough results in misleading or too general anomaly or attack recognition. The pattern can represent IP flow corresponding to an attack, network traffic of a particular application, or a unique user and device behavior. To derive such pattern, a trace dataset containing sufficient samples of the desired network events is required independently of derivation approach. Freely available datasets of network traffic can be used. However, these datasets typically suffer by lack of variety (e.g., contain only one version of the attack), updates [12], and correct labels [1]. For these reasons, we recommend generating a new trace dataset that is primarily focused on the desired event of network traffic.

Dataset suitable for a definition of patterns should contain only network traffic of interest without any other traffic [4]. Dataset satisfying these conditions can be created in two ways. The first is the analysis of real-world traffic and filtering of desired events. Nevertheless, this method requires recognition of events in the network traffic which is hard to achieve reliably. The second approach consists of a created virtualized

environment (e.g., virtual machines and networks, or cyber ranges) where only traffic of interest is present. For example, to generate a dataset with traffic associated with dictionary attacks on SSH service, a simple virtual client-server network can be created with several common attack applications. Using this approach, we can obtain a trace dataset containing only relevant traffic needed for pattern definition. This approach also enables to create various modifications and types of desired network traffic. In the case of SSH authentication attack, all important variations of the attack must be covered including attack speed, success rate, and differences in versions of the attack application or service itself.

Pattern definition based on the dataset can be done by various approaches using simple statistical operations, machine learning, or clustering methods [9]. Thanks to the requirement of noise traffic absence, the dataset can be easily labeled and directly used by a selected pattern definition approach. In the case of simple statistical operations, connections related to the same attack type can be aggregated, and median or average IP flow characteristic (e.g., number of packets) can be computed. The aim is to identify the most common IP flow representing the desired pattern corresponding to the anomaly-related connection. The advantage of the dataset is a possibility of its combination with real-world network traffic to improve derived patterns further.

PatternFinder application allow improving patterns in four different ways. The first is a definition of appropriate distance function used to compare the pattern with outgoing IP flows. There are many different distance functions [7] whereas its choice must take into account not only the desired functionality but also the input data. The second approach to improvement is the setting of distance intervals determining the weight of the IP flow regarding pattern similarity. If the analyzed IP flow is similar to the pattern (using given distance function), it is necessary to define smaller interval spacing so that regular connections will be on the right side of the interval while less similar connections on the left side. The third approach of patterns improvement is a definition of weights for each distance interval slot. Generally speaking, the most similar IP flows should have more significant weight as well as entirely different ones to better distinguish unambiguously identified connections. The last approach for improvement of analysis success rate is a definition of limit determining a minimal number of IP flows that need to be recognized as similar to the pattern. A higher number reduces the number of false positives but also reduces the ability of the application to recognize less significant anomalies and attacks. Each of the mentioned distribution values must always be set relative to the target network and its traffic characteristics.

A. Simple SSH Authentication Attack Pattern Definition

Usage of *PatternFinder* is best suited to detection of anomalies and attacks with similar and repetitive network connection characteristics (e.g., denial-of-service, brute-force, or wide-area attacks). To demonstrate our approach, we selected detection scenario of a dictionary attack on SSH authentication that is characterized by similar IP flows corresponding to repeated connections trying to guess user credentials. We created an annotated dataset using the above mentioned virtual environment and common tools to attack the SSH service – Medusa [11], Hydra [17], and Ncrack [16]. Each of attack tools was run five times in different settings (e.g., number of threads) to capture fifteen attack modifications in total. A core of each tool is based on a different connection library, and approach resulting in distinct IP flows with a different number of transferred packets, bytes and duration times. The dataset and derived detection patterns are publicly available in the Stream4Flow project repository.

TABLE I
PATTERNS DEFINITION FOR SSH DICTIONARY ATTACK DETECTION.

Tool	*Request*			*Response*		
	Pkts	Bytes	Duration	Pkts	Bytes	Duration
Hydra	16	1973	11959.5	25	3171	11959.5
Medusa	18	2528	6079	25	3715	6079
Ncrack-1	13	2860	2549.5	14	2103	2548.5
Ncrack-2	16	3340	10050	21	2675	10048

To show a quick and straightforward approach of patterns definition, we used the median of response and request IP flow characteristics correlated by Quadratic form distance function [7]. We counted the median of packets, bytes, and duration for each connection of the attacking tools and its setting. Hydra and Medusa tools were characterized by similar IP flow properties for all their settings. Therefore, we have decided to create patterns recognizing these tools based on the median of all their IP flows available in the dataset. In the case of Ncrack tool, one of the settings leads to completely different connections characteristics. Therefore we decided to define two distinct detection patterns. Defined patterns are summarized in the Table I. Based on the analysis of regular SSH connections and corresponding medians of request and response IP flow characteristics, we divided the distances interval in the following way to emphasize more similar and completely different connections: `[0, 2, 3, 4, 5, 7]`. Corresponding weights were selected in a standard way scaled by one: `[3, 2, 1, 1, 2, 3]`. The minimum limit of weight for reporting was set to 7, to restrict the minimum of similar IP flows to three.

VI. PATTERNFINDER EVALUATION

To demonstrate the capabilities of the *PatternFinder* application, we have deployed the Stream4Flow framework within the backbone network of Masaryk University. The framework processed IP flow data from network probes capturing IP flows from two 40 Gb links connecting the university network to the Internet. The application was deployed in the framework on four workers with two cores and 4 GB memory each for one week in August 2018 to measure its performance and real-time detection abilities. With these resources, the application was able to process a large number of IP flows without any delays. With the micro-batch size of 5 s, the average processing time of each batch was 3.39 s whereas every batch

processing finished successfully. The same processing time remained when we additionally configured comparison with 100 detection patterns. These features have persisted even if the volume of traffic increased during DDoS attacks on the university network reaching 60,000 flows per second. Our results show that the application could be deployed within the framework with fewer resources than was tested.

We selected one day of the week to deeper analyze detection capabilities of the application to show not only its performance but also its pattern detection properties. In total, 478.98 M flows were processed with 5.54 k flows per second on average and 9.9 k flows per second in peak which corresponds to 21.91 TB of data transferred. The *PatternFinder* application detected 1734 possible attacks on SSH authentication. These detections may be regular network traffic associated with connections created by a user that forgot his password or by incorrectly configured scripts. To distinguish these attempts from attacks, we analyzed the cumulative distribution function of attack login attempts. The result is depicted on Fig. 3. The analysis shows that, if the greater degree of detection certainty is needed, the required number of connections should be limited to a minimum of six. In the case of patterns used for attack detection, the limit of weight should be set to value 16 to cover five unambiguous similarities with a smaller number of less similar connections.

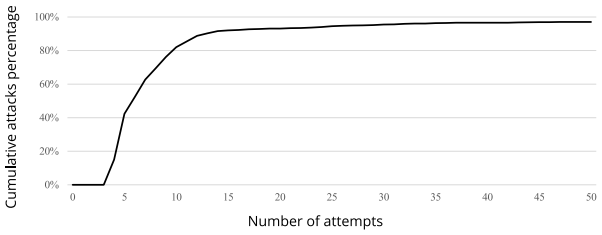

Fig. 3. Cumulative distribution of attacks based on login attempts number.

To evaluate the detection capabilities, we compared the obtained results with Flowmon Anomaly Detection System (ADS) [10] deployed in a production network of our university. The system processes 5-minute discrete batches of IP flow data and is set by default to consider 30 or more log-in attempts as an attack. With these settings, it detected 264 events from 75 unique IP addresses (a continuous attack is reported in every processed batch of data). With the same settings, the *PatternFinder* application detected 78 events from 42 unique IPs. These events overlap in some cases in the sense that ADS system reports a continuous attack every five minutes whereas *PatternFinder* can report one event covering a longer period. To evaluate *PatternFinder* performance by the standard metrics we have set the events from ADS as ground truth and counted the numbers of true and false positives, and false negatives, finally, we have set the true negative value to zero. With those values, the accuracy of *PatternFinder* detections was 39.9%, precision 82.7%, and recall 43.6%. We conclude that in real networks there are more tools in use and their pattern distance was too large for a detection. On the other hand, there were several events detected solely by *PatternFinder*. They were identified as slow stealthy attacks spanning across the time border of fixed time windows.

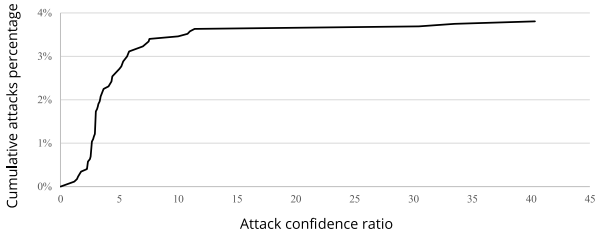

Fig. 4. Cumulative distribution of attacks based on confidence ratio.

Another essential aspect of distance-based attack detections is the confidence ratio. IP flows are aggregated by *PatternFinder* in distribution vector according to their connection vector and weight of resulting position. We divided the array into the left part (close flows) and the right part (distant flows) with the center in the middle of the array, and calculate their ratio. Surprisingly, only 66 attacks out of the total of 1734 had this ratio lower than infinity. The rest of the attacks had all flows in the left side of the distribution vector which means the attackers stick to one tool during one attack and its signature is similar to our pattern. The cumulative distribution function of attack with the non-infinity ratio is depicted in Fig. 4.

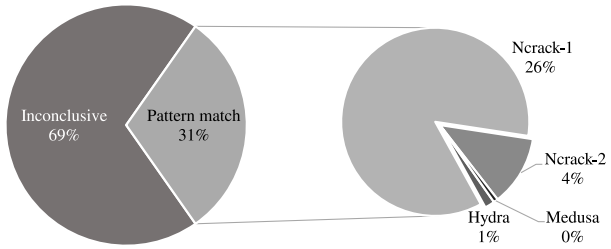

Fig. 5. Distribution of attack tools.

The most interesting feature of *PatternFinder* is its ability to distinguish some tools used for the attack based on provided patterns. We have used four patterns of three popular tools – Medusa, Hydra, Ncrack-1, and Ncrack-2. In 1205 cases of reported attacks, the distance value was the same for multiple patterns, and it cannot be determined which tool produced the attack. This problem is prominent especially in attacks with a small number of attempts where the distribution vector is not distinguishing enough. From the rest of attacks, Ncrack tool clearly dominates the detections with 516 events combined from its both patterns. Hydra and Medusa tools were identified only in 9, respectively 4 attacks. The distribution of tools in detection is depicted in Fig. 5. The analysis result can be further refined by modifying the distribution interval or weight for each of the used patterns to take into account important features of each attacking tool. However, the result may also indicate that another application or programming library is used in real-world network traffic to attack SSH authentication.

VII. Conclusion

We have proposed a novel approach for real-time pattern detection of IP flow data that can utilize various distance functions and pattern definitions. The approach is based on the Stream4Flow analysis framework [14] able to process data in real time and providing MapReduce programming model by inclusion of Apache Spark engine. The provided programming model allows the creation of an aggregation of IP flows associated with the same communication peers. Such an approach can be used to continuously compare ongoing IP flows with predefined patterns of known attack or anomalies and form aggregation vectors determining the similarity of these patterns with all connections associated with communication peers. To demonstrate this concept, we created *PatternFinder* analysis application. It is a highly flexible and modular application capable of analyzing IP flow data in real time and comparing them with known patterns using various distance functions. The application can detect not only previously known variants of network attacks and anomalies but also their unknown variations such as stealthy but long-lasting attacks.

In addition to the analytical application itself, we also introduced an approach of the definition of new analysis patterns utilizing trace datasets with a reduced noise of network traffic. These datasets can be simply used by various statistical, machine learning, and clustering applications. An advantage of such datasets is their ability to be combined with real-world network traffic. This process enables to improve pattern detection capabilities of the application and take into account the properties of the analyzed network while being able to recognize the given attacks or anomalies. The creation of patterns and their usage by *PatternFinder* has been demonstrated on the university backbone network where the application was able to analyze all ongoing IP flows (5.54 k flows per second on average) and successfully detect attacks on SSH authentication. The results showed not only good performance of the application but also its ability to identify less significant attacks and distinguish used attacking tools.

The source code of the Stream4Flow analysis framework together with *PatternFinder* and the dataset containing packet trace of SSH authentication attacks are publicly available at https://github.com/CSIRT-MU/Stream4Flow.

Acknowledgement

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). Martin Laštovička is Brno Ph.D. Talent Scholarship Holder – Funded by the Brno City Municipality.

References

[1] S. Abt and H. Baier, "Are We Missing Labels? A Study of the Availability of Ground-Truth in Network Security Research," in *2014 Third International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*. IEEE, sep 2014, pp. 40–55.

[2] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network Anomaly Detection: Methods, Systems and Tools," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[3] M. Cermak, T. Jirsik, and M. Lastovicka, "Real-time Analysis of NetFlow Data for Generating Network Traffic Statistics Using Apache Spark," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, April 2016, pp. 1019–1020.

[4] M. Cermak, T. Jirsik, P. Velan, J. Komarkova, S. Spacek, M. Drasar, and T. Plesnik, "Towards Provable Network Traffic Measurement and Analysis via Semi-Labeled Trace Datasets," in *2018 Network Traffic Measurement and Analysis Conference (TMA)*, June 2018.

[5] M. Cermak, D. Tovarnak, M. Lastovicka, and P. Celeda, "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*, 2016, pp. 919–924.

[6] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[7] M. M. Deza and E. Deza, *Encyclopedia of Distances*, 3rd ed. Springer Berlin Heidelberg, 2014.

[8] Elastic.co, "Open Source Search & Analytics · Elasticsearch," Web page, 2018, accessed January 6, 2018. [Online]. Available: https://www.elastic.co/

[9] G. Fernandes, J. J. P. C. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, and M. L. Proença, "A comprehensive survey on network anomaly detection," *Telecommunication Systems*, Jul 2018.

[10] Flowmon Networks, "Flowmon ADS ISP 9.01.00 User Guide," 2017.

[11] Foofus Advanced Security Services, "Medusa Parallel Network Login Auditor," Web page, 2016, accessed August 23, 2018. [Online]. Available: http://foofus.net/goons/jmk/medusa/medusa.html

[12] C. Grajeda, F. Breitinger, and I. Baggili, "Availability of datasets for digital forensics – And what is missing," *Digital Investigation*, vol. 22, pp. S94–S105, aug 2017.

[13] R. Hofstede, M. Jonker, A. Sperotto, and A. Pras, "Flow-Based Web Application Brute-Force Attack and Compromise Detection," *Journal of Network and Systems Management*, vol. 25, no. 4, pp. 735–758, Oct 2017.

[14] T. Jirsik, M. Cermak, D. Tovarnak, and P. Celeda, "Toward Stream-Based IP Flow Analysis," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 70–76, 2017.

[15] J. Kreps, N. Narkhede, and J. Rao, "Kafka: A distributed messaging system for log processing," in *Proceedings of 6th International Workshop on Networking Meets Databases (NetDB), Athens, Greece*, 2011.

[16] G. Lyon, "Ncrack – High-speed network authentication cracker," Web page, 2018, accessed August 23, 2018. [Online]. Available: https://nmap.org/ncrack/

[17] ——, "THC Hydra – SecTools Top Network Security Tools," Web page, 2018, accessed August 23, 2018. [Online]. Available: https://sectools.org/tool/hydra/

[18] M. Swarnkar and N. Hubballi, "OCPAD: One class Naive Bayes classifier for payload based anomaly detection," *Expert Systems with Applications*, vol. 64, pp. 330–339, 2016.

[19] M. F. Umer, M. Sher, and Y. Bi, "Flow-based intrusion detection: Techniques and challenges," *Computers & Security*, vol. 70, pp. 238–254, 2017.

[20] P. Velan and R. Krejčí, "Flow Information Storage Assessment Using IPFIXcol," in *Dependable Networks and Services*, ser. Lecture Notes in Computer Science, R. Sadre, J. Novotný, P. Čeleda, M. Waldburger, and B. Stiller, Eds., vol. 7279. Heidelberg: Springer Berlin Heidelberg, 2012, pp. 155–158.

[21] J. Vykopal, T. Plesnik, and P. Minarik, "Network-Based Dictionary Attack Detection," in *2009 International Conference on Future Networks*, March 2009, pp. 23–27.

[22] H. Wang, J. Gu, and S. Wang, "An effective intrusion detection framework based on SVM with feature augmentation," *Knowledge-Based Systems*, vol. 136, pp. 130–139, 2017.

[23] D. J. Weller-Fahy, B. J. Borghetti, and A. A. Sodemann, "A Survey of Distance and Similarity Measures Used Within Network Intrusion Anomaly Detection," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 1, pp. 70–91, 2015.

[24] U. Wijesinghe, U. Tupakula, and V. Varadharajan, "An Enhanced Model for Network Flow Based Botnet Detection," in *Proceedings of the 38th australasian computer science conference (ACSC 2015)*, vol. 27, 2015, p. 30.

[25] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster Computing with Working Sets," in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'10. USENIX Association, 2010.