Graph-based Feature Enrichment for Online Intrusion Detection in Virtual Networks

Igor Jochem Sanz[1,2] **and Otto Carlos M. B. Duarte**[1]

[1]GTA / PEE-COPPE / UFRJ – Brazil
[2]Samsung R&D Institute – Brazil

Abstract. *The ubiquitousness of Internet-of-Things devices paves the way for distributed network attacks at an unprecedented scale. Graph theory, strengthened by machine learning techniques, improves an automatic discovery of group patterns of distributed network threats omitted by traditional security systems. This dissertation proposes an intrusion detection system for online threat detection enriched by a graph-learning analysis. By using different machine learning techniques, we evaluated our system for three network traffic datasets. Results show that the proposed enrichment improves the detection accuracy up to 15.7% and significantly reduces false-positive rate. Moreover, we propose SFCPerf, a framework for automating performance evaluation of service function chaining. To demonstrate SFCPerf, we design and implement a service chain composed of an intrusion detection system and a firewall.*

1. Introduction

Network attacks are one of the main threats in a fully connected world. The growing increase in Internet-connected devices brings a range of unknown vulnerabilities. With the advent of the Internet of Things (IoT), vulnerabilities exploits affect millions of devices simultaneously. Distributed denial-of-service (DDoS) attacks and probe scans on connected devices are critical points for large-scale vulnerability exploitation and attack execution. Zombie networks composed of infected IoT devices were responsible for a DDoS rate higher than 1 Tb/s [Kolias et al. 2017]. The DDoS peak record reached 1.7 Tb/s in 2018 through amplification techniques [Morales 2018]. Late analysis of the attack showed a massive probe scan targeting *memcached* protocol few days prior the execution [Rudis 2018]. Therefore, security mechanisms that accurately detect and prevent attacks are mandatory. Additionally, detecting threats on execution time and promptly reacting to attacks are essential to reduce the impact of security threats.

Machine-learning techniques provide to security systems the capacity of learning and improving from prior experience without being explicitly programmed for it. Machine-learning techniques benefit from the huge amount of data generated by Big Data sources, to infer hidden patterns which are extremely difficult to be inferred by humans. Traditional machine-learning techniques, however, rely on classifying stored historical datasets, which restricts real-time response due to the high latency associated to the vast consumption of computational resources. Alternatively, distributed processing from a cluster of machines, assisted by stream processing frameworks, allows the construction of agile and real-time algorithms to treat huge amount of data. Therefore, there is a need for machine-learning solutions that enhance security system detection capabilities adapted to online traffic classification.

Other fundamental aspect of maintaining the network secure is the placement of the security system. The possibility of deploying traffic monitors and intrusion detection systems anywhere in the network is important to reduce the zero-day detection time and to accelerate reaction to threats. Network function virtualization (NFV) achieved notable prominence in

telecommunications and security as it reduces hardware expenditures and network operational complexity. When deploying a network security function, such as firewall or intrusion detection system, as a VNF from a chain of VNFs, the high latency or incorrect ordering of the functions imply failure of packet handling policies, increase of vulnerabilities or occurrence of security incidents. Performing repeatable and comparable experiments through an infrastructure-agnostic framework is essential to identify and avoid performance bottlenecks on NFV and SFC platforms, as well as to correctly define resource constraints. Therefore, this dissertation proposes a machine-learning solution for network threat detection adapted to real-time security systems and evaluates its implementation on a NFV scenario.

2. Objectives and Contributions

The objectives of this dissertation are fourfold: (i) improve the detection rate of distributed network threats such as denial-of-service attacks and botnets; (ii) model the network threat detection as a graph problem and investigate the use of graph-based features to improve detection capabilities; (iii) automate the experimentation and performance evaluation of different NFV scenarios; (iv) deploy an evaluate a prototype of security network function chain composed of a firewall and an intrusion detection. Hence, our proposal accomplish the objectives in two parts.

Graph-based feature enrichment for network threat detection. We propose a feature enrichment algorithm that applies concepts of Graph Theory to online intrusion detection systems. Our algorithm represents a group of network flows comprised in a time window as a graph to infer characteristics based on complex networks. The algorithm infers different metrics from the snapshots of time windows, separated in three classes: vertex metrics, edge metrics and component metrics. Then, we design an architecture that incorporates the online enrichment process for online intrusion detection systems. Figure 1 depicts the proposed system architecture. Five modules compose our proposed intrusion detection system architecture: data capture module, enrichment module, processing module, historical database, and visualization module. On the data capture module, distributed sensors collect data over the network, while all other modules run in a cluster for distributed data processing. One of the main contribution of this dissertation, the graph-based feature enriched algorithm, runs on the enrichment module. Once we generate the graph model for each snapshot in a time window, the initial feature vector, composed of 26 TCP/IP features are enriched with 39 metrics inferred for the graph analysis of the network traffic. The new features are divided in three categories and, in our study case, represents the following metrics: a) 7 local metrics: total number of vertices (distinct IP addresses) and edges (IP-IP flows) of the component (2), total number of bytes, flows and packets transmitted in the component (3), total number of distinct destination and source ports occurred in the component (2); b) 4 edge metrics: fraction of bytes, flows and packets transmitted in the IP-IP flow in comparison to the total transmitted in the component (3), betweenness centrality of the edge (1); and c) 14 vertex metrics: simple input and output degree (2), input and output degree of TCP, UDP, ICMP and IP packets (8), and input and output degree of source and destination ports from the source and destination vertices of the edge (4). Thus, the resultant feature vector is composed of 65 features.

Intrusion Detection Systems as VNFs and the Automatic Performance Evaluation. We propose SFCPerf[1], a framework to automate the performance evaluation of virtual network functions, such as virtual IDS and virtual firewall, deployed over different scenarios and conditions. Not only restricted to security functions, SFCPerf is a framework for automating experimentation of service function chaining. The framework generalizes the automation for

[1] Available at https://github.com/ijochem/SFCPerf.

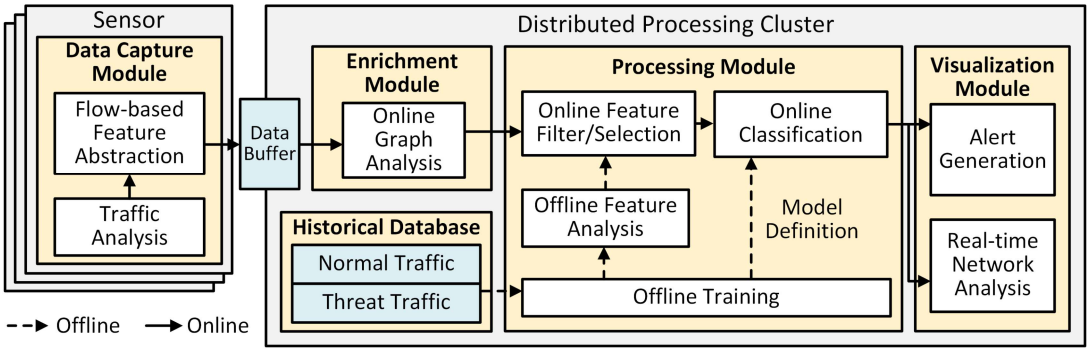


Figure 1. Architecture of the proposed classification system. The architecture is divided into five modules, data collection module, enrichment module, processing module, visualization module and historical database.

any virtual network function and service function chain orchestrated in a NFV environment. The main goal of SFCPerf is to provide repeatability to experimentation through the definition of a testing workflow. Thus, results obtained by the framework allow comparison to any other service function chain configuration, as the scenario and the experiments are strictly defined by a workflow description file. We modularly design the SFCPerf framework to allow automated SFC testing, agnostic to the underlying NFV-SFC infrastructure. Our testing framework provides experiment repeatability for all scenarios, which is essential to compare different approaches for VNF. When testing different SFC infrastructure providers, we are able to perform exactly the same experiment over the different infrastructures and, thus, we assure repeatability required to compare the performance. The SFCPerf framework is illustrated in Figure 2 and comprises the following main components: control, management, driver, passive and active data collection, analysis, and visualization modules. To demonstrate the functionality of our framework in a real use case, we develop and evaluate the performance of a security service prototype based on service function chaining. The prototype is composed of two security network functions: an intrusion detection system based on machine-learning techniques and stream processing; and an adjustable firewall with a RESTful interface.

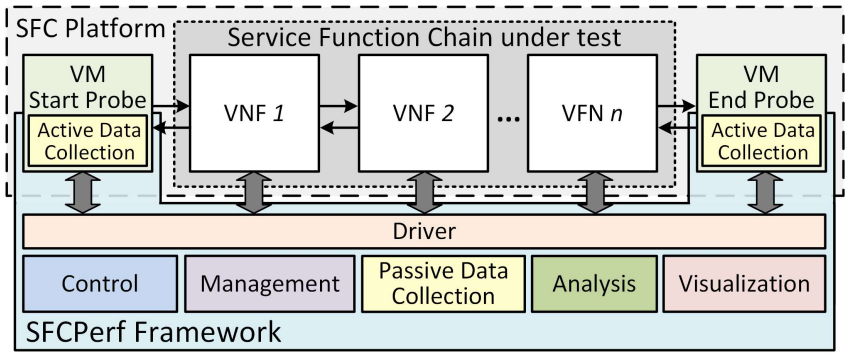


Figure 2. The proposed SFCPerf framework. The driver module provides an adaptive interface between the framework and different SFC platforms. The control module coordinates other modules to provide the environment setup, data collection and data delivery to user.

Therefore, the main contributions of this work can be summarized as follows.

- An algorithm for online enrichment of machine-learning features based on a graph-based approach.

- An architecture for online intrusion detection systems that incorporates the online enrichment process
- A framework for automating the performance evaluation of Service Function Chaining.
- The identification of major bottlenecks of VNF deployment in a NFV-SFC environment using the OPNFV platform
- A proof-of-concept and prototype for an intelligent security chain composed of a VNF IDS with a VNF firewall. The combination of both VNF provides the automatic reaction to threats when a threat is detected by IDS.

3. Related Work

Considering graph-theory approaches for threat classification, Liu *et al.* proposed an approach for detecting threats on HTTP communication using graph-based techniques to analyze data [Liu et al. 2014]. The authors restrict their proposal for the HTTP protocol and for the identification of malicious ISP clients. Iliofotou *et al.* proposed traffic dispersion graphs as a network monitoring tool [Iliofotou et al. 2007]. The authors, however, focus in classifying network traffic among different applications and not between malicious and benign. Many researches propose graph clustering and partitioning techniques to leverage the efficiency and reduce the processing load of graph analysis for large-scale datasets [Mingqiang et al. 2012, Chowdhury et al. 2017]. Anomaly-detection proposals often abstract graph streaming problems, such as the replacement of the dynamism of the Internet data traffic into consecutive static graph snapshots [Manzoor et al. 2016, Eswaran et al. 2018]. While all the above proposals focus on anomaly detection, our system focuses in the feature enrichment for classification techniques. The combination of graph theory with machine-learning-based approaches are also found in the literature to reduce false-positives [Li et al. 2016], and by extracting graph features to perform online learning to detect botnets [Chen et al. 2011]. Unlike the aforementioned work, our work proposes a graph-based enrichment to support machine-learning techniques for online intrusion detection systems.

Regarding the performance evaluation of NFV scenarios, Emmerich *et al.* evaluate the performance of virtual switches during VNF chaining [Emmerich et al. 2014]. Callegati *et al.* present a performance comparison of network virtualization and the main components of the OpenStack cloud operating system [Callegati et al. 2014]. Bonafiglia *et al.* compare the performance of different network function virtualization technologies [Bonafiglia et al. 2015]. Although most works focus on evaluating performance of NFV and SFC on a given scenario, there is a need for solutions to automate the evaluation performance of NFV use cases, regarding the interoperability problem of the early stage of NFV [Mijumbi et al. 2016]. Unlike all aforementioned works, in this manuscript we propose SFCPerf, a framework for automating the experimentation of performance evaluation of service function chaining. The framework aims to be used for performance comparison of service function chains composed of virtual network functions from different manufacturers and running on distinct NFV-SFC platforms.

4. Obtained Results

Graph-based feature enrichment for network threat detection. To evaluate the performance of this proposal, we use three different network traffic datasets, a synthetic dataset constructed in our lab (GTA/Lab), a real traffic from a Brazilian network operator (NetOp) and realistic and publicly available traces of botnet (ISCX botnet). We also compare the performance of our enriched classifier for three most used machine-learning algorithms in intrusion detection, Naive Bayes, Decision Tree and Multilayer PErceptron (MLP). Finally, we compare different set of features, by using different feature selection techniques, such as Principal Component

Analysis (PCA) and linear correlation filter. Tables 1, 2 and 3 show results of network threat detection of the evaluated scenarios for all datasets.

Table 1. Classification accuracy and area under the ROC curve (AUC) for the network operator dataset.

	Decision Tree		Naive Bayes		ML Perceptron	
Feature set (# feat.)	Acc. (%)	AUC (%)	Acc. (%)	AUC (%)	Acc. (%)	AUC (%)
TCP/IP features (26)	99.95	99.99	81.46	84.10	99.12	99.03
Graph features (39)	99.96	99.99	95.38	99.65	99.96	99.98
Enriched features (65)	99.98	99.99	94.61	99.88	99.99	99.99
PCA reduction (51)	99.96	99.99	96.03	99.84	99.19	99.71
Linear corr. filter (31)	99.94	99.99	94.60	99.86	99.99	99.99

Table 2. Classification accuracy and area under the ROC curve (AUC) for the GTA/UFRJ dataset.

	Decision Tree		Naive Bayes		ML Perceptron	
Feature set (# feat.)	Acc. (%)	AUC (%)	Acc. (%)	AUC (%)	Acc. (%)	AUC (%)
TCP/IP features (26)	99.98	99.97	75.13	86.65	99.29	99.80
Graph features (39)	99.96	99.97	82.60	97.67	99.70	99.86
Enriched features (65)	99.96	99.97	80.32	98.30	99.96	99.98
PCA reduction (46)	99.96	99.94	94.65	96.47	96.53	97.52
Linear corr. filter (33)	99.96	99.98	90.24	98.93	99.95	99.87

Table 3. Classification accuracy and area under the ROC curve (AUC) for the ISCX dataset.

	Decision Tree		Naive Bayes		ML Perceptron	
Feature set (# feat.)	Acc. (%)	AUC (%)	Acc. (%)	AUC (%)	Acc. (%)	AUC (%)
TCP/IP features (26)	99.03	99.08	85.39	89.74	95.17	96.63
Graph features (39)	99.92	99.91	82.61	84.91	98.98	99.65
Enriched features (65)	99.79	99.77	84.19	89.12	98.88	99.61
PCA reduction (27)	99.77	99.78	79.30	93.36	93.06	98.41
Linear corr. filter (38)	99.69	99.56	79.97	81.93	98.34	99.36

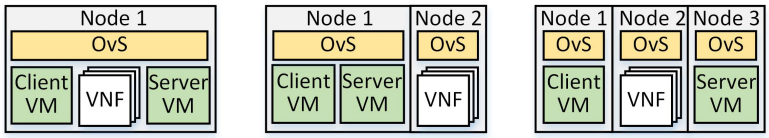
Results show that our feature enrichment proposal improves both accuracy and Area under the ROC curve (AUC) on the TCP/IP feature set for all algorithms using the NetOp dataset, and for two algorithms on the GTA/UFRJ and the ISCX dataset. Furthermore, when using a feature set composed of only the 39 graph-based features, the classification presents higher accuracy than if using the complete enriched set of 65 features. This behavior suggests that, in this case, the inclusion of particular TCP/IP features introduces noise information into the classification. Considering all evaluated algorithms, naive Bayes presents the higher accuracy gain after the enrichment process, 15.7% for the operator dataset and 9.9% for the GTA/UFRJ dataset. Indeed, when we introduce a linear correlation filter, we reduce by half the number of features to be processed while the accuracy remains stable.

To compare the relative gain after the enriched process, Table 4 shows the true positive rate (TPR) and false positive rate (FPR) for all 9 scenarios. Despite decision tree algorithm presents the best overall classification performance among all algorithms, the higher performance gain occurred for the naive Bayes algorithm in the network operator dataset, which increased the true positive rate on 25.7% while the false positive rate remained stable. Finally, we highlight the significant gain in the classification performance achieved for the MLP algorithm over all evaluated datasets. Indeed, concerning botnet traffic detection, the enrichment with MLP algorithm reduce the false positive rate in 9.4%.

Table 4. True positive rate and false positive rate comparison for all scenarios.

		DT		NB		MLP	
		TPR (%)	FPR (%)	TPR (%)	FPR (%)	TPR (%)	FPR (%)
NetOp	Ori.	99.99	0.02	62.94	0.09	98.40	0.08
	Enr.	100.00	0.00	88.62	0.10	100.00	0.02
GTA/Lab	Ori.	99.98	0.00	74.80	24.52	99.28	0.69
	Enr.	99.96	0.02	62.08	1.44	99.97	0.03
ISCX	Ori.	99.52	2.82	85.39	14.62	97.30	13.00
	Enr.	99.88	0.59	91.36	43.30	99.53	3.59

Intrusion Detection Systems as VNFs and the Automatic Performance Evaluation. To demonstrate the functionality of the proposed SFCPerf framework, we develop and evaluate a service function chain prototype considering different scenarios. We use the open platform for network function virtualization (OPNFV) with an SDN and NFV hybrid architecture to implement service function chaining. Figure 3 shows the topologies used for evaluation, in which we vary the virtual resource allocation among different physical hypervisor nodes.

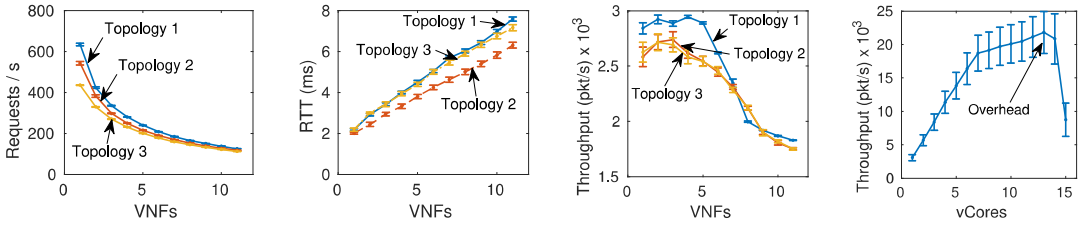


(a) Topology 1. (b) Topology 2. (c) Topology 3.

Figure 3. Topologies of the performance evaluation scenarios of the service function chaining: a) client, server, and chain of VNFs in the same node; b) client and server on a separated node from the node that hosts the chain; c) client, server, and chain on three separated nodes.

Figure 4 shows the impact on performance results obtained in each topology relative to the number of VNFs in the chain with 95% of confidence interval. Figure 4(a) compares the three topologies in relation to the rate of HTTP requests performed from a client VM to a server VM that traverses the chain. Topology 1 provides the best rate of HTTP requests for short chains of VNFs. The difference, however, becomes negligible when the chain length exceeds 8 VNFs. It demonstrates that for short chains, the overhead introduced by spreading the client, server and VNFs on different nodes is the performance limiting factor. Nevertheless, longer chains introduce an overhead that exceeds this factor. Figure 4(b) shows that the round-trip time in all topologies grows linearly with the increase of the chain length. Topology 2 presented a significantly lower latency increase than the other two, because the client and server VMs are on the same physical node, which decreases the packet round-trip time. We conclude that the increase of physical link hops, as well as the sharing of resources on the same node are factors that compromise the end-to-end delay. Hence, Topology 2 presents a fair trade-off of these factors. Figure 4(c), in turn, shows the maximum throughput in packets per second for each topology as a function of the chain length. Topology 1 presents a better throughput comparing to others when chaining few VNFs, due to a lower number of physical link hops between nodes. The major limiting factor for throughput is the *vxlan_tool* application that decapsulates the NSH packets. This application, by default, operates sequentially in only one processing core, and we extended it for parallel execution on multiple cores. The effect of this change is observed in Figure 4(d), which shows the increased throughput of a VNF in a unit-length chain in which we allocate more dedicated virtual processing cores (vCPUs). Thus, VNF retains more processing power and is able to perform more packet operations per second until it reaches the hypervisor

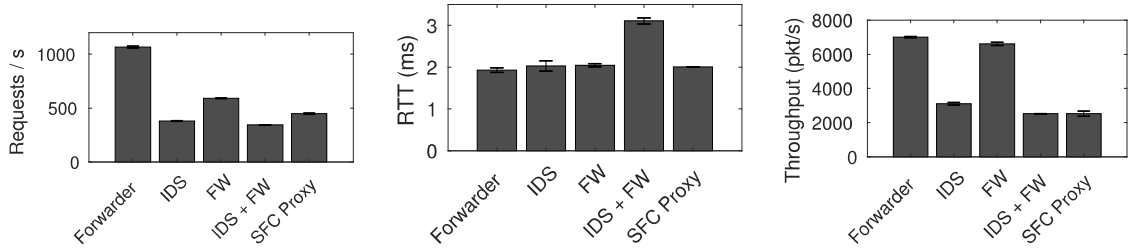
processing limit.



(a) Maximum HTTP requests rate. (b) Round trip time. (c) Maximum throughput. (d) Maximum throughput according to the number of virtual cores.

Figure 4. Impact on the performance of network function chains regarding different topologies.

As SFCPerf assures repeatability, we designed a scenario and performed a comparison experiment of four VNF approaches. Figure 5 shows the performance of one-length chains of two different simple VNFs (a forwarder, and an SFC proxy), two virtual security functions (a firewall, and a IDS), and of the composition of the two virtual security functions. Figure 5(a) and Figure 5(c) show similar results regarding the maximum rate of HTTP requests and the supported throughput of each chain. The firewall VNF presents better results than IDS VNF in both metrics, thus the chain of both VNFs has the performance limited by the IDS, with a small overhead due to the extra hop between the two VNFs. Figure 5(b), however, shows that the latency overload introduced by each virtual security function is very low, remaining at a similar time to the baseline threshold. The chain of firewall and IDS increases by 50% the packet round-trip time, which was already predicted from Figure 4(b). Nevertheless, this value is 33% lower in the case where the firewall and IDS functions separately operate over the traffic.



(a) Maximum rate of HTTP requests. (b) Average round-trip time. (c) Maximum throughput.

Figure 5. Impact on the performance introduced by each virtual security function and by the chaining of the two VNFs regarding different metrics.

5. Final Considerations

This manuscript proposed a graph-based algorithm for feature enrichment for online intrusion detection systems. The proposal enhances the detection of distributed network threats, such as distributed denial of service, port scans or botnet traces. We propose an online intrusion detection architecture that employs an enrichment module containing the proposed enriched algorithm to infer graph-based features from traffic samples collected in a fixed time window. Results of three distinct network traffic datasets show that our enrichment proposal improved traffic classification compared to the original set of TCP/IP features. In the best scenarios, the detection accuracy of distributed network-layer attacks and botnet C&C traces increased

15%, while the false positive rate was reduced by 9.4%. Furthermore, this manuscript also proposed SFCPerf, an automatic performance evaluation framework for service function chaining. SFCPerf assures repeatability for testing and comparison of different virtual network function chains. We demonstrate SFCPerf framework functionality by assessing the network performance bottlenecks of different NFV scenarios and a network function chain prototype composed of an online intrusion detection system and a firewall. Results showed that the main impact factors on the performance were the number of physical link hops between nodes and the competition for resources at shared physical nodes. As future work, we will evaluate the impact of graph-based enrichment for detecting network traffic anomalies as well as evaluating new set of features inferred from graph analysis. Moreover, we will extend SFCPerf to automatically detect SFC performance bottlenecks.

References

Bonafiglia, R., Cerrato, I., Ciaccia, F., Nemirovsky, M., and Risso, F. (2015). Assessing the performance of virtualization technologies for NFV: A preliminary benchmarking. In *2015 Fourth European Workshop on Software Defined Networks*, pages 67–72.

Callegati, F., Cerroni, W., Contoli, C., and Santandrea, G. (2014). Performance of network virtualization in cloud computing infrastructures: The OpenStack case. In *IEEE 3rd International Conference on Cloud Networking (CloudNet)*, pages 132–137.

Chen, F., Ranjan, S., and Tan, P.-N. (2011). Detecting bots via incremental ls-svm learning with dynamic feature adaptation. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 386–394, New York, NY, USA. ACM.

Chowdhury, S., Khanzadeh, M., Akula, R., Zhang, F., Zhang, S., Medal, H., Marufuzzaman, M., and Bian, L. (2017). Botnet detection using graph-based feature clustering. *Journal of Big Data*, 4(1):14.

Emmerich, P., Raumer, D., Wohlfart, F., and Carle, G. (2014). Performance characteristics of virtual switching. In *IEEE 3rd International Conference on Cloud Networking (CloudNet)*, pages 120–125.

Eswaran, D., Faloutsos, C., Guha, S., and Mishra, N. (2018). Spotlight: Detecting anomalies in streaming graphs. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD'18, pages 1378–1386, New York, NY, USA. ACM.

Iliofotou, M., Pappu, P., Faloutsos, M., Mitzenmacher, M., Singh, S., and Varghese, G. (2007). Network monitoring using traffic dispersion graphs (tdgs). In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, pages 315–320, New York, NY, USA. ACM.

Kolias, C., Kambourakis, G., Stavrou, A., and Voas, J. (2017). DDoS in the IoT: Mirai and other botnets. *Computer*, 50(7):80–84.

Li, W., Meng, W., Luo, X., and Kwok, L. F. (2016). Mvpsys: Toward practical multi-view based false alarm reduction system in network intrusion detection. *Computers & Security*, 60:177 – 192.

Liu, L., Saha, S., Torres, R., Xu, J., Tan, P.-N., Nucci, A., and Mellia, M. (2014). Detecting malicious clients in ISP networks using HTTP connectivity graph and flow information. In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 150–157. IEEE.

Manzoor, E., Milajerdi, S. M., and Akoglu, L. (2016). Fast memory-efficient anomaly detection in streaming heterogeneous graphs. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1035–1044. ACM.

Mijumbi, R., Serrat, J., Gorricho, J. L., Bouten, N., Turck, F. D., and Boutaba, R. (2016). Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys Tutorials*, 18(1):236–262.

Mingqiang, Z., Hui, H., and Qian, W. (2012). A graph-based clustering algorithm for anomaly intrusion detection. In *2012 7th International Conference on Computer Science Education (ICCSE)*, pages 1311–1314.

Morales, C. (2018). Netscout arbor confirms 1.7 tbps ddos attack; the terabit attack era is upon us.

Rudis, B. (2018). The flip side of memcrashed.