INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2016, VOL. 62, NO. 3, PP. 247-252
Manuscript received May 5, 2016; revised June, 2016. DOI: 10.1515/eletel-2016-0033

On Efficiency of Selected Machine Learning Algorithms for Intrusion Detection in Software Defined Networks

Damian Jankowski, Marek Amanowicz

***Abstract*—We propose a concept of using Software Defined Network (SDN) technology and machine learning algorithms for monitoring and detection of malicious activities in the SDN data plane. The statistics and features of network traffic are generated by the native mechanisms of SDN technology. In order to conduct tests and a verification of the concept, it was necessary to obtain a set of network workload test data. We present virtual environment which enables generation of the SDN network traffic. The article examines the efficiency of selected machine learning methods: Self Organizing Maps and Learning Vector Quantization and their enhanced versions. The results are compared with other SDN-based IDS.**

***Keywords*—Software Defined Network, intrusion detection, machine learning, Mininet**

I. Introduction

Software-defined network (SDN) is a modern approach to networking that abolishes the complexity and static nature of legacy distributed network architectures. It is achieved through the use of a standards-based software abstraction between the network control plane and underlying data forwarding plane, including both physical and virtual devices. This gives an opportunity to use the native SDN functionality for monitoring malicious activities within the data plane. The SDN-based intrusion detection system can be considered as an additional monitoring mechanism, besides the classic security solution [1] [2]. In our approach, the parameters and statistics, extracted from the fine-grained SDN flows, create tuples of features that are classified by detection mechanisms. It allows to identify a specific connections representing different network activities. However, selection of appropriate machine learning classification technique enabling proper identification of particular types of malicious traffic become an important issue for our approach. The variety proposals for using IDS in IP-based networks are presented in many papers (e.g. [3][4][5][6]) but only few publications deal with the implementation of this technique in Software Defined Networks [7][8][9][10][11][12]. In this paper we present the results of evaluation of selected machine learning algorithms and provide recommendations on their use for intrusion detection in SDN environment.

This paper is organized as follows. Chapter II describes the architecture of our approach. The third chapter introduces the selected machine learning methods SOM and LVQ1. Chapter IV presents the results of the experiments and contains the comparison of performance of our conception with other SDN-based IDS methods. The paper is summarized with some conclusions.

II. System Architecture

We propose a concept of using the native features of SDN technology for monitoring and detection of malicious activities, which we call MADMAS. The intrusion detection system consists of 4 main modules as presented in Fig. 1.The Flow Bundle Module extracts traffic statistics from the fine grained data flows. The Integrator collects and processes the traffic statistics enabling generation some additional features. These two modules compose the Features Generator (FG). It outcomes are passed to the Machine Learning-based Classifier, which is responsible for detection of the malicious activities in the data plane. Controller supervises operations performed by the modules and processes the results of the traffic classification for visualization. At this stage of our study the Classifier and Controller reside on the workstation.

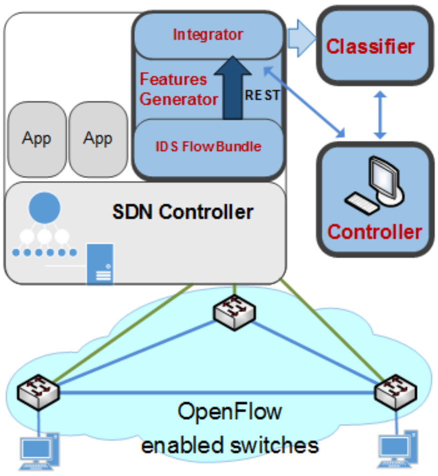

Fig. 1. SDN-based intrusion detection architecture

We assumes that the features of network traffic are measured by the native mechanisms of SDN technology. The Flow Bundle Module works at the SDN Opendaylight controller as an OSGi bundle.[13] The flows are matched base on the following parameters:

- destination IP address,

This work was supported by the research project PBS 17/WAT.

All the authors are with the Institute of Telecommunication, Faculty of Electronics, Military University of Technology, Poland (e-mail: {damian.jankowski, marek.amanowicz}@wat.edu.pl).

- source IP address,
- destination TCP/UDP port,
- source TCP/UDP port,
- protocol type (ARP, IP, TCP, UDP or unknown).

For each flow an idle timeout parameter is set. It defines the period after the entries are deleted from the flow table. In addition, an identification number is assigned to each flow. This way of matching traffic, distinguishes flows in terms of different port numbers and IP addresses. The exemplary network activity may consist of multiple flows. In presented approach the collected features are used as input vector for detection mechanism. For each flow, a set of parameters is determined. For classification purposes, the input vector $X(x_1, x_2, \ldots, x_9)$ is represented by:

x_1 -packet count in a flow,
x_2 -bytes count in a flow,
x_3 -destination TCP/UDP port,
x_4 -source TCP/UDP port,
x_5 -duration,
x_6 -flows with different ports from source host,
x_7 -flows with same ports to destination host,
x_8 - flow rate to the host,
x_9 -single flow rate to the host.

III. Selected machine learning algorithms

Let us consider the following algorithms that can be used for classification of malicious activities in the SDN data plane, i.e:

- Self-Organizing Maps (SOM)
- Multi-pass Self-Organizing Maps (M-SOM),
- Learning Vector Quantization (LVQ1),
- Multi-pass Learning Vector Quantization (M-LVQ1)
- Hierarchical Learning Vector Quantization (H-LVQ1).

Above algorithms are types of artificial neural networks (ANN) that is trained using unsupervised (SOM based) or supervising (LVQ1 based) learning technique. In response to input signals, network indicates the activation of neurons in varying degrees. Neurons (nodes) compete for the right to respond to a subset of the input data. The neuron whose weight vector is most similar to the input is called the best matching unit (BMU) or best matching neuron (BMN). A distance measure between input patterns must be defined, in our case it is a Euclidean distance (see equation 2). In the presented algorithms, it is necessary to carry out an initialization phase. This is the arrangement of specific positions of neurons in the considered space. Typically, the initial distribution of neurons can be created in a random way [14].

Neurons in SOM can be associated with its other 6 neighbours in a hexagonal manner. The most stimulated neuron and neighbouring neurons update the weights in response to learning vectors.

$$W_i(k+1) = W_i(k) + \eta_i G(r)[X - W_i(k)] \quad (1)$$

where: X – input vector of features, W_i – i weight vector of the neuron at k time, η_i – learning rate, $G(r)$ - neighbourhood function (3).

The distances of input vector $X(x_1, x_2, \ldots, x_j)$ to winner neurons $W(w_1, w_2, \ldots, w_j)$ are calculated on base of the Euclidean distance.

$$d(X, W_i) = \|X - W_i\| = \sqrt{\sum_{j=1}^{N}(x_j - w_{ij})^2} \quad (2)$$

where: X – input vector of features, x_j – j feature in X input vector, W_i - i weight vector of the neuron, w_{ij} – j value of weight in i weight vector of the neuron.
The degree of weight adaptation $G(i)$ of winner and neighbourhoods neurons is calculated by Gaussian formula.

$$r = d(i, W) = \|i - W\| = \sqrt{\sum_{j=1}^{N}(i_j - w_j)^2} \quad (3)$$

$$G(r) = \exp\left(-\frac{r^2}{2\lambda^2}\right) \quad (4)$$

where: r - Euclidean distance of i neuron from winner neuron, W –winner neuron, λ– neighbourhood radius.

During the learning process , weights of neurons are adapted to learning input vector. In other words, neurons or groups of neurons, are activated in response to stimulation, adapting to the form of specific patterns. When input vectors are labelled, SOM can be used as a classification mechanism. The SOM network allows to create a type of structure, which can represent input vectors in the best way. It can be said, that the single neuron represents many vectors from the dataset. The class is assigned to the neuron with the consideration of which class is the most numerous, from stimulating vectors. The classification step is preformed after learning. During the network testing, the test vectors activate neurons of a trained network which are the most similar. This involves determining which labelled neuron is activated under the input vector. After the process of learning, SOM network can be presented to low dimension space by Sammon mapping and visualised by U-matrix, U*-matrix, P-matrix. [14][15].

Multi-Pass SOM is the implementation of the SOM algorithm where two passes are performed on the same underlying model. The first pass is a rough ordering pass with large neighbourhood radius, learning rate and small training time. The second pass is the fine tuning pass that has a longer training time, small initial neighbourhood radius value and smaller initial learning rate [16].

Learning Vector Quantization (LVQ) may also be considered as special case of an artificial neural network architecture, learned in a supervised way. The LVQ network has a set of units and weight vectors W_i associated to them. In this paper, we consider 3 versions of LVQ1 for traffic classification. In LVQ1 each input vector has a class assigned to them that the network would like to learn. At step k, given a vector X randomly chosen from the input data. Then the nearest W_i vector is selected, according to the Euclidean distance $d(X, W_i)$ given by (2). After that the vector W_i is updated in the following way:

$$W_i\ (k{+}1) = \begin{cases} (W_i\ (k) + \eta_i\ (X{-}W_i\ (k)), & \textit{if X in the same class as } W_i \\ W_i\ (k) - \eta_i\ (X{-}W_i\ (k)), & \textit{if X in the other class than } W_i \end{cases} \quad (5)$$

where: X – input vector of features, $W_i(k)$ – i weight vector of the neuron at k time, η_i – learning rate [17].

In Multi-Pass, the quick rough pass is made on the model using LVQ1 with relative large learning rate, then a long fine tuning pass is made on the model with LVQ1 and smaller learning rate. In Hierarchical LVQ implementation each codebook vector is treated as a cluster centroid. All codebook vectors are evaluated and part of that vectors are selected as candidates for sub-models. The sub-models are constructed for all candidate codebook vectors and those sub-models that outperform their parent codebook vector are kept as part of the model. During testing, a dataset tuple is first mapped onto its BMU, if that BMU has a sub-model, the sub model is used for classification, otherwise the class value in the BMU is used for classification [16].

IV. EXPERIMENTS

A. *Dataset generation*

The architecture of the testbed we used for evaluation of machine learning algorithms and their applicability for intrusion detection in SDN environment is shown in the Figure 2. A simplified SDN network is emulated in the Mininet [18], while the server is emulated by Metasploitable 2 virtual machines with the Ubuntu operating system. Vulnerabilities in services and operating systems, default passwords and misconfigurations are intentionally left on the server environment. The clients generate requests to the server, at the same time, the malicious host performs unauthorized activities directed to servers by using attack tools. The course of emulation is automated by Python scripts. Generated traffic is probing by the measurement module. The servers reside on separate virtual machines and clients are virtualized at the level of Mininet OS. In order to achieve the most realistic character of the generated attacks, malicious activity are conducted using special tools (see Tab I). Each class of such traffic has subclasses, which define the detailed course of action, types of attack tools or exploits that are directed at the network or server resources.

For instance, the malicious hosts perform a flooding attack on the SDN network by Nping tool with specific parameters. These events have an impact both on the SDN controller performance and the available data plane resources, and can cause delays in processes of matching flows. The probe class includes attacks that are intended to obtain information about the object of attack. These attacks include ports, version, services or vulnerability scanning. Such malicious activities performed by Metasploit or Nmap tools give information to the intruders about the potential targets of the attacks. This kind of activity may be a preliminary phase of the main attack, i.e. DoS or buffer overflow.

The U2R class includes network activities related with the back doors and remote exploitation attacks. The attacks are performed by Metasploit Framework scripts and commands. These malicious activities are carried out against the vulnerable services, which are used in normal traffic. Therefore, these attacks are characterised by a high degree of similarity to the short duration normal traffic. Firstly, the malicious requests prepared by Metasploit are sent to the vulnerable service. The payload contains the exploit and shellcode for the specified service. After the exploiting operation, the malicious host gains access to the shell with root privileges. At the next step, the malicious host establish the connection to the exploited service, with the reverse shell and execute a few Linux commands. The type of exploit and detailed course of the attack may vary for individual services. For instance, before exploit steps, the user login may take place. Each stage of the attack is reflected in SDN fine granulated flows.

The R2L class includes the credentials guessing and the unauthorised access to IT accounts. The password guessing is conducted in the form of dictionary attack. The potential passwords and logins are stored in external file. The malicious hosts try to authorise with parameters from lists of credentials. Sequential requests are sent to the service. When the authorisation succeeds, the corresponding credentials are stored. These activities generate moderate number of flows.

TABLE I
CLASSES OF NETWORK ACTIVITIES

Classes of traffic	Description of activities	Tools for traffic generation
normal	Traffic between clients and servers	Clients and servers of following services FTP, SSH, SMB, Apache, Web, Tomcat, RMI Ruby, Java RMI, Postgres, Telnet
probe	Port probe, vulnerability scan, version scan	Metasploit, Nmap
R2L	Credentials guessing	Metasploit, Hydra
DoS	Denial of service attacks	Metasploit, Hping3, Nping
U2R	Remote exploits, backdoors,	Metasploit

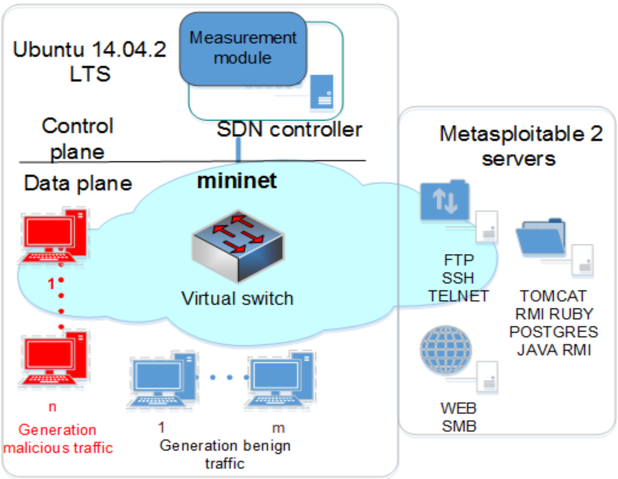

Fig. 2. Testbed architecture

B. *Evaluation Methodology*

All our experiments were performed using the WEKA with additional plugin [16][19]. For testing purposes, 10 fold cross validation was used. The features of input vector X are normalized in the range [0,1]. The formulas below show the metrics used for evaluation of classification models, i.e.:

- True Positive Rate

$$TPR = \frac{TP}{TP + FN} \tag{6}$$

where: TP – True Positive, FN – False Negative ,

- False Positive Rate

$$FPR = \frac{FP}{FP+TN} \tag{7}$$

where: FP – False Positive, TN – True Negative,

- Precision or Positive Predictive Value

$$PPV = \frac{TP}{TP+FP} \tag{8}$$

The conformity of the neural net with the input data was assessed by calculation of the average quantization error according to the following formula [20]:

$$E_t = \frac{1}{N}\sum_{i=1}\left\|X_i - M_c\right\| \tag{9}$$

where: X_i – input vector, M_c – best matching neurons (BMN) The best neural net is expected to have the smallest average quantization error.

C. *Results*

The overall results are summarized in Table II. The analysis indicate that it is possible to achieve an average value of TPR greater than 94%. However, in constructed models, size of the networks exceeds the number of 800 neurons. H-LVQ1 algorithm is an effective way to improve TPR, precision and FPR compared to SOM, M-SOM, LVQ1, M-LVQ1 for all classes. Class U2R has the worst TPR and FPR metrics for all classification algorithms. The best TPR and PPV results are achieved for Probe and DoS classes. M-SOM and M-LVQ algorithms slightly improve efficiency in comparison to LVQ and SOM. Moreover, there is the visible advantage of H-LVQ1 in efficiency for all classes.

TABLE II
EFFICIENCY OF SELECTED ALGORITHMS

Evaluation metrics	SOM	Multipass SOM	LVQ1	Multipass LVQ1	Hierarchical LVQ1
TPR [%]	94,4	94,6	95,6	95,6	98,1
FPR [%]	3,9	3,9	3,2	3,1	1,9
PPV [%]	93,8	94,2	95,2	95,3	98
Total Model Preparation Time [ms]	2151	4502	634	820	920

TABLE III
TRUE POSITIVE RATE [%] PER CLASS

TP Rate of class	SOM	Multipass SOM	LVQ1	Multipass LVQ1	Hierarchical LVQ1
Normal	97,8	98,0	98,1	98,1	98,6
Probe	96,3	96,1	96,1	96,2	98,7
R2L	74,8	77,6	83,5	83,4	94,6
DoS	47,0	48,3	80,5	83,2	99,6
U2R	3,1	5,6	0,8	1,1	48,3

TABLE IV
FALSE POSITIVE RATE [%] PER CLASS

FP Rate of class	SOM	Multipass SOM	LVQ1	Multipass LVQ1	Hierarchical LVQ1
Normal	4,8	5	4,3	4,1	2,1
Probe	3,1	2,9	2	1,9	0,9
R2L	1	0,8	0,5	0,5	0,3
DoS	0,6	0,7	0,8	0,9	0
U2R	0	0	0	3,1	0,2

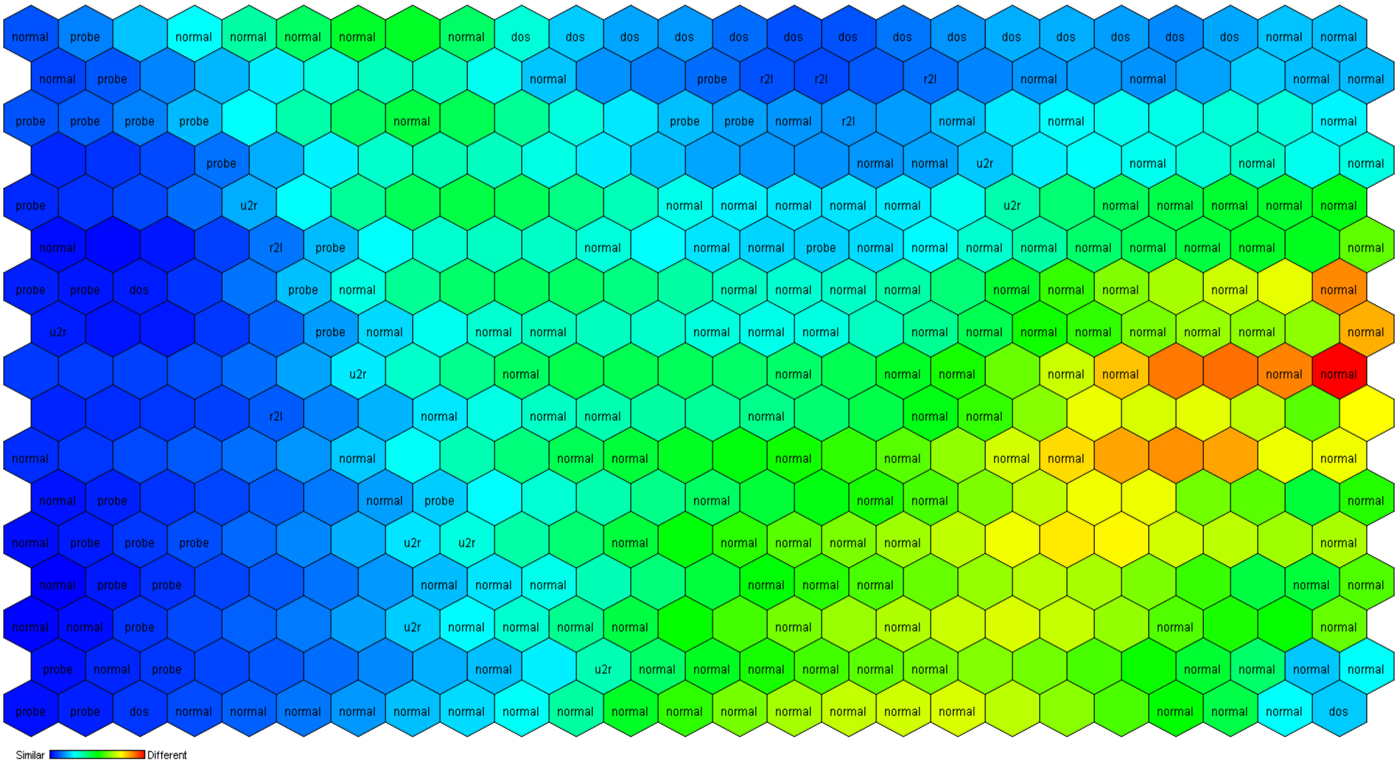

Fig. 3. Self-organizing maps U*-matrix visualization

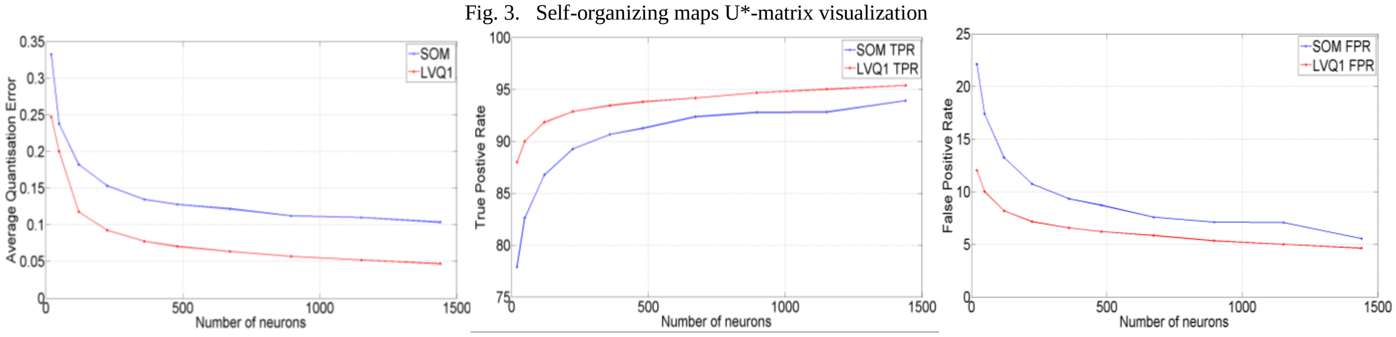

Fig. 4. Average Quantization Error , TPR, FPR versus the number of neuron

SOM and M-SOM methods are characterized by the highest time required to build the classification model. At the same time, SOM and M-SOM have the worst values of TPR and FPR. Visualization of U*-matrix, reflecting the average Euclidean distance between the codebook vectors of neighbouring neurons is shown in Fig 3. Let us consider 2 clusters, represented by the blue areas of the map. The first cluster at left side is represented by neurons mainly assigned to probe and normal class. There is also individual hexagons with DoS, R2L and U2R class. The second cluster is located at the top of the map. There is a preponderance of neutrons assigned to class DoS. In the second cluster, there are single hexagons with classes normal, R2L and U2R. The green area between clusters contains normal class. A small cluster in the lower right corner, includes class DoS and normal. This means that the classes are not well separated.

Fig 4 illustrates the Average Quantisation Error (AQE), TPR and FPR versus the number of neurons in the SOM and LVQ1. It is evident that LVQ1 has smaller AQE and FPR and bigger TPR then SOM for over the range of curves. As we can see, there is significant growth of efficiency to specified threshold (about 1000 neurons), above this level we do not get a significant increase in value of TPR, FPR and AQE.

Attack classes probe, DoS and R2L are characterized by the best TPR and FPR. The results indicate poor efficency for U2R class. The most likely explanation of the negative result is that the features generated from flows are not optimal for remote exploits attacks [21]. One possible solution is to develop additional methods of features extraction. To overcome this drawback, it is necessary to adapt a Deep Packet Inspection (DPI) technique.

D. *MADMAS Evaluation*

Table V presents the efficency of MADMAS in compare to other selected SDN-based IDS methods. We considered the following alternatives:

- Method 1 - Revisiting Traffic Anomaly Detection Using Software Defined Net-working [7],
- Method 2 - A Fuzzy Logic-Based Information Security Management for Software Defined Networks [8],
- Method 3 - Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments [9],
- Method 4 - Lightweight DDoS flooding attack detection using NOX/OpenFlow [10],
- Method 5 - Efficient Anomaly Detection And Mitigation In Software Defined Networking Environment [11],
- Method 6 - Flexible Network-Based Intrusion Detection and Prevention System on Software-Defined Networks [12].

It needs to be highlighted that verification of considered methods was carried out in different environments, according to various methodologies. Nevertheless the results presented in Tab. V can give a generic view on their efficiency. The considered methods can detect certain types of malicious activities, i.e.: denial of service, distributed denial of service port scan, but only our method and Method 5 detect U2R and R2L attacks (remote exploits, passwords guessing etc.). It is evident that MADMAS gives higher TPR values for DoS, Probe, U2R classes in compare to other solutions. Method 3 gives better results of TPR for Probe and DDoS attacks, however at high value of FPR (23-27%). It should be also noticed that efficiency of U2R detection by MADMAS is still too low that would require further works.

TABLE V
COMPARISON OF EFFICIENCY (TPR AND FPR IN [%])

SDN based IDS methods		DoS, DDoS	Probe, Scan	R2L	U2R
MADMAS	TPR	99,6	98,7	94,6	48,3
	FPR	2,1	0,9	0,3	0
Method 1	TPR	94	90	x	x
	FPR	0	0-4		
Method 2	TPR	95	x	x	x
	FPR	1,2			
Method 3	TPR	100	100	x	x
	FPR	27	23		
Method 4	TPR	99,11	x	x	x
	FPR	0,46			
Method 5	TPR	90,9	91,9	80,2	98,1
	FPR	0,1	0,24	0,69	0,88
Method 6	TPR	96,4	92,1	x	x
	FPR	x	x		

V. CONCLUSIONS

In the paper we presented the convincing concept of detection of malicious activities in SDN data plane. We show the benefits of using MADMAS for identification the selected threats and its advantage over other considered solutions. However, an additional work has to be done to improve the efficiency of detection of U2R attacks that would include implementation of deep packet inspection technique. The obtained results indicate also some advantage of using the Hierarchical LVQ1 in compare to other techniques. On the basis of the promising findings presented in this paper, work on the remaining issues is still continuing. The next stage of our research will focus on improving of features generation and on applicability of other statistical techniques for detection and classification of malware traffic. Further research on monitoring of traffic in SDN control plane is also planned. Therefore, it would allow to expand functionality of MADMAS to detect attacks against the controllers and management stations.

REFERENCES

[1] D. Kreutz, F. M Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," in *Proceedings of the IEEE 103.1*, 2015, pp. 14-76. doi:10.1109/JPROC.2014.2371999

[2] S. Hayward, Sandra, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," 2015. doi:0.1109/COMST.2015.2474118.

[3] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel and M. Rajarajan,. "A survey of intrusion detection techniques in cloud," in *Journal of Network and Computer Applications*, vol 36(1), 2013, pp. 42-57. doi:0.1016/j.jnca.2012.05.003

[4] H. J. Liao, C. H. R. Lin, Y. C.Lin, and K. Y. Tung, "Intrusion detection system: A comprehensive review," in *Journal of Network and Computer Applications*, vol. 36(1), 2013, pp. 16-24. doi:10.1016/j.jnca.2012.09.004

[5] N. F. Haq, A. R. Onik, M. Avishek, K. Hridoy, M. Rafni, F. M. Shah, and D. M. Farid, "Application of Machine Learning Approaches in Intrusion Detection System: A Survey," in *International Journal of Advanced Research in Artificial Intelligence*, 2015. doi:10.14569/IJARAI.2015.040302

[6] M. Kruczkowski, E. Niewiadomska-Szynkiewicz, and A. Kozakiewicz. "FP-tree and SVM for Malicious Web Campaign Detection," in *Intelligent Information and Database Systems*, Springer International Publishing, 2015, pp. 193-201. doi: 10.1007/978-3-319-15705-4_19

[7] M. S. Akbar, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *Recent Advances in Intrusion Detection*, Springer Berlin Heidelberg, 2011, pp. 161-180. doi:10.1007/978-3-642-23644-0_9

[8] S. Dotcenko, A. Vladyko, and I. Letenko, "A fuzzy logic-based information security management for software-defined networks," in *Advanced Communication Technology*, 16th International Conference on IEEE, 2014, pp. 167-171. doi:10.1109/ICACT.2014.6778942

[9] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," in *Computer Networks,* vol 62, 2014, pp. 122-136. doi:10.1016/j.bjp.2013.10.014

[10] R. Braga, E. Mota, A. Passito, "Lightweight DDoS Flooding Attack Detection Using NOX/OpenFlow," in *Local Computer Networks (LCN)*, 35th Conference on. IEEE, 2010. pp. 408-415. doi: 10.1109/LCN.2010.5735752

[11] R. Sathya and R. Thangarajan, "Efficient Anomaly Detection And Mitigation In Software Defined Networking Environment," in *Electronics and Communication Systems*, 2nd International Conference on IEEE, 2015, pp. 479-484. doi:10.1109/ECS.2015.7124952

[12] A. Le, P. Dinh, H. Le, and N. C. Tran, "Flexible Network-Based Intrusion Detection and Prevention System on Software-Defined Networks," presented at *International Conference on Advanced Computing* and Applications, November 2015, pp. 106-111. doi:10.1109/ACOMP.2015.19

[13] OpenDaylight Platform [Online]. Available: https://www.opendaylight.org/

[14] T. Kohonen, "Essentials of the self-organizing map," in *Neural Networks*, vol. 37, 2013, pp. 52-65. doi:10.1016/j.neunet.2012.09.01

[15] T. Kohonen, "The self-organizing map," in *Proceedings of the IEEE*, vol. 78(9), 1990, pp. 1464-1480.

[16] WEKA Classification Algorithms, A WEKA Plug-in, [Online]. Available: http://wekaclassalgos.sourceforge.net/

[17] T. Kohonen,, "Learning vector quantization," Springer Berlin Heidelberg, 1995, pp. 175-189.

[18] Mininet, An Instant Virtual Network on your Laptop (or other PC), [Online]. Available: http://minimet.org

[19] M. Hall, E. rank, G. Holmes, B. Pfahringer, P. Reutemann and I. H. Witten, "The WEKA data mining software: an update," in *ACM SIGKDD explorations newsletter*, vol. 11(1), 2009, pp. 10-18. doi:10.1145/1656274.1656278

[20] G. Pölzlbauer, "Survey and comparison of quality measures for self-organizing maps," 2004.

[21] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An overview of IP flow-based intrusion detection," in *Communications Surveys & Tutorials*, IEEE, 12(3), 2010, pp. 343-356. doi: 10.1109/SURV.2010.032210.00054