Support Vector Machine meets Software Defined Networking in IDS domain

Luca Boero, Mario Marchese, and Sandro Zappatore
Department of Electrical, Electronic and Telecommunications Engineering, and Naval Architecture (DITEN)
University of Genoa
Via all' Opera Pia 13, 16145 Genoa Italy
Email: luca.boero@edu.unige.it, mario.marchese@unige.it, sandro.zappatore@unige.it

***Abstract*—Intrusion Detection Systems (IDS) are aimed at analyzing and detecting security problems. IDS based on anomaly detection and, in particular, on statistical analysis, inspect each traffic flow in order to get its statistical characterization, which represents the fingerprint of the flow. Software Defined Networking (SDN) is revolutionizing the networking industry by enabling programmability, easier management and faster innovation. These benefits are made possible by its centralized control plane architecture which allows the network to be programmed and controlled by one central entity. The fusion of these two technologies can lead to an innovative system of malware detection. This paper tries to join these two concepts in order to obtain the best from the two worlds. We use a well known machine learning scheme (Support Vector Machine) as core system for detecting malware by using only traffic features that can be extracted using an SDN controller.**

***Index Terms*—intrusion detection system, malware detection, software defined networking, machine learning**

I. Introduction

Nowadays a lot of important applications such as public services, Internet banking, and also systems devoted to defense are dependent on networks and computers. For this reason they are often the target of malicious software (malware, spyware, etc...) attacks. Malware is software specifically designed to insert itself in a computer system without the approval of the owner using techniques such as trojans, backdoors, keylogger, and worms [1]. To prevent these type of attacks it is necessary to accurately detect malware and other type of intrusions [2]. An Intrusion Detection System (IDS) is a piece of hardware/ software designed to alert when someone or something is trying or has tried to compromise systems. In general it is possible to use IDS in order to reveal anomalies and tackle malicious intrusions. [3] proposes a classififcation of anomaly detection methods. In particular, concerning the processing method, [3] suggest: Misuse and Anomaly Detection. The former tries to fix the abnormal behavior and considers the rest as normal. The latter describes the normal behavior and marks as abnormal what is not considered normal. Operatively the former contains: signature based, rule based, state transition algorithms, and data mining. The latter includes: statistical, distance, profile, and model-based schemes. Misuse Detection (MD) systems, in order to collect signature and information of the flow under analysis, have to open each packet of the flow up to the application layer (deep packet inspection). This type of approach is often very efficient but it has also some limitations: for example, the signature of an attack can be dated, or, considering the processing time, to open each single packet can be computationally heavy. Anomaly Detection, and, in particular, statistical analysis based ones, which are taken as a reference in this paper, would like to avoid these drawbacks also at the cost of lower accuracy results: packets are not deeply inspected but each traffic flow is monitored over time by measuring the statistics of a set of variables (called features) to distinguish between anomalies (possible malware) and normal behavior (normal, not infected, traffic).

Software Defined Networking (SDN) [4] [5] is a recent networking architecture that decouples user and control plane. In practice SDN separates data and control actions operated by networking devices such as switches and routers. Data functions are located within devices, control functions are concentrated in SDN controllers. The communication between an SDN controller and the devices under its domain is implemented through a signalling protocol called OpenFlow. This paper proposes a novel Statistical Analysis SDN-based IDS. It uses the typical flow definition at TCP/IP (Transfer Control Protocol/Internet Protocol) layers and is aimed at deciding whether a flow is malware-affected or not under the framework of the SDN architecture. It is structured into a training phase developed by using a ground truth of known flows and an operative classification and decision phase. Both training and classification/decision phases are based on the definition and extraction of a group of statistical parameters related to each flow, which represent the Statistical Fingerprint of the flow and on machine learning-based classification devoted to distinguish normal from malicious traffic.

The paper is organized as follows: Section II contains the state of the art concerning deep packet inspection MD and Statistical Analysis-based Anomaly Detection. Section III describes the differences between an SDN and non-SDN approach. Section IV contains the proposed architecture. Section V explains the operative steps to implement the architecture and shows the results of the proposed approach. Section VI reports the conclusions.

II. State of the Art

Table I presents a comparison between deep packet inspection MD and Statistical Analysis-based AD methods

978-0-9883-0453-6/17 $31.00 © 2017 ITC
DOI 10.1109/ITC.29.147

about complexity, speed ,processing method, and limitations. Concerning the family of Misuse Detection, [6] proposes

TABLE I
MB INTRUSION DETECTION VERSUS SABID SYSTEMS.

	Deep Packet Inspection MD	Statistical Analysis Based AD
Processing method	Inspection of the whole packet content	It opens only packet headers (e.g. at the IP and TCP/UDP layers)
Complexity	High	Low
Speed	Slow	Fast
Limitations	It cannot detect new virus or encrypted flow	A training data set is involved

a host-rule-behavior-based detection method composed of a clustering engine that groups the objects of a suspicious program together into a cluster. The authors show that their results are more satisfying than the ones got by commercial antivirus software. [7] is a paper whose experimental results show the detection ability of the system to learn effective rules from repeated presentations of a tagged training set. [8] develops an automatic categorization system to automatically group phishing websites or malware samples by using a cluster ensemble. Among signature-based approaches: [9] compares the performance of the intrusion detection systems Suricata and Snort. [10] selects the possible signatures and uses only a subset of the necessary ones. [11] classifies packed and polymorphic malware through a fast application-level emulator.

Considering the systems that use Anomaly Detection (or also hybrid Statistical Analysis/Misuse Detection): [12] proposes a hybrid IDS combining packet header anomaly detection (PHAD) and network traffic anomaly detection (NETAD). [13] describes a two stage architecture to tackle intrusions. In the first stage a probabilistic classifier is used to detect potential anomalies in the traffic. In the second stage a HMM (Hybrid Markov Model) traffic model is used to narrow down the number of IP addresses carrying the attack. [14] introduces a hybrid intrusion detection system that combines k-Means and two classifiers for anomaly detection: K-nearest neighbor and Naive Bayes. [15] introduces a hybrid detection framework combining misuse detection, which uses a Random Forest classification algorithm, and anomaly detection, which exploits the weighted k-Means scheme.

[16] and [17] are aimed at detecting application-layer tunnels, which are the considered anomalies, throughout Statistical Fingerprints. [16] presents a statistical classification mechanism, called Tunnel Hunter, devoted to recognize a generic application protocol tunneled on top of HTTP or of SSH. [17] aims to detect DNS tunnels. Another important paper that uses similar techniques to the one used in this paper, is [18], where streaming content changes are detected only through traffic patterns built from the traffic volume achieved by routers. [19] introduces a scheme for intrusion detection operating in WEKA. [20] proposes to structure Machine-Learning-based intrusion detection systems into Artificial Intelligence based and Computational Intelligence based ones. The former refer to the methods from domains such as statistical modeling, whereas the latter include methodologies such as genetic algorithms, artificial neural network, fuzzy logic, and artificial immune systems. [21] extracts a long list of features from the dataset in [22] and compares the performance of different machine learning classifiers such as DTNB, JRIP, PART, Ridor for malware detection. [23] uses classifier J48, Random Forest and Random Tree in the same operating environment by using the same dataset and list of features presented in [21] and proposes to use a combination of classifiers to enhance the performance. [24] introduces a selection of features by using swarm intelligence algorithms, such as Artificial Bee Colony (ABC) or Particle Swarm Optimization (PSO), and evaluates the performance through the same dataset used in [22].

III. SDN VS NON-SDN APPROACH

Our previous work [25] describes the architecture of an Intrusion Detection System based on Statistical Fingerprint that is aimed at distinguishing malicious from normal traffic. The model of the system is based on the TCP/IP architecture, composed of a flow analyzer and a “filter”. The flow analyzer checks the IP and TCP/UDP headers of all the flows traversing the interface in order to gather the necessary features for each flow. The features used in [25] are reported in Table II. The ”filter“ takes the features as input and applies a machine learning technique to the purpose of detecting if the flow is affected by malware or not.

This paper focuses on the use of the SDN paradigm as network infrastructure for malware detection. What is the motivation to have an SDN-based IDS? Software Defined Networking (SDN) is revolutionizing the networking industry by enabling programmability, easier management and faster innovation. These benefits are made possible by its centralized control plane architecture, which allows the network to be programmed by the application and controlled from one central entity. The SDN architecture is composed of both switches/routers and a central controller (SDN controller). The peculiarity of this approach is that it decouples control and data planes in two separated entities:

- **Forwarding element**: it is a networking device (i.e. switch/router) but it is called “switch” in the SDN paradigm. The only task that is responsible for is the forwarding of packets inside the network. The switch processes packets according to rules stored in the flow tables filled by the controller.
- **Controller**: it is the brain of the entire network, it has the role of making decisions about all the flows that traverse the network, and, consequently, to fill the flow tables inside each SDN switch under its control.

The two entities communicate in order to exchange information and commands suited to manage the entire network. The protocol standard that makes possible the communication between the controller and the switches composing the network is OpenFlow [26]. Embedding a malware detector IDS within SDN is a clear step forward in the service provided by SDN and allows simplifying the IDS design being each action left to the SDN controller. Of course the implementation of malware detection on SDN presents some issues to investigate. The first problem to tackle is that the SDN standard does not allow to get all parameters in Table II. This leads to a reduction of the features involved for the malware detection. Consequently we have selected a limited number of features, both to be compliant to the SDN-OpenFlow standard and also to adapt to the features that most switches available in the market can really measure. The new set of features that can be collected using the SDN architecture are shown in Table III. As one can note their number is drastically reduced: starting from 14 in Table II only 7 features can be used to detect if a flow is affected by malware or not in the SDN environment.

TABLE II
NON SDN FEATURES FOR EACH FLOW AS STATISTICAL FINGERPRINT.

Features	Description
Num_Pack	Number of packets
Tot_Byte_Flow	Number of bytes
Flow_Duration	Duration of the flow in seconds
Byte_Rate	Byte rate
Packet_Rate	Packet rate
Delta_Mean	Average inter-arrival time of packets
Delta_Std	Standard deviation of inter-arrival time
LE	"Entropy" of the packet lengths[1]
DPL	Total number of subsets of packets having the same length divided by the total number of packets of the flow
First_Len	Length of the first packet
Max_Len	Length of the longest packet
Min_Len	Length of the shortest packet
Mean_Len	Average packet length
Std_Len	Standard deviation of the packet length

TABLE III
SDN FEATURES FOR EACH FLOW AS STATISTICAL FINGERPRINT.

Features	Description
Num_Pack	Number of packets
Tot_Byte_Flow	Number of bytes
Flow_Duration	Duration of the flow in seconds
Byte_Rate	Byte rate
Packet_Rate	Packet rate
First_Len	Length of the first packet
Mean_Len	Average packet length

As said, the feature limitation is due to the SDN protocol and architecture. Only the first packet of a flow, if and only if there are no rules to forward it, is received by the controller. For this reason we can extract the length of the first packet of a flow (First_Len) but we cannot compute the parameters Delta_Min and Std, LE, DPL, Max_Len, Min_Len and Std_Len. Referring to Table III: only the Number of packets, the Number of bytes, and the Duration of the flow can be directly measured by an SDN Switch and sent to the Controller through a suitable message. Byte and Packet rate, as well as Average packet length may be computed by the Controller on the basis of the received information.

IV. SYSTEM ARCHITECTURE

The architecture of the entire system is shown in Figure 1. The system is composed of an SDN switch responsible to route the packets coming from the external interface and directed to the LAN and vice-versa. Inside the architecture, thanks to the SDN paradigm, it is possible to implement the malware detector IDS needed to reveal the malicious traffic. The main component of the system is the Controller, which periodically collects traffic statistics, makes computations so to get the features in Table III and, based on the Malware Database, applies a configurable machine learning scheme that classifies the traffic as malware or normal traffic.

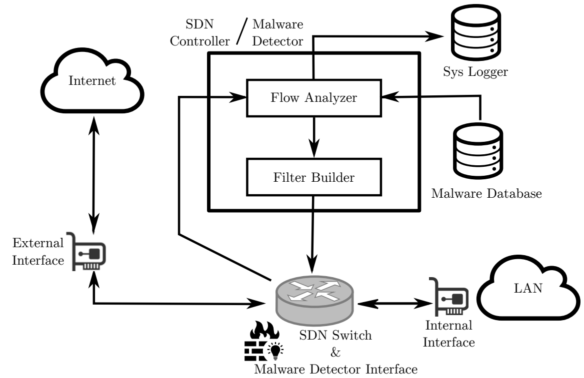

Fig. 1. Architecture of the proposed solution

The system works as described in the following: packets from the Internet traverse an SDN switch under the control of the SDN controller. If the switch does not have any rule about the arrived packet, it sends the packet to the controller which takes the information related to this packet and computes the rule needed to route it. After that the controller sends the rule back to the switch that will be able to forward/manage the corresponding flow. A flow is defined here by the vector $\{$Source IP Address, Destination IP Address, Source TCP/UDP Port, Destination TCP/UDP Port, Protocol$\}$ extracted from the IP, and TCP/UDP headers of the first packet. From now on the flow is continuously monitored by the switch using the given rule. The process is repeated for each "first packet" of any flow.

After a certain time period (called T_{stat}) the controller sends a feature request packet to the switch in order to collect all the features of the flows that have traversed the switch. Once the controller has received the feature reply that contains, as said, Number of packets, Number of bytes, and Duration of the flows in $[s]$, it processes this information to the purpose of extracting the other features: Byte rate, Packet rate, and Average packet length. The length of the first packet of the flow is already stored in the Controller. After that the Controller classifies each flow as malware affected or not.

[1]LE is calculated starting from the normalized occurrences of the packet lengths. Specifically, being L_i the number of times a packet has a length equal to i, LE is computed as $LE = -\sum_{i=0}^{1526} \frac{L_i}{N} \log_2(\frac{L_i}{N})$, where N is the total number of packets belonging to the flow.

The module of the Controller responsible of the classification of flows is called Flow Analyzer, as reported in Figure 1. This classification is made by a configurable machine learning technique. We have chosen the Support Vector Machine [27] algorithm in this paper. The module loads a previously trained model of the selected SVM algorithm, receives the statistics of the traffic traversing the network, and applies the prediction scheme in order to decide if a particular flow is affected by malware or not. Through this information the controller can make decisions on what to do about the flow. For example, it is possible to immediately stop the flow in order to prevent possible further infections or to mirror the traffic to a deep packet inspector to further analyze the flow.

V. Experimental Results

The architecture shown in Figure 1 is used as a reference for the experimental results. In order to tune and test the Flow Analyzer we have simulated the behavior of a network in which there is both malware affected traffic and regular not affected flows. To this purpose we have mixed traffic from traces surely containing only malware packets and from ones containing only normal traffic.

A. Malware Traffic

The malware traffic in this paper is composed of a mix of different malwares presented in Table IV:

TABLE IV
Used malware

Malware	Description
AlienspyRat	Remote Access Trojan
Cutwail	Botnet used for DDos
Kuluoz	Phishing botnet
Purplehaze	Advertising botnet
Ramnit	Trojan
Tbot	Trojan
ZeroAccess	Trojan
Zeus	Trojan
Asprox	Spam botnet
Madness	DDos botnet
Neris	HTTP botnet

All used traces can be found online at [28]–[31].

B. Normal Traffic

Concerning used normal traffic, we have captured all traffic in our laboratory. In order to be sure that no malware is involved, we have configured our laboratory switch so to forward all the traffic on a specific physical port; we have connected this port to a pc configured as a virtual switch that forwards the traffic coming from the ingress line card to an egress line card connected with the router; and we have mirrored the traffic to the local port connected to a sniffer.

C. Preliminary Performance Analysis

The first step of performance analysis is the comprehension of the more relevant features that strongly impact the results of the machine learning technique. The aim is to check the relevance of the features available in the SDN environment and of the complete set in Table II. Different feature ranking and selection techniques have been proposed in the machine learning literature. All these approaches have the purpose of discarding redundant features. In this framework, we consider the Information Gain (IG).

The evaluation using the IG uses the entropy (1) of a variable Y

$$H(Y) = -\sum_{y \in Y} p(y) \log_2(p(y)) \tag{1}$$

where $p(y)$ is the marginal probability density function for the random variable Y. If the observed values of Y in the training data set S are partitioned according to the values of a second feature X and the entropy of Y with respect to the partitions induced by X is below the entropy of Y before partitioning, then there is a relationship between the features Y and X. The entropy of Y after observing X is:

$$H(Y/X) = -\sum_{x \in X} p(x) \sum_{y \in Y} p(y/x) \log_2(p(yx)) \tag{2}$$

where $p(y/x)$ is the conditional probability of y given x. Considering the entropy as a criterion of impurity in a training set S, we can define a measure $H(Y/X)$ reflecting the additional information about Y provided by X. $H(Y/X)$ represents the decrease of the entropy of Y. The difference between $H(Y)$ and $H(X/Y)$ is known as IG:

$$IG = H(Y) - H(Y/X) = H(X) - H(X/Y) \tag{3}$$

The information gained about Y after observing X is equal to the information gained about X after observing Y.

Using this technique we can investigate the information brought by each feature and we can rank the features in order to understand their importance. After an investigation using a 10 fold cross validation method, the features in Table II are ranked as shown in Table V. The "Average Merit" is the measure of the importance averaged over the folds of the cross validation.

TABLE V
Ranked Features

Average Merit	Rank	Attribute
0.922	1	**First_Len**
0.675	2	Max_Len
0.661	3	Min_Len
0.648	4	**Tot_Byte_Flow**
0.631	5	Delta_Std
0.621	6	Delta_Mean
0.583	7	**Num_Pack**
0.569	8	**Packet_Rate**
0.567	9	DPL
0.563	10	**Flow_Duration**
0.538	11	Byte_Std
0.533	12	**Mean_Len**
0.505	13	**Byte_Rate**
0.249	14	LE

The features that the SDN architecture can gather are evidenced in bold. Their rank is: $1, 4, 7, 8, 10, 12$, and 13. Their Average Merit is relatively high and so it is reasonable to proceed with the investigation, checking which is the practical results of malware detection by using the limited set of features in Table III with respect to the full set in Table II.

D. Operative Analysis

The Controller extracts the Statistical Fingerprint in Table III from the trace of all the flows and forwards it as input to the machine learning scheme. The considered classification technique is the Radial Basis Functions (RBF) SVM [32] that needs two phases in order to work properly:

1) Training Phase: We define $\boldsymbol{x}_f$ as the feature vector of the f-th flow (its statistical fingerprint). $\boldsymbol{y}$ is the vector containing the two possible classes of assignation ("malware" or "normal"). $(\boldsymbol{x}_f, y_f)$ $\forall f \in [1, F]$ is the tuple containing all the information regarding each single flow. F is the total number of flows in the training set. SVM is trained building an hyperplane in order to separate the two considered classes.

SVM performs the classification by using the *kernel function*. There are many *kernel functions* but one of the most used is the Radial Basis (RBF), chosen for this paper. RBF uses two different parameters in the training phase: C (complexity parameter) and γ (kernel parameter), set to 20 and 2 respectively, in this paper, after experimental tests.

2) Test Phase: Let N_f be the number of features of each flow and being F the total number of flows, the matrix $\boldsymbol{\omega} \in F \times N_f$ contains all the feature vectors:

$$\boldsymbol{\omega} = \begin{bmatrix} \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_f & \cdots & \boldsymbol{x}_F \end{bmatrix}^T \tag{4}$$

The single flow f is associated to the predicted class $y_{f^*} \in \boldsymbol{y}$ evaluating its position with respect to the previously built hyperplane by using the matrix $\boldsymbol{\omega}$.

E. Classifier Performance

The performance of the classifier has been evaluated by comparing the results of the classification with the ground truth on the basis of the following metrics:

- True Positive (TP) - A flow is assigned to the right class.
- False Positive (FP) - A Flow is assigned to the wrong class.

We performed tests by training the SVM with different percentage (33%, and 50%) of training samples extracted from the dataset. We have tested our scheme with malware type belonging to the same dataset used for training and with malware whose type does not belong to the training data in order to prove the effectiveness and the feasibility of the malware detector. We have compared the classification algorithm with all the features reported in Table II and reducing the feature as reported in Table III.

Table VI shows the results obtained through a dataset containing the following malwares: AlienspyRat, Cutwail, Kuluoz, Purplehaze, Ramnit, Tbot, ZeroAccess, and Zeus; and mixing this traffic with normal flows. The SVM is trained with 33% and 50% of the flows in the dataset randomly chosen by using both all the features reported in Table II and only the SDN features reported in Table III. The test phase is performed by using the flows in the dataset not used for the training. In all considered cases the RBF SVM classifier recognizes malware traffic with a True Positive Rate above 98%. The limited set of features in Table III provides approximately the same performace of the full set of features concerning this metric. This is not true for normal traffic that is correctly classified with a rate of about 87% by using the limited set of features instead of 98% provided by the full set. The opposite is true for the FP rate that, in the SDN set of features case, is practically the same of the full set case for normal traffic but is meaningfully different for malware.

TABLE VI
SDN VS FULL SET OF FEATURES

Experiment	TP Rate	FP Rate	Class
33% Training, full set of features	0.990 0.985	0.015 0.010	malware normal
50% Training, full set of features	0.989 0.989	0.011 0.011	malware normal
33% Training, SDN set of features	0.984 0.862	0.138 0.016	malware normal
50% Training, SDN set of features	0.984 0.876	0.124 0.016	malware normal

Finally we have tested our system with malware type samples different from the ones used for training. The training set is the same as in Table VI: AlienspyRat, Cutwail, Kuluoz, Purplehaze, Ramnit, Tbot, ZeroAccess, and Zeus. The test set is composed by flows belonging to Asprox, Madness, and Neris, Table VII shows the results. The performance are really satisfying. The malware True Positive rate is over 80% both in the case of full and SDN features. The two cases are practically overlapped. The FP rate for malware is 2.7% for the full set of features and 5.4% for the SDN features. The TP rate for normal traffic is also quite satisfying: 94.6% for the limited set of vs 97.3% provided by the full set. The weakness concerns the FP rate for normal traffic: 18.5% both for the SDN and the full set of features.

VI. Conclusions

The paper combines the advantage of a Statistical Fingerprint IDS with the potentiality of a Software Defined Networking (SDN) architecture. In SDN the brain of the system is decoupled from the nodes that compose the network

TABLE VII
RESULTS USING DIFFERENT TEST DATA

Experiment	TP Rate	FP Rate	Class
33% Training, full set of features	0,818 0,973	0,027 0,182	malware normal
50% Training, full set of features	0,818 0,973	0,027 0,182	malware normal
33% Training, SDN set of features	0,815 0,946	0,054 0,185	malware normal
50% Training, SDN set of features	0,815 0,946	0,054 0,185	malware normal

and is located in a centralized and well separated entity (the controller). This entity has the control of the entire network and can act at higher level coordinating all the network nodes in order to avoid possible malware intrusions. This approach can act by using hardware already in the market. The only requirement is to use the OpenFlow protocol, which is already standardized and employed in the network environment. The proposed system acts as follows: network nodes, also called Switches, are responsible for the collection of the features needed to infer information from the flows traversing the network. The Controller contains a configurable machine learning module that, starting from the features extracted by switches, completes the number of needed features through computations and decides if a flow is malware affected or not. The scheme presented in this paper can lead to an innovative solution aimed at stopping the proliferation of malware inside the network. Using SDN implies a reduction of the number of features that can be practically used to detect malware. The shown performances evaluation shows that the performance by using this limited set of features is still satisfying. In particular the detection of not trained malware is above 80% while the detection of normal traffic is about 95%. False positive rate is quite low for malware (5.4%) but needs to be improved for normal traffic (18.5%).

REFERENCES

[1] Y. Ye, T. Li, Q. Jiang, and Y. Wang, "Cimds: adapting postprocessing techniques of associative classification for malware detection," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 40, no. 3, pp. 298–307, 2010.

[2] V. G. Cerf, "Defense against the dark arts," *Internet Computing, IEEE*, vol. 16, no. 1, pp. 96–96, 2012.

[3] F. Sabahi and A. Movaghar, "Intrusion detection: A survey," in *Systems and Networks Communications, 2008. ICSNC'08. 3rd International Conference on*. IEEE, 2008, pp. 23–26.

[4] W. Stallings, "Software-defined networks and openflow," *The internet protocol Journal*, vol. 16, no. 1, pp. 2–14, 2013.

[5] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1617–1634, Third 2014.

[6] Z. Shan and X. Wang, "Growing grapes in your computer to defend against malware," *Information Forensics and Security, IEEE Transactions on*, vol. 9, no. 2, pp. 196–207, 2014.

[7] J. J. Blount, D. R. Tauritz, and S. A. Mulder, "Adaptive rule-based malware detection employing learning classifier systems: a proof of concept," in *Computer Software and Applications Conference Workshops (COMPSACW), 2011 IEEE 35th Annual*. IEEE, 2011, pp. 110–115.

[8] W. Zhuang, Y. Ye, Y. Chen, and T. Li, "Ensemble clustering for internet security applications," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 42, no. 6, pp. 1784–1796, 2012.

[9] A. Alhomoud, R. Munir, J. P. Disso, I. Awan, and A. Al-Dhelaan, "Performance evaluation study of intrusion detection systems," *Procedia Computer Science*, vol. 5, pp. 173 – 180, 2011, the 2nd International Conference on Ambient Systems, Networks and Technologies (ANT-2011) / The 8th International Conference on Mobile Web Information Systems (MobiWIS 2011). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050911003498

[10] S. K. Cha, I. Moraru, J. Jang, J. Truelove, D. Brumley, and D. G. Andersen, "Splitscreen: Enabling efficient, distributed malware detection," *Communications and Networks, Journal of*, vol. 13, no. 2, pp. 187–200, 2011.

[11] S. Cesare, Y. Xiang, and W. Zhou, "Malwise - an effective and efficient classification system for packed and polymorphic malware," *IEEE Transactions on Computers*, vol. 62, no. 6, pp. 1193–1206, June 2013.

[12] M. A. Aydın, A. H. Zaim, and K. G. Ceylan, "A hybrid intrusion detection system design for computer network security," *Computers & Electrical Engineering*, vol. 35, no. 3, pp. 517–526, 2009.

[13] R. R. Karthick, V. P. Hattiwale, and B. Ravindran, "Adaptive network intrusion detection system using a hybrid approach," in *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*. IEEE, 2012, pp. 1–7.

[14] H. Om and A. Kundu, "A hybrid system for reducing the false alarm rate of anomaly intrusion detection system," in *Recent Advances in Information Technology (RAIT), 2012 1st International Conference on*. IEEE, 2012, pp. 131–136.

[15] R. M. Elbasiony, E. A. Sallam, T. E. Eltobely, and M. M. Fahmy, "A hybrid network intrusion detection framework based on random forests and weighted k-means," *Ain Shams Engineering Journal*, vol. 4, no. 4, pp. 753–762, 2013.

[16] M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli, "Tunnel hunter: Detecting application-layer tunnels with statistical fingerprinting," *Computer Networks*, vol. 53, no. 1, pp. 81–97, 2009.

[17] M. Aiello, M. Mongelli, and G. Papaleo, "Dns tunneling detection through statistical fingerprints of protocol messages and machine learning," *International Journal of Communication Systems*, vol. 28, no. 14, pp. 1987–2002, 2015.

[18] H. Nakayama, A. Jamalipour, and N. Kato, "Network-based traitor-tracing technique using traffic pattern," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 2, pp. 300–313, June 2010.

[19] M. N. Mohammad, N. Sulaiman, and O. A. Muhsin, "A novel intrusion detection system by using intelligent data mining in weka environment," *Procedia Computer Science*, vol. 3, pp. 1237–1242, 2011.

[20] M. Zamani and M. Movahedi, "Machine learning techniques for intrusion detection," *arXiv preprint arXiv:1312.2177*, 2013.

[21] G. V. Nadiammai and M. Hemalatha, "Perspective analysis of machine learning algorithms for detecting network intrusions," in *Computing Communication Networking Technologies (ICCCNT), 2012 Third International Conference on*, July 2012, pp. 1–7.

[22] "Kdd cup 1999 data," http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 2016.

[23] S. Saravanan, S. Vijay Bhanu, and R. Chandrasekaran, "Study on classification algorithms for network intrusion systems," *Journal of Communication and Computer*, vol. 9, no. 11, pp. 1242–1246, 2012.

[24] A. C. Enache and V. V. Patriciu, "Intrusions detection based on support vector machine optimized with swarm intelligence," in *Applied Computational Intelligence and Informatics (SACI), 2014 IEEE 9th International Symposium on*, May 2014, pp. 153–158.

[25] L. Boero, M. Cello, M. Marchese, E. Mariconti, T. Naqash, and S. Zappatore, "Statistical fingerprint-based intrusion detection system (SF-IDS)," *International Journal of Communication Systems*, vol. 30, no. 10, 2017.

[26] "Openflow switch specification - version 1.4.0," https://goo.gl/GEXWQc, Open Networking Foundation, October 14, 2013, last view at October, 2016.

[27] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[28] Available on line http://www.malware-traffic-analysis.net, last visit March 2017.

[29] Available on line http://contagiodump.blogspot.it, last visit March 2017.

[30] Available on line http://www.pcapanalysis.com, last visit March 2017.

[31] Available on line at: www.mediafire.com/?a49l965nlayad, last visit March 2017.

[32] R. K. Dash, "Selection of the best classifier from different datasets using weka," in *International Journal of Engineering Research and Technology*, vol. 2, no. 3 (March-2013). ESRSA Publications, 2013.