Distributed Network Anomaly Detection on an Event Processing Framework

Atanas Pamukchiev, Simon Jouet and Dimitrios P. Pezaros
School of Computing Science, University of Glasgow, Glasgow, G12 8QQ, Scotland
2031647P@student.gla.ac.uk, simon.jouet@glasgow.ac.uk, dimitrios.pezaros@glasgow.ac.uk

***Abstract*—Network Intrusion Detection Systems (NIDS) are an integral part of modern data centres to ensure high availability and compliance with Service Level Agreements (SLAs). Currently, NIDS are deployed on high-performance, high-cost middleboxes that are responsible for monitoring a limited section of the network. The fast increasing size and aggregate throughput of modern data centre networks have come to challenge the current approach to anomaly detection to satisfy the fast growing compute demand.**

In this paper, we propose a novel approach to distributed intrusion detection systems based on the architecture of recently proposed event processing frameworks. We have designed and implemented a prototype system using Apache Storm to show the benefits of the proposed approach as well as the architectural differences with traditional systems. Our system distributes modules across the available devices within the network fabric and uses a centralised controller for orchestration, management and correlation. Following the Software Defined Networking (SDN) paradigm, the controller maintains a complete view of the network but distributes the processing logic for quick event processing while performing complex event correlation centrally. We have evaluated the proposed system using publicly available data centre traces and demonstrated that the system can scale with the network topology while providing high performance and minimal impact on packet latency.

I. Introduction

Data centers have been used to provide a large and varying numbers of services, from large private compute nodes for data analytics, search engine indexing, redundant storage to public Cloud hosting a myriad of tenants. Regardless of their nature, data centers critically rely on the availability of the infrastructure and therefore anomalies, either malicious or legitimate, must be detected, analyzed and resolved. With the rapid growth in scale, volume and complexity of modern data centers the task of detecting network anomalies in short timescales is becoming more and more complex. Networks speeds have increased over the last ten years by two orders of magnitude, scaling from 1GbE infrastructure to 10GbE and 40/100GbE currently being deployed. The current approach of placing high-performance middleboxes at the core of the fabric to analyze all incoming and outgoing traffic requires a matching growth in network and compute speed. However, processor speeds have been increasing at a much lower pace, requiring a radical change on the way anomaly detection is performed.

In insight this scalability problem has been occurring in many areas of computing science; databases moved from single node relational databases to NoSQL geographically distributed systems, data analytics moved out of dedicated highly specialized infrastructures to map-reduce frameworks such as Hadoop and network infrastructures moved away from high port density devices to low-cost commodity devices. On the control and orchestration aspect, Software-Defined Networking (SDN) has allowed operators to rely on an open-API to implement new network behaviour opening new doors for routing, traffic engineering and network policies with Network Function Virtualization (NFV). For the network and compute infrastructure alike the recent trend has been to move away from high-performance dedicated systems towards commodity off-the-shelf devices in order to significantly reduce cost, prevent vendor lock-in and increase customizability and repurposability as the demand changes faster than the infrastructure. This approach to scaling changed from the traditional horizontal scaling, where a single node resources are improved, to vertical scaling where low-cost commodity nodes are added to the system.

The increasing importance and demand of data analytics generated by large-scale event-logging, machine learning, trend calculation or suggestion engines has reached scales unsuitable for node-local processing. To process realtime data streams, Twitter released in 2011 to the open source community Storm, a realtime distributed event processor that follows the horizontal scaling model. It has been designed to classify, correlate and analyze realtime streams of data from multiple sources at high speed. Through this process meaningful patterns within the streams can be detected, thus allowing events to be extracted as well as for instance their correlation, relationship, hierarchy and timing.

In this paper we propose a novel approach to address anomaly detection in data centers using the distributed event processing framework Storm. In the proposed system, the nodes and edges of the Storm topology have a one-to-one relationship to the switches and links of a traditional multi-layer topology of a datacenter. By collocating the network devices and the processing node using NFV or switch-local deployment, realtime anomaly detection can be performed across the entire fabric without the need of a high-performance middleboxes at the core of the fabric. Following the SDN approach, a central controller orchestrates the infrastructure by dynamically deploying the processing modules across the fabric and collecting the traffic characteristics as well as the reported anomalies.

The remainder of the paper is structured as follows: Section II discusses existing work. Section III presents the architecture and implementation of the proposed system. Section IV evaluates the deployment of the system and finally, Section V concludes the paper.

978-1-5090-6196-9/17/$31.00 ©2017 IEEE

II. Related Work

Currently there are two main approaches to network anomaly detection: Online and Offline. [1] Online detection is performed as traffic propagates through the network in realtime while offline detection relies on the recording and later analysis of the traffic after it has been processed. With the increasing network speeds and the necessity for realtime anomaly detection, offline anomaly detection is not suitable due to the significant lag between the anomalous event and the inspection as well the very large datasets that become hard to handle if the monitoring is done at multiple points in the network. In this paper we will focus on online anomaly detection and how our approach can address the challenges associated.

Online anomaly detection is one of the most widespread methods enforcing a security policy within a protected network. This is currently achieved by using network based intrusion detection and prevention systems (NIDS/NIPS) which apply pattern matching in order to detect abnormal behaviours. Intrusion detection/prevention systems can be classified as signature based and statistics based. Existing signature based IDS include Snort [2] - the de facto standard for intrusion detection/prevention system and a recent development - Suricata [3]. Examples of statistics based IDS are Prelude and the Analysis Console for Intrusion Databases (ACID) plugin for Snort.

Typical NIDS/NIPS are deployed on dedicated middleboxes which are connected to key points within the network topology [4]. Those IDS monitor streams of data at fixed points within networks and therefore the network-wide network characteristics such as the network topology or traffic flowing through other branches of the network are unknown. As traditional middleboxes, the IDS is transparent to the traffic and should process the stream of data with a minimum impact on the packet latency. This approach is used by softwares such as Snort and Surricata and is most common, due to being relatively easy to deploy and maintain and its ability to run on commodity servers. However the software implementations have performance limitations and compromises often need to be done such as sampling to prevent big impact on network performances. Specialized and highly dedicated hardware middleboxes have been designed to perform anomaly detection at line-rate on the aggregate of traffic, but the high-cost, high-maintenance and low-upgradability is unattractive for DC operators.

An alternative approach to intrusion detection is proposed by Steven R. Snapp et. al and relies on multiple points of attachment/monitoring that collect data and either analyse part of it locally or forward reports to a central node specifically tasked with analysis [5]. The proposed distributed approach provides a number of potential improvements such as ability to configure different sensors according to the expected workloads and increased scalability and fault tolerance due to the lack of single point of failure. However, the system proposed by Steven R. Snapp et. al is a host based intrusion detection system and it does not address the issues of underlying networks and detection performance.

Due to the increasing speed of the data centre networks it has been recognised that the centralised approach is bound to become a bottleneck [6] and some recent developments aim to improve the performance of NIDS. A tool based on Snort that divides the network traffic into manageable parts and processes them separately has been proposed by Kruegel et. al [7]. However this approach results in delays and additional overhead for distributing the network traffic, thus preventing the technique from being used for real-time analysis, as well as excluding global network knowledge as part of the anomaly detection. Another technique was proposed by Vallentin et. al which instead of correlating detection results from multiple intrusion detection sensors, the correlation is done on the underlying analysis by exchanging low level data about the traffic characteristics and thus achieving better detection results and the desired workload distribution amongst the sensors [8].

The rise of distributed computing and the emergence of parallel processing frameworks such as Hadoop has created new opportunities to address the poor scalability and limited performance of centralised NIDS solutions. A network measurement tool based on Hadoop that shows how parallel computing frameworks can be used in the context of high speed networks has been proposed by Yeonhee Lee & Youngseok Lee [9]. Hadoop based intrusion detection systems have been developed by Ibrahim Aljarah and Simone A. Ludwig [10] and Sanjay Veetil & Qigang Gao [11] that significantly improve parallelization of data analysis while achieving a similar detection rate when compared to existing NIDS such as Snort and Surricata. However, due to the evident limitations of Hadoop – being created as a batch processing framework, none of the proposed solutions can be used to analyze traffic in realtime. They rely on conversion of the network traffic which is captured live into a batch of data to be processed by the Hadoop cluster. This is a major issue, limiting the adoption of parallel processing techniques into network data analysis as the long and unstable latency between batch processes makes it infeasible to use Hadoop for online anomaly detection.

The lack of realtime parallel processing frameworks has been addressed by Apache Storm, which does not require data to be organised into batches before processing, but directly processes all data as soon as it is available. A Storm application processes streams of *tuples* flowing through *topologies* [12] where each topology is made up of a directed acyclic graph of Bolts and Spouts. Spouts introduce data to the Storm topology in the form of tuples, which is typically achieved by polling external queues. Bolts are responsible for processing the tuples, data aggregation and consequently for forwarding the tuples to the next set of bolts. The concept of tuples and topologies has been the main driver for this work as it not only allows realtime (low-latency) processing of input data, but also strongly resembles the structure of data centre networks with tuples representing the individual packets and the DAG topology matching the physical network topology. This approach makes it significantly easier to produce a Distributed IDS that adapts to the architecture of specific data centres. Moreover, the functionality provided by bolts and spouts can be deployed directly on network infrastructure or collocated with some network devices by leveraging NFV [13], [14], [15] which gives Storm the potential to bridge the gap between the Hadoop based systems and the requirements of the current high speed networks.

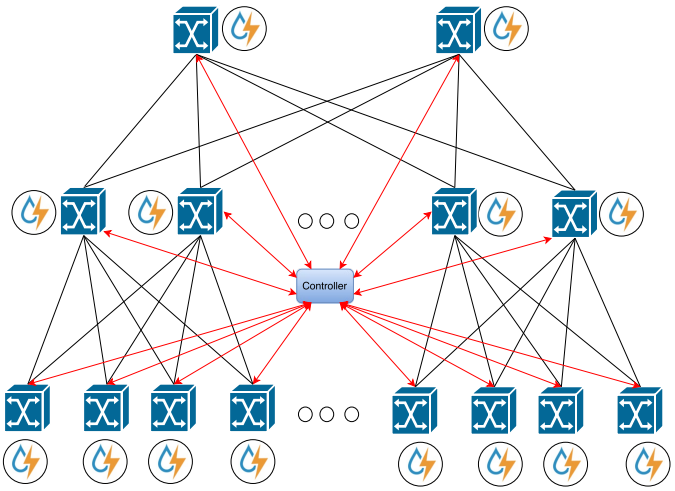


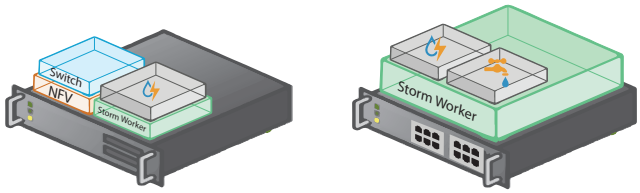
Fig. 1. Data Centre canonical tree topology and the corresponding Apache Storm DAG topology including bolts and spouts

III. System Architecture

A. Storm on the Network Fabric

The architecture of the proposed anomaly detection system is closely coupled with the architecture of Storm applications and leverages Storm's *tuple* and *topology* abstractions to construct a Directed Acyclic Graph (DAG) that represents the flow of data within a specific target data centre. An advantage of Storm for this particular use case is the direct mapping between the components of Storm and the components of a data centre network. Bolts in a Storm topology are similar to network switches, the former processes tuples of data while the second processes packets with the outcome dictating where the tuple or packet should be forwarded. Spouts are analogous to the hosts within the data centre and to the core routers. Similarly to how the hosts generates egress traffic and core routers receive ingress traffic from the outside world, the spouts generate tuples to be processed by the topology. This similarity allows seamless generation of the required Storm topology regardless of the complexity of the underlying network.

Figure 1 shows a typical data centre canonical tree topology in interconnecting the edge hosts as well as the corresponding Storm topology made up of bolts and spouts. The two topologies are identical as Storm bolts are deployed on top of the existing data centre infrastructure. The black arrows represent the flow of data, which can be in any direction. This allows the system to be deployed facing either the outside world or the edge hosts. Since Storm restricts tuples to flow in a single direction across the DAG, if bidirectional protection is required, two identical Storm topologies can be deployed on the same infrastructure, each performing traffic monitoring in one of the directions. Two symmetrical deployments can be used to monitor the two-way traffic, allowing fine grain allocation of detection modules based on whether the traffic to analyze is ingress where anomalies such a DDoS might be prevalent, or egress with different anomalies such as malware traffic.



(a) Deployed on commodity infrastructure using NFV

(b) Deployed directly on network switches

Fig. 2. System Deployment at the infrastructure level

B. Storm NIDS architecture

The proposed NIDS builds on top of the existing data centre network by deploying Storm Bolts on network switches and performing anomaly detection at line-rate on the traffic. Deploying the anomaly detection as part of the switching fabric allows the packet processing and the packet forwarding logic to be collocated, allowing high performanbce by preventing the same packets to be parsed multiple times and by distributing the packet processing logic across the entire fabric. However consideration must be taken into account before applications are deployed into the switches across the fabric not to adversely impact performance. The anomaly detection module must be kept lightweight in processing cost to not impact the packet processing time and in per-packet memory utilisation to allow large number of flows to be monitored simultaneously. However, the performance limitation of the individual switches is becoming less and less an issue, with many providers virtualizing the network functions on traditional commodity hardware using NFV. Figure 2 shows two possible deployments of the proposed architecture, in a) the NIDS is deployed as a software collocated with the software switch using NFV and in b) a storm worker agent is deployed on a high-performance switch.

To better leverage the intrinsic network topology for the anomaly detection, the location of a network switch must be considered while deploying the anomaly detection modules. Different anomaly detection algorithms require different views of the network, ranging from simple features of the aggregate traffic such as aggregate flow volume over time across a particular device or port to per-flow complex features to detect more fine-grain anomalies such as malwares. By deploying the anomaly detection modules for a particular feature on a specific switch or set of switches can be beneficial for the performance of the system and the accuracy of the detection. Typically the core of the network fabric is highly utilized due to the aggregate of ingress and egress traffic as well as the cross-rack traffic. This particular traffic pattern makes the core suitable to detect large volume anomalies and surges possibly impacting the entire fabric. The aggregation layer contains a finer-grain view of the network traffic and therefore a more in-depth analysis can performed such as per-host network behaviour highlighting anomalies such a DoS or flash-crowds. Finally at the edge layer, fine-grain anomaly detection can be performed as the traffic volume is limited.

The detection modules are responsible for extracting relevant fields from the packets processed and checking them for anomalies. Table I shows the current features that the sensors can perform anomaly detection on, and collect statistics. The

TABLE I. PACKET FEATURES EXTRACTED BY THE DISTRIBUTED ANOMALY DETECTION MODULES

Layer	Fields
Link	SrcMACAddress, DstMACAddress, 802.11Q tag
Internet	SrcIPAddress, DstIPAddress, ProtocolType
Transport	SrcPortNum, DstPortNum
Application	PDU contents (e.g. HTTP URI and body)

anomaly detection data extracted at each layer by the detection modules can also be aggregated to detect more complex anomalies that need both a flow aggregate and per-flow view. A straightforward example of the benefit of such approach is for the detection and remediation of a DDoS. At the core layer, the aggregate view of the traffic will show a surge highlighting that the network is under attack (DDoS). At the aggregation layer, if per-host characteristics are monitored, the host under attack can be identified. Finally at the edge the service under attack can be identified as well as the type of attack used such as a simple SYN flood or more complex attacks such as NTP augmentation. To deploy the modules across the different layers of the topology and analyze the collected network features a central controller was designed as part of the proposed architecture.

C. Centralized management and configuration

Due to the distributed nature of the proposed system the overall approach to anomaly detection differs from the traditional NIDS. As the system relies on modules (bolts) being deployed on all of the switches in the data center fabric, the system needs to be designed to perform anomaly detection based on data collected from multiple points. The architecture of the system therefore focuses on modules with their specific configuration in addition to a single centralised node used for reconfiguration and complex event correlation, rather than the traditional monolithic middlebox approach. Each module within the system can be individually configured using an API between the controller node and the modules. When a module is instantiated in the Storm topology, a connection is established between it and all its neighbour modules as well as the controller node. This allows the controller to push new configuration at runtime to adapt the anomaly detection over time as the network behaviour changes. In an SDN environment the Storm NIDS controller node can be collocated with the SDN controller as a Southbound interface allowing a single controller to manage the routing policies as well as the network state and security.

Using the southbound interface, new bolts can be deployed in and out of the topology in order to adapt the anomaly detection checks performed, allowing the network anomaly detection to evolve over time as new threats emerge. The system is capable of swapping bolts that implement different levels of functionality ranging from signature based detection bolts to statistical features bolts. The system control plane is responsible for bringing the newly introduced bolt up to date with the latest configuration by pushing series of configuration messages that sync the bolt with the rules and signatures it requires for its checks. Secondly the controller acts as an hypervisor for the bolts, monitoring their state and behaviour over time and allowing a quick recovery on failure of the bolt. The resulting highly flexible system provides a number

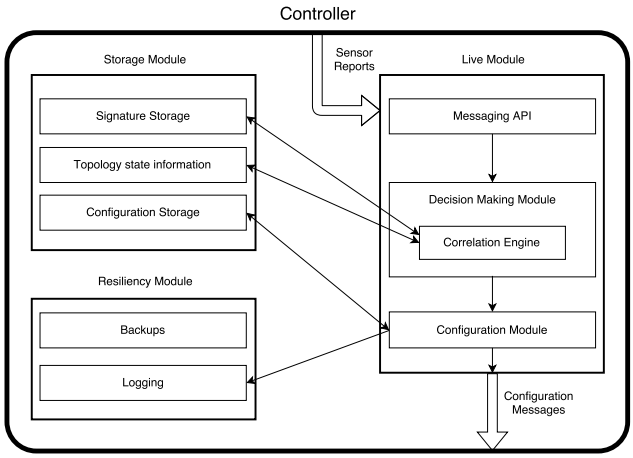


Fig. 3. Overview of the architecture of the proposed network intrusion detection system using Apache Storm

of use cases such as dynamic reconfiguration following the detection of a complex event. For example during a DDoS attack if multiple modules at the same layer of a data centre network report service request anomalies, the controller may trigger the reallocation or instantiation of more DDoS detection modules in order to better detect the source of the anomaly. Subsequently after the attack is mitigated the controller may restore the configuration back to its original settings, possibly releasing resources for other anomaly detection modules to be instantiated. This approach opens up a new level of freedom towards an adaptive high-performance intrusion detection system.

The controller node is responsible for maintaining stateful system operation and for tracking the status and reports from the modules. It maintains a Storage Module seen in Fig. 3 that contains the known signatures and any topology state such as sensor locations and neighbours and the current configuration of each sensor, thus supporting the reconfiguration capabilities of the system. The second important role of the controller node is to perform event correlation across the data collected from the multiple modules to detect more complex anomalies that do not follow a simple signature or statistical pattern. This is achieved as individual sensors send reports through the messaging API seen in Fig.3. It processes and passes the information to the Decision Making Module which, consults the Signature and Topology state Storages in order to detect anomalies. By correlating data from multiple layers different network views can be used to analyze network-scale coarse grain behaviour and node-local per-flow characteristics allowing to investigate anomalies at the macro- and micro-scales. Finally if reconfiguration is required the configuration module generates the required messages and outputs them to the sensors concerned.

As most anomaly detection approaches, the event correlation may lead to true negatives or duplicate detection events which are highly undesirable in critical infrastructures such an IDS. As the controller correlates inputs from multiple sources, it uses its network-knowledge such as the sensor location to deduce if series of reports have the same locality. The separation of responsibilities between the modules and the controllers results in a two layer architecture with Tactical and

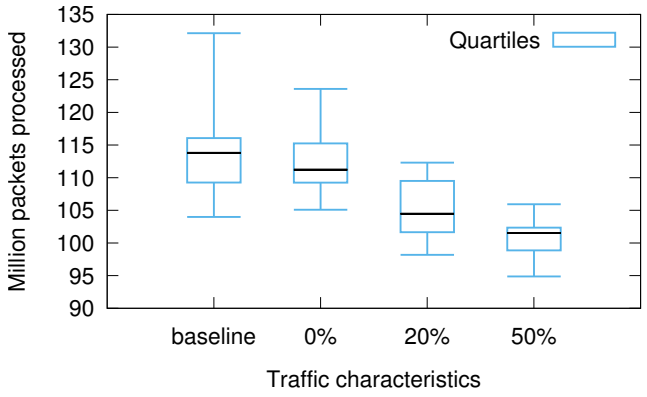


Fig. 4. Packet processing performance per deployment with a varying percentage of anomalous traffic.

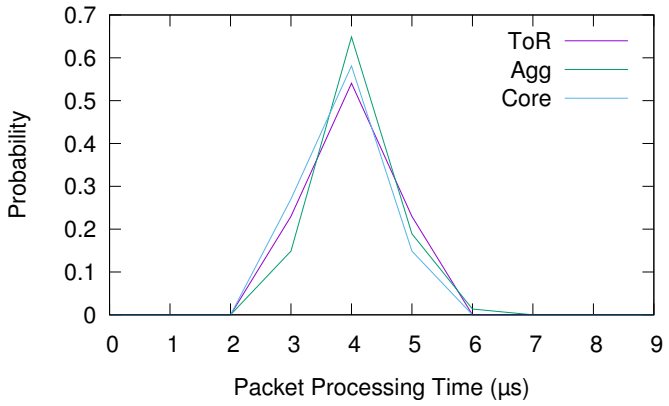


Fig. 5. Probability Density Function (PDF) of the per packet processing time depending on the modules' deployment location

Strategic decisions. The decisions made by sensors themselves are classified as Tactical as they rely only on their own knowledge, and do not take into account the state of the rest of the network. Any decisions made by the controller node are Strategic as they rely on data from, and concerning multiple modules. As a result the controller is capable of correcting duplicate detections thanks to its knowledge of all the events detected by sensors.

IV. Evaluation

In this section we will cover the evaluation process for the proposed intrusion detection system and provide details on the experimental deployments and the overall performance achieved by the prototype system.

A. System Deployment Characteristics

The proposed system is based on Storm and therefore a Storm cluster has to be set up in order to deploy the system. A storm cluster consists of a number of Storm workers, a single Nimbus and a single Zookeeper was created. All of the above were deployed on computers running Scientific Linux release 6.7 on quad core Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz with 4GB of memory and gigabit connectivity. Experiments were performed with 23 workers deployed apart from the independent Nimbus and Zookeeper and were based on the canonical tree topology shown in Fig. 1. This network involves 16 distinct network switches, each deployed on a separate worker thread. The remaining 7 workers are deployed as follows: 1 worker hosting the controller node, 4 workers responsible for introducing network traffic to the topology referred to as Spouts , 2 responsible for collecting the traffic processed by the topology and verifying the anomaly detection results with the traffic processed referred to as Aggregators.

To evaluate the system with a realistic workload, traffic and packet distribution the anonymized dataset *univ1* from IMC 2010 - Network Traffic Characteristics of Data Centers in the Wild has been used [16]. The use of offline traffic captures from a data centre ensures consistent results between experiments and provides a highly realistic testbed for the proposed system.

B. Performance

As anomaly detection is the most important feature of the system, the performance evaluation begins by comparing the performance of the system when operating in a pass-through mode where all of the bolts are switched off and no anomaly detection is performed,to a normal operation mode when all of the anomaly detection modules are operational. Fig. 4 shows the difference in the system performance in both scenarios. It highlights that the performance degradation when operating in anomaly detection mode are minimal compared to normal operation. In particular the difference in total number of packets processed during operation between the two modes equals to 4.5% while the difference in the average number of packets processed between the two modes is 0.5%. The results clearly show that performing anomaly detection does not add significant overheads for the system during operation.

As we have measured the system performance during normal operation, it is important to measure the efficiency of the system when it is tackling anomalous traffic as the distributed nature of the system requires additional synchronisation to occur when anomalous traffic is detected. For example, when a module detects an anomalous packet, a report is sent to the controller indicating the type of anomaly detected. This allows the controller to infer anomalies spanning multiple modules and take action, but the generation and transmission of those messages can impact the performance of the module. Figure 4 shows the number of packets processed by the system under different quantities of anomalous traffic. For the purposes of the experiment anomalous traffic was introduced by a dedicated module responsible for reading the static packet trace and introducing packets with random anomalies of different types. The use of a dedicated module allowed fine-grained control over the amount and characteristics of the anomalous traffic introduced which allowed more accurate experiment reproduction.

The experiment was split in two stages. During the first stage 20% of the traffic introduced contained various anomalies such as invalid fields, blocked source and destination addresses, application layer packet contents marked as anomalous and even Denial of Service attacks. As a result of the increased anomalous traffic the overall throughput of the system drops by 3.35% on average compared to normal operation due to

the increased average computational load on each module. The second experiment injected 50% of the packets with anomalies and resulted in a total of 7.32% lower number of packets processed overall. As in the previous experiment, increasing the amount of anomalies in traffic results in a small performance penalty.

Figure 5 shows the probability density function (PDF) per-packet processing time taken by the system for each of the incoming packets. The results are grouped by the three layers of bolts mapping to the layers of the underlying network topology. It highlights that the individual packet processing time within the event processing framework is highly stable across the different layers of the fabric. Regardless of the layer all the packets are processed in less than 7 microseconds with over 50% of the packet being processed in 4 microseconds. This low processing time demonstrates that realtime anomaly detection using an event processor is possible without impacting the packet latency or resorting to sampling techniques.

V. Conclusion

The current centralised approach to network intrusion detection systems is bound to become a bottleneck with the rapid adoption of 40GbE and 100GbE networks as it requires the underlying hardware to scale vertically, thus resulting in significant cost increases and vendor lockdown. In this paper we proposed a distributed approach to intrusion detection that is based on a parallel processing framework and is capable of exploiting the underlying network topology in order to distribute the anomaly detection workload. We have prototyped the proposed approach as an intrusion detection system with distributed sensors and a centralised controller.

The system was evaluated using publicly available data centre traces in order to measure the performance and latency of the system in a realistic scenario. The experimental results indicate that the system is capable of efficiently distributing the anomaly detection workload and operating under heavy load without suffering from degraded performance or increased packet processing time. The proposed system shows that a distributed intrusion detection system based on event processing framework is a promising approach to anomaly detection and can be used where scalability and efficient resource utilization is key.

Acknowledgments

The work has been supported in part by the UK Engineering and Physical Sciences Research Council (EPSRC) projects EP/L026015/1, EP/N033957/1, and EP/L005255/1.

References

[1] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Comput. Netw.*, vol. 51, no. 12, pp. 3448–3470, Aug. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2007.02.001

[2] M. Roesch, "Snort - lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX Conference on System Administration*, ser. LISA '99. Berkeley, CA, USA: USENIX Association, 1999, pp. 229–238. [Online]. Available: http://dl.acm.org/citation.cfm?id=1039834.1039864

[3] E. Albin and N. C. Rowe, "A realistic experimental comparison of the suricata and snort intrusion-detection systems," in *Proceedings of the 2012 26th International Conference on Advanced Information Networking and Applications Workshops*, ser. WAINA '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 122–127. [Online]. Available: http://dx.doi.org/10.1109/WAINA.2012.29

[4] E. H. Spafford and D. Zamboni, "Intrusion detection using autonomous agents," *Comput. Netw.*, vol. 34, no. 4, pp. 547–570, Oct. 2000. [Online]. Available: http://dx.doi.org/10.1016/S1389-1286(00)00136-5

[5] S. R. Snapp, J. Brentano, G. V. Dias, T. L. Goan, L. T. Heberlein, C. lin Ho, K. N. Levitt, B. Mukherjee, S. E. Smaha, T. Grance, D. M. Teal, and D. Mansur, "Dids (distributed intrusion detection system) - motivation, architecture, and an early prototype," in *In Proceedings of the 14th National Computer Security Conference*, 1991, pp. 167–176.

[6] L. Schaelicke, T. Slabach, B. Moore, and C. Freeland, *Characterizing the Performance of Network Intrusion Detection Sensors*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 155–172. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-45248-5_9

[7] C. Kruegel, F. Valeur, G. Vigna, and R. Kemmerer, "Stateful intrusion detection for high-speed network's," in *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, 2002, pp. 285–293.

[8] M. Vallentin, R. Sommer, J. Lee, C. Leres, V. Paxson, and B. Tierney, *The NIDS Cluster: Scalable, Stateful Network Intrusion Detection on Commodity Hardware*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 107–126. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74320-0_6

[9] Y. Lee and Y. Lee, "Toward scalable internet traffic measurement and analysis with hadoop," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 1, pp. 5–13, Jan. 2012. [Online]. Available: http://doi.acm.org/10.1145/2427036.2427038

[10] I. Aljarah and S. A. Ludwig, "Mapreduce intrusion detection system based on a particle swarm optimization clustering algorithm," in *2013 IEEE Congress on Evolutionary Computation*, June 2013, pp. 955–962.

[11] N. Sharma and S. Mukherjee, "Layered approach for intrusion detection using naĭve bayes classifier," in *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, ser. ICACCI '12. New York, NY, USA: ACM, 2012, pp. 639–644. [Online]. Available: http://doi.acm.org/10.1145/2345396.2345500

[12] A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, N. Bhagat, S. Mittal, and D. Ryaboy, "Storm@twitter," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '14. New York, NY, USA: ACM, 2014, pp. 147–156. [Online]. Available: http://doi.acm.org/10.1145/2588555.2595641

[13] "Network functions virtualisation introductory white paper."

[14] R. Cziva, S. Jouet, and D. Pezaros, "Gnfc: Towards network function cloudification," November 2015. [Online]. Available: http://eprints.gla.ac.uk/105962/

[15] R. Cziva, S. Jouet, K. J. White, and D. P. Pezaros, "Container-based network function virtualization for software-defined networks," pp. 415–420, 2015. [Online]. Available: http://eprints.gla.ac.uk/105926/

[16] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 267–280. [Online]. Available: http://doi.acm.org/10.1145/1879141.1879175