A Hierarchical Intrusion Detection System using Support Vector Machine for SDN Network in Cloud Data Center

Quentin Schueller
Kashinath Basu
Muhammad Younas
School of Engineering, Computing and Mathematics
Oxford Brookes University
Oxford, United Kingdom
{kbasu, m.younas}@brookes.ac.uk

Mohit Patel
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Mumbai, India
mohitpatil@cse.iitb.ac.in

Frank Ball
Frank Ball Consulting
Oxford, United Kingdom

***Abstract*— Software-Defined Networks (SDN) has emerged as a dominant programmable network architecture for cloud based data centers. Its centralised programmable control plane decoupled from the data plane with a global view of the network state provides new opportunities to implement innovate security mechanisms. This research leverages this features of SDN and presents the architecture of a hierarchical and lightweight Intrusion Detection System (IDS) for software enabled networks by exploiting the concept of SDN flows. It combines advantages of a flow-based IDS and a packet-based IDS in order to provide a high detection rate without degrading network performances. The flow-based IDS uses an anomaly detection algorithm based on Support Vector Machines (SVM) trained with DARPA Intrusion Detection Dataset . This first line of defence detects any intrusions on the network. When an attack is detected, the malicious flow is mirrored to a packet-based IDS, for further examination and actions. The results show that this scheme provides good detection rates and performances with minimal extra overhead.**

Keywords—intrusion detection system, machine learning, support, vector machine, software defined network, cloud computing

I. Introduction

Cloud based data center provide a wide range of services such as Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS). The very nature of the cloud based service model requires that services are open ended and available to its customers with appropriate level of quality of service (QoS). The SDN network has emerged as a popular networking choice for managing the networking aspects of complex QoS requirements within the cloud [1]. A key requirement of the networking infrastructure is also to provide adequate level of cyber security and protection to cloud services without compromising QoS and preserving the openness appropriate to the service. Though SDN is extremely robust in handling different categories of traffic with varying QoS requirements it lacks in-built security mechanisms. Similar to traditional networks, additional layers of security are required to protect data sent over the networks. The typical security lines of preventive defence are anti-virus, encryption, firewall, access control list (ACL), etc. However, even with a strong defence line, security threats still exist and an intrusion may occur. Thus, complementary reactive mechanisms of security like IDS are required [2]. An IDS detects intrusions by analysing the information that can be gathered from the system. In the case of network-based IDS (NIDS), data is collected directly from the network through an inspection of transiting packets. However, traditional IDS like Snort [3] or Suricata [4] analyses every packet. However, due to large volume of data, this degrades the performance of the network and results in increase in the network delay and additional processing overhead on the infrastructure.

Traditional network has inherently a stateless core and services each packet individually. But in an SDN the concept of stateful core with flows and centralised control can be exploited to provide better security features. Based on these advantages of SDN, this research presents the architecture of a hierarchical IDS scheme that is both lightweight and scalable and provides identical performance of that of a traditional packet-based scheme. The designed solution is composed of a lightweight flow-based IDS that implements an anomaly-based detection scheme trained with a SVM engine. At the next level, any malicious flow that is detected is passed through a packet-based IDS using a signature-based detection for further investigation. The proposed scheme exploit the advantages of SDN as well as recent development in machine learning to provide a solution that is modular and can be regularly updated by training the algorithm with new data set.

The rest of the paper is organised as follows: section II provides a background and analysis on SDN, IDS, SVM machine leaning scheme and the dataset used in the research; section III presents the architecture of the proposed IDS; section IV explains the testbed and the evaluation criteria; section V presents the analysis of the experiment results; and finally section VI concludes with a reflection of the key achievements and potential future work.

II. Background

A. *Software-Defined Networks*

The traditional networks are built on proprietary devices with embedded software requiring manual configurations of each device. This concept is error-prone and complex. Moreover, response to changes based on network conditions is extremely slow and labour intensive [5]. The SDN model decouples the network into application, control and infrastructure (or data) plane. The infrastructure plane (both physical and virtual) consists of the data forwarding devices such as routers, switches, access points, etc. which are managed by the control plane. A controller sits in the control plane and manages a high level network wide state of the relationship between network resources, policies and services. The application plane can host third party applications for network monitoring, load, balancing, QoS, security, etc. The IDS developed in this research is hosted on this plane. The communication between the planes is via the northbound and southbound interfaces. The three planes can be developed independently and services added to them as long as the communication interfaces between the planes are compatible. In this research, the OpenFlow (OF) [6] signalling protocol has been used at the southbound interface for communication between the control and infrastructure plane and the rest API at the northbound for communication between the IDS and the controller. Both these protocols are actively developed and are popular choice for future emerging SDN based data center solutions.

Any OpenFlow compatible forwarding device such as a OpenFlow switch will contain a flow table containing information about flows and the corresponding actions on the matching flows. Any unknown flow is always passed to the control for further processing. The controller can also query and update the flow table with appropriate signalling primitives. Our IDS uses these primitives to collect information about the flows and to remove malicious flows.

B. *Intrusion Detection Systems*

The role of IDS is detecting events occurring in a computer or network system and identifying any violations of policies, malicious activities, unauthorized or abuse of computer systems. Although IDS can be implemented both in host (Host-based IDS) or network (Network-based IDS), the focus of this research is on network-based IDS. Some popular open-source network based IDS include Snort, Suricata and Bro [7]. One way of classifying the different types of IDS is based on the type of analyser and detection algorithm used.

The detection algorithms can be divided into the signature-based (or misused based) algorithm and the anomaly-based (or behaviour based) algorithm. In the former case, detection of suspicious behaviour is based on string pattern matching with pre-defined patterns of known attacks stored in a database as signatures. This type of detection can only detect pre-defined known attacks with great accuracy, but new attacks cannot be detected until updating the signature database. Signature-based IDS has a very low false positive rate but a higher false negative rate [8]. It also requires every packet to be compared with a serialised list of signatures. It can be a resource intensive and a slow process depending on the volume of traffic and the size of the signature database. Signature based schemes cannot take advantage of the flow level abstraction supported in SDN where only flows which needs further analysis are meant to be forwarded to the controller. This contradicts with the working process of signature packet-based IDS scheme such as Snort where every packet transiting the network needs processing and hence has to be mirrored to the controller for inspection thus degrading the performance and overhead on the controller [9]. Hence this type of scheme is only suitable for suspicious flows which have already been marked for further analysis. In the proposed IDS, signature based detection is only used to a subset of the flows identified as potentially suspicious at the second level of analysis. This reduces the impact of processing overhead associated with the scheme.

Anomaly-based detection involves comparing observed activities with a pre-defined profiles considered as normal behaviour. If a behaviour deviates from the stored profile, then this behaviour is considered as an intrusion attempt. Hence, unlike signature-based detection, anomaly based detection schemes can detect unknown attacks. The detection schemes in this category is classified into three classes: machine learning, knowledge-based engines and statistical engines [10]. Machine-learning techniques can recognize complex patterns automatically and make intelligent decisions. The quality of the detection varies based on the learning algorithm and training dataset. This research uses the SVM algorithm for anomaly detection at the top level of the proposed IDS. The knowledge-based engines are composed of a knowledge-base that contains the pre-defined profiles and an inference engine that use the knowledge-base and rules to deduce the claimed behaviour. The statistical engines determine how far the observed behaviour is deviating from the previously measured threshold in terms of metrics such as CPU usage, consumed network bandwidth, number of service invocations, etc. Anomaly-based IDS can identify previously unknown attacks as it does not rely on a database with exact signatures, but the rate of false positives is higher [11].

C. *Machine Learning Algorithms for IDS*

Machine learning algorithms are suitable for training an IDS based on past intrusion detection dataset and prevent similar intrusion in the future. However, the suitability of an algorithm and the training dataset are crucial. Machine learning algorithms can be broadly classified into four categories: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning algorithms. Supervised learning include learning from pre-labelled classified dataset and making prediction using classification, regression or forecasting [12]. Unsupervised learning uses clustering and dimension reduction techniques to model the co-relation and relationships within the dataset in the absence of any pre-labelled data. Semi-supervised algorithms can process and learn from dataset which are partially labelled generally due to the unavailability of complete classified dataset or where the cost and time of full classification outstrips the benefit. The reinforcement learning category of algorithms uses feedback in the form of cumulative response /reward from its actions on the environment to adapt to the ideal behaviour for a specific context.

The proposed IDS scheme is based on SVM for anomaly detection at the first level of the two-tier hierarchical IDS. Here the SVM engine matches the SDN traffic flows against its database of conforming flows. Out of profile flows are then passed to the next level for more detailed packet level inspection. There are several advantages of using SVM over the other algorithms. SVMs do not require a reduction in the number of features in order to avoid over fitting--an apparent advantage in applications such as intrusion detection. Another primary advantage of SVMs is the low expected probability of generalization errors. SVMs also have significantly shorter training time in comparison to other suitable supervised algorithms such as neural networks [ref - Mukkamala, Janoski and Sung [13]. This makes it ideal choice for building efficient classifiers that can detect intrusions and unknown attacks in real time environments. Another advantage of SVM is scalability: SVMs are relatively insensitive to the number of data points and the classification complexity does not depend on the dimensionality of the feature space, so they can potentially learn a larger set of anomaly patterns and thus be able to scale better than neural networks. SVM also performs well on data sets that have many attributes, even if there are very few cases on which to train the model. There is no upper limit on the number of attributes, the only constraints are those imposed by hardware. This enables SVM for higher rate in IDS than other similar algorithms.

In its basic form, SVM is limited to only supporting binary classifications unlike neural network which can provide deeper sub classification of the attacks. However, suitable optimizing algorithm can be used if necessary for further feature sub-classification by choosing suitable kernel function. This, however, is resource intensive. Nevertheless, SVM's superior properties of fast training, scalability and generalization capability give them an advantage in the IDS.

III. Architecture of the Proposed Scheme

The proposed hierarchical IDS scheme is composed of both the advantages of a flow-based IDS and a packet-based IDS in order to provide good detection results with minimal impact on performance. The flow-based IDS acts as the first line of defence in the proposed IDS scheme. When an attack is detected further investigations is done by the packet- based IDS. The IDS application can run independently on top of the SDN stack and communicate with the controller via suitable northbound API (Fig. 1). This can make the IDS independent of the type of network controller used and the underlying network topology. In large networks, the intrusion detection processes at each level of the hierarchy can be run on a separate server to provide scalability without compromising the response time.

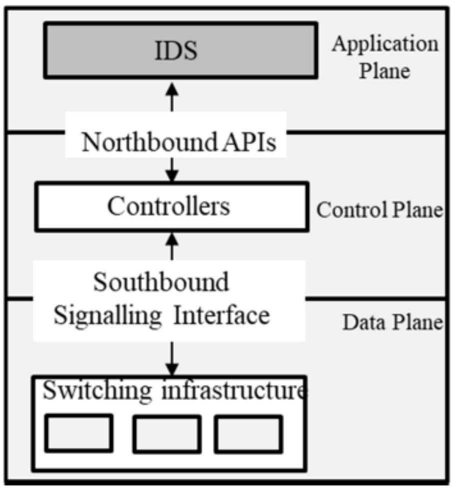

Fig. 1. The location of the proposed IDS in a SDN network

A. Components of the IDS

The proposed IDS is made of five modules (Fig. 2):

1) The Layer 2 learning switch is a mandatory module for the learning process of MAC addresses and the associated ports in the SDN network. Appropriate flow entries are inserted and deprecated flow are removed from the flow tables in order to reduce overhead. In addition, this module is responsible for the connection between the flow-based IDS and the packet-based IDS through a Unix Socket.

2) The Flow Aggregator is responsible for gathering flows statistics and ports statistics from OpenFlow switches in the SDN network. . This module makes periodical requests to the switches via the controller to retrieve flow statistics, port statistics and aggregated statistics. OpenFlow switches responds with corresponding reply event messages. The flow statistics reply contain information on source/destination MAC and IP addresses, protocol, byte count, packet count, duration, etc.; the port statistics reply contain information on send and received packets, bytes, packets dropped, frame and CRC errors, collisions, etc.; and the aggregate statistics reply contains the overall aggregated byte count and the packets count processed by a switch. In the proposed IDS, only the flow statistics are used by the detection algorithm. The two others messages are used to assess the performances of the solution (see section IV). The flow and port information are collected, normalised and structured and passed to the Flow Logger and Extractor module.

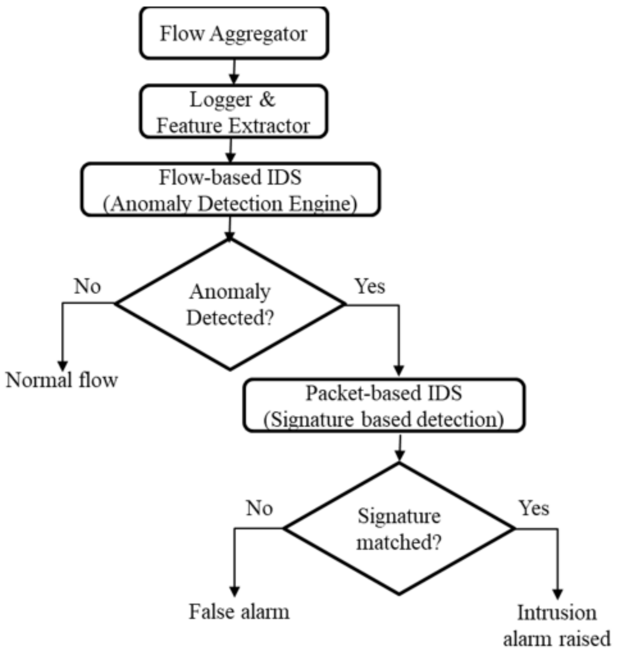
Flow Aggregator
Logger & Feature Extractor
Flow-based IDS (Anomaly Detection Engine)
Anomaly Detected?
No
Yes
Normal flow
Packet-based IDS (Signature based detection)
Signature matched?
No
Yes
False alarm
Intrusion alarm raised

Fig. 2. The components of the proposed IDS

3) The Flow Logger and Feature Extractor module processes the flow information passed by the Flow Aggregator and computes 6 tuples based on [14]. These include average number of packets per flow, average number of bytes per flow, average duration of a flow, percentage of symmetric paired flows, rate of increase in the number of single flows and growth of new ports. These six key signature feature captures relevant flow characteristics which is then passed to the flow-based IDS. In addition, this module also logs all flow information into log files for future analysis.

4) The Flow-based IDS acts as the first line of defence in the proposed IDS scheme. It operates on all traffic entering the network. An anomaly-based scheme is used as the detection algorithm. As discussed in section II.B, compared to signature-based detection, anomaly based scheme can detect unknown attacks and is comparatively lightweight with low overhead. This is particularly suitable in this case since it operates on all traffic entering the network. The chosen detection algorithm uses machine learning through SVM. The classification and prediction of flows use the 6-tuples received from the Flow Extractor and Logger. The inherent model of the SDN network with its support for flow level probing and statistics gathering is exploited by the anomaly based scheme for computing the tuples used for classification.

When a suspected intrusion is detected by the flow-based IDS, it would request a deeper analysis by mirroring and redirecting the corresponding flow to a packet-based IDS. This reduces the amount of false positive prevalent in an anomaly-based scheme as discussed in section II.B. The flow-based IDS also records and logs flows information, alarms and packets for further analysis.

5) The Packet-based IDS acts as the second level of defence in the proposed hierarchical system. It uses a signature-based intrusion detection that analyses only packets from flows considered as suspicious by the flow-based IDS. It has got a comparatively high overhead and processing delay because of packet by packet processing. However, since it only operates on a small fraction of the traffic, its effect is negligible on the overall system. This intrusion detection system can be any popular IDS (e.g. Bro, Suricata, Snort, etc.). There are trade-offs of using one over the other. For example, Suricata provides more detailed alerts compared to Snort but it has considerable higher processing overhead [15]. In the current prototype, Snort was chosen. Snort can operate in three modes: In the *sniffer mode*, it reads packet from the NIC and dumps them on the standard output; in *logger mode* Snort can store the trace files on disk in various format ranging from flat ASCII, XML, database, etc. for later analysis; and in *IDS mode* Snort can be configured with a variety of intrusion detection signature rules and the network traffic cross-checked against those rules for potential intrusion. In the proposed system, Snort has been configured in the IDS mode (see section IV.A). After analysing the traffic Snort sends back results in the form of alerts. These alerts could then be used by the controller to remove suspicious flows from flow tables of the target switches using OpenFlow signalling primitives.

IV. EXPERIMENT SETTINGS

A. Testbed

The cloud network topology was constructed using the Mininet network simulator [16]. It consists of eight OpenFlow v1.3 enabled switches with five switches forming edge of the network and each connected to 16 server hosts (Fig. 3). Addressing of the hosts is based on the DARPA dataset (section IV.B). The switches in the infrastructure plane is connected to the Ryu SDN controller [17]. Ryu was primarily selected because of its support for REST API and for being Python native. However, for a cloud level data center implementation other controllers can be suitable (Table 1). The Flow Aggregator, Logger and Flow-based IDS were implemented as northbound application on top of the Ryu controller. The SVM classification algorithm was implemented using scikit-learn [18] Python library. This part of the topology was setup on an Ubuntu virtual machine (VM).

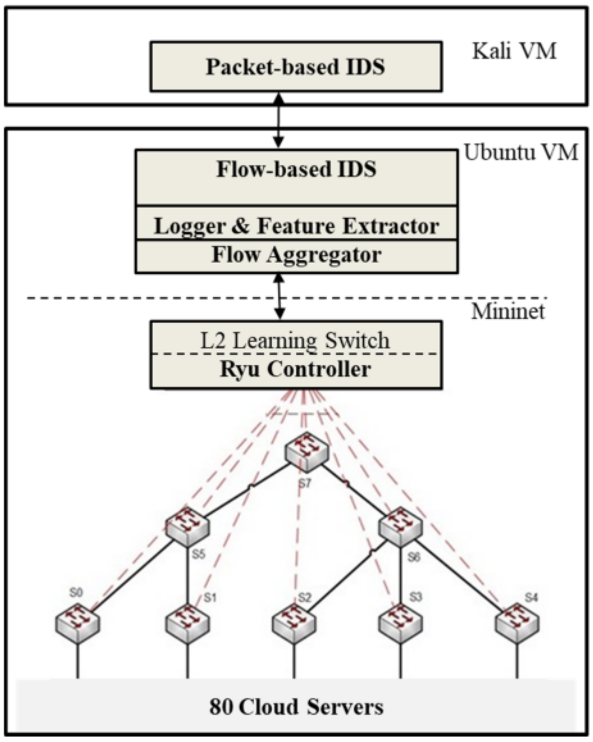

Fig. 3. The topology of the testbed

The packet IDS, Snort, was running on a separate Kali Linux 2.0 VM and communicating with the controller through a Unix Domain TCP socket using pigrelay [19]. Snort is installed as a classical configuration along with optional modules in order to simplify administration and maintenance. PulledPork [20] updates Snort with the latest available signatures for up-to-date detection. When the flow-based IDS detects an attack, the packet-based IDS receives the mirrored copy of the malicious flow to analyse. After analysis, the result is sent back to the controller in the form of alerts. The controller can then remove any malicious flow from the flow table of the corresponding switch. In addition, Barnyard2 [21] converts the binary unified2 logs produced by Snort into readable events that are then inserted into a MySQL database and can then be viewed on Snorby by network administrators for analysis. [22].

B. Dataset

Selecting the right dataset is a critical part because the overall efficiency of the proposed IDS relies on it, both for training and testing. Producing a dataset from scratch is time consuming, prone to errors and could be biased [23]; hence a subset of the DARPA dataset from the MIT Lincoln Laboratory was used [24]. This dataset is widely used for IDS research [25]. The dataset consists of three weeks of training data and two weeks of testing data. The week one and three from the training data are composed of normal traffic. The week two of training data consisted of some of the common attacks seen in a cloud data center which includes: Probe - scanning the network or host to retrieve IP addresses, ports, OS versions, etc. ; Denial of Service (DoS) - disruption of host or network; Remote to Local (R2L) - gains access to local machine remotely; User to Remote (U2R) - elevation of privileges; and Data attacks - extraction of non-authorized data.

C. Evaluation criteria

The efficiency of the proposed IDS is analysed in terms of its capabilities in detecting attacks and the overheads used in the process such as computing resources and false alarms. In that context, the following evaluation criteria has been used:

The *Detection Rate (DR)* gives a measurement of the fraction of attacks that has been detected and is computed as:

$$DR= \frac{TP}{TP+FN} \tag{1}$$

where *True Positive (TP)* is the number of attacks detected and *False Negative (FN)* is the number of attacks missed by the IDS. Their sum gives the total number of attacks in the system. DR provides a measurement of the accuracy of the system.

The False Alarm Rate (FAR) gives a measurement of the proportion of false alarm and is computed as:

$$FAR= \frac{FP}{TP+FP} \tag{2}$$

where *False Positive (FP)* is the amount of normal traffic classified incorrectly as malicious. A high false alarm rate results in unnecessary removal of ordinary legitimate flows and has additional overhead of further analysis and processing on the part of network administrator.

The *Error Rate (ER)* gives a measurement of the overall incorrect classification by the system and is represented as:

$$ER= \frac{FP+FN}{TP+TN+FP+FN} \tag{3}$$

where *True Negative (TN)* is the amount of correctly detected ordinary flows and the sum of the denominator represents the total number of flows in the system.

The additional overhead of the IDS on the performance of the SDN network is evaluated by the processing overhead to compute the periodical flow statistics requests sent to switches which is then used by the IDS for intrusion detection. The overhead is calculated as follows:

$$Overhead= \frac{n \text{ x Frame size}}{\text{Total Traffic}} \tag{4}$$

where *n* corresponds to the number of flow statistics events captured within the testbed by the controller. Wireshark exposes a frame size of 162 bytes for these events. Finally, the *Total Traffic* is the total number of bytes sent by OpenFlow switches during the evaluation.

V. RESULTS

Before replaying weeks four and five of the dataset, the data was converted from the tcpdump format into the pcap format to replay the file with tcpreplay. The controller and the simulated Mininet network were launched in separate terminals on the Ubuntu Machine.

A. Performance Evaluation

Table I and II shows the breakdown of the results of running the dataset of week four and five respectively on the IDS. The *Flow-based IDS* column shows only the results from the anomaly based SVM whereas the *Packet-based IDS* column shows only the results from processing the packets classified as malicious by the flow-based IDS. The *Flow+Packet-based IDS* shows the overall results from the proposed hierarchical IDS. Although the subset datasets of the two weeks are quite varied, the performance in terms of *DR*, *FAR* and *ER* shows identical trends.

TABLE I. WEEK 4 RESULTS

	Flow-based IDS	**Packet-based IDS ***	**Flow + Packet based IDS**
TN	588	84	672
TP	1,377	1,369	1369
FN	259	8	267
FP	146	62	62
DR	84.17%	99.42%	83.68%
FAR	9.59%	4.33%	4.33%
ER	17.09%	4.60%	13.88%

TABLE II. WEEK 5 RESULTS

	Flow-based IDS	**Packet-based IDS ***	**Flow + Packet based IDS**
TN	647	22	669
TP	461	448	448
FN	71	13	84
FP	58	36	36
DR	86.65%	97.18%	84.21%
FAR	11.18%	7.44%	7.44%
ER	10.43%	9.44%	9.70%

[a] only flows identified as suspicious by flow-based IDS are processed

The *DR* of the packet-based IDS is high for both the weeks (99.42% and 97.18% respectively) as it only operates on traffic classified as malicious by the flow-based IDS. The *DR* of the flow+packet IDS is slightly lower compared to only the flow-based part (difference of 0.049% and 2.44% respectively) is a small penalty for the performance gain in terms of *FAR* (improvement of 5.25% and 3.74% respectively for the two weeks) and *ER* (reduction by 3.2% and 0.73% respectively) which has a bigger impact in a cloud data center. Table III shows the cumulative performance of the IDS over the two weeks.

TABLE III. OVERALL RESULTS

	Flow-based IDS	Packet-based IDS *	Flow + Packet based IDS
TN	1235	106	1341
TP	1838	1817	1817
FN	330	21	351
FP	204	98	98
DR	84.78%	98.86%	83.81%
FAR	9.99%	5.12%	5.12%
ER	14.80%	5.83%	12.45%

B. Overhead

The overhead generated by periodical flow statistics messages required by the flow-based IDS has a minimal impact on the performance (Table IV) with only an overhead of 0.88% and 0.56% over week four and five respectively. OpenFlow also has a number of additional state, port and aggregate signalling messages but these are not used in the IDS and hence not used here for computing the IDS specific overhead.

TABLE IV. OPENFLOW SIGNALLING OVERHEAD

	Week 4	Week 5
EventOFPFlowStatsReply	1,748	752
Total traffic (bytes)	32,048,629	21,587,087
Overhead	0.88%	0.56%

VI. CONCLUSION

Compared to related research, the dataset used here had a very low number of redundant records making intrusion detection more challenging. Still, the results shows that the proposed hierarchical IDS scheme has a good detection rate, a low false positive and error rate and operates with a negligible overhead. The research provides an architectural framework of integrating different categories of intrusion detection schemes hierarchically by exploiting the strengths of individual schemes and addressing their weaknesses at the next level of the hierarchy to provide an overall robust IDS system. One notable aspect of the implementation is the modularity of the components and the bi-directional communication channel via the centralised controller. Both the SVM learning engine and the Snort module can be readily replaced based on the scenario and requirements. Moreover, to achieve higher scalability in an enterprise environment, the controllers can be distributed and the individual northbound applications can be hosted on dedicated servers.

REFERENCES

[1] A. Singh, et. al., “Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network”, In Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM '15). ACM, New York, NY, USA, vol. 45, no. 4, pp. 183-197.

[2] K. Scarfone and P. Mell, “Guide to intrusion detection and prevention systems (idps)”, NIST special publication, vol. 800, no. 2007, pp. 9–14, 2007.

[3] Snort, Network intrusion detection and prevention system, 2016. [Online], https://www.snort.org/ (accessed: 11.06.2018).

[4] Suricata, Open source ids / ips / nsm engine. [Online]. Available: https://suricata-ids.org/ (accessed: 07.06.2018).

[5] Á. L. Valdivieso Caraguay, A. Benito Peral, L. I. Barona López, and L. J. García Villalba, “SDN: Evolution and Opportunities in the Development IoT Applications”, International Journal of Distributed Sensor Networks, vol. 2014, pp. 1–5, 2014.

[6] Open Networking Foundation: OpenFlow switch specification. https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf, (2013) (accessed: 07.06.2018).

[7] Bro, The bro network security monitor, 2014. [Online]. Available: https://www.bro.org/ (accessed: 07.06.2018).

[8] M. Alenezi and M. J. Reed, “Methodologies for detecting dos/ddos attacks against network servers”, in The Seventh International Conference on Systems and Networks Communications ICSNC, 2012, pp. 92–98.

[9] D. Alsmadi IzzatXu, “Security of software defined networks: A survey”, Computers and Security, vol. 53, pp. 79–108, 2015.

[10] V. Kumar, J. Srivastava, and A. Lazarevic, Managing cyber threats: Issues, approaches and challenges. Springer Science & Business Media, 2006, vol. 5.

[11] M. Alenezi and M. J. Reed, “Methodologies for detecting dos/ddos attacks against network servers,” in The Seventh International Conference on Systems and Networks Communications ICSNC, 2012, pp. 92–98.

[12] C.F. Tsai, Y.F. Hsu, C.Y. Lin and W.Y. Lin, “Intrusion detection by machine learning: A review. Expert Systems with Applications; vol. 36(10):11994-2000, 2009.

[13] M.J. Kang, and J.W. Kang, Intrusion detection system using deep neural network for in-vehicle network security. PloS one, 11(6), p.e0155781, 2016.

[14] R. Braga, E. Mota and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," IEEE Local Computer Network Conference, Denver, CO, pp. 408-415, 2010.

[15] B. Brumen and J. Legvart, "Performance analysis of two open source intrusion detection systems," 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO 2016), Opatija, pp. 1387-1392, 2016.

[16] Mininet, An instant virtual network on your laptop, 2016. [Online]. Available: http://mininet.org/ (accessed: 26.05.2018).

[17] RYU the Network Operating System(NOS), Available: https://ryu.readthedocs.io/en/latest/index.html (accessed: 15.05.2018).

[18] S. Jain, et. al., “B4: Experience with a globally-deployed software defined wan”, SIGCOMM Comput. Commun. Rev., vol. 43, no. 4, pp. 3–14, Aug. 2013.

[19] PigRelay, 2015. [Online]. Available: https://github.com/John-Lin/pigrelay (accessed: 15.05.2018).

[20] Pulledpork, Pulled pork for snort rule management, 2016. [Online]. Available: https://github.com/shirkdog/pulledpork (accessed: 15.05.2018).

[21] Barnyard2, 2016. [Online]. Available: https://github.com/firnsy/barnyard2 (accessed: 4.06.2018).

[22] Snorby, Network security monitoring tool, [Online]. Available: https://github.com/Snorby/snorby (accessed: 7.06.2018).

[23] C. Jeong, T. Ha, J. Narantuya, H. Lim, and J. Kim, “Scalable network intrusion detection on virtual sdn environment”, in Cloud Networking (CloudNet), IEEE 3rd International Conference on, 2014, pp. 264–265.

[24] Lincoln Laboratory, Massachusetts Institute of Technology, Darpa intrusion detection data sets, 1998. [Online]. Available: https://www.ll.mit.edu/ideval/data/index.htm (accessed: 2.05.2018).I.

[25] C. Thomas, V. Sharma, and N Balakrishnan, “Usefulness of darpa dataset for intrusion detection system evaluation”, in SPIE Defense and Security Symposium, International Society for Optics and Photonics, 2008, 69730G–69730G

[26] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.