ADENOIDS: softwAre DEfined NetwOrking-based Intrusion Detection System

Luca Boero, Mario Marchese, and Sandro Zappatore

University of Genoa, Genoa, Italy
luca.boero@edu.unige.it, mario.marchese@unige.it, sandro.zappatore@unige.it

Abstract

Intrusion Detection Systems (IDS) are systems 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 introduces ADENOIDS a Statistical Fingerprint based Intrusion Detection System that is build on top of an SDN architecture.

1 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 [27]. To prevent these type of attack it is necessary to accurately detect malware and other type of intrusions [8]. In general it is possible to use Intrusion Detection Systems (IDS) in order to tackle malicious intrusions. An IDS is a piece of hardware/software designed to alert when someone or something is trying or has tried to compromise systems. [22] describes two of the major classifications for what concern the processing method in anomaly detection: Misuse and Anomaly Detection. The first one tries to fix the abnormal behavior and considers the rest as normal. On the contrary 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. 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 a 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) [26] [19] 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 called ADENOIDS (Software Defined Networking-based Intrusion Detection System). ADENOIDS uses the typical flow definition at TCP/IP (Transfer Control Protocol/ Internet Protocol) layers and is aimed at deciding whether an IP 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 IP flow, which represent the Statistical Fingerprint of the flow and on machine learning-based classifiers devoted to distinguish normal from malicious traffic.

The paper is organized as follows: Section 2 contains the state of the art concerning deep packet inspection MD and Statistical Analysis-based Anomaly Detection, Section 3 describes the differences between an SDN and non-SDN approach, Section 4 describes in detail the proposed architecture, Section 5 shows the obtained results and Section 6 contains the conclusions.

2 State of the Art

Table 1 presents a comparison about processing method, complexity, speed, and limitations between deep packet inspection MD and Statistical Analysis-based AD methods.

	Deep Packet Inspection MD	**Statistical Analysis Based AD**
Processing method	It examines the whole packet content, analysing data at application layer looking for signatures/rules	It opens packet headers (e.g. at the IP and TCP/UDP layers) to identify flows and examines traffic statistically
Complexity	High	Low
Speed	Slow	Fast
Limitations	It cannot detect new virus or encrypted flow	A training data set is involved

Table 1: MB Intrusion Detection versus SABID systems.

Concerning the family of Misuse Detection, [25] proposes 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. [6] 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. [29] develops an automatic categorization system to automatically group phishing websites or malware samples by using a cluster ensemble. [20] and [23] present algorithms based on the analysis of operational code (operational code are part of machine language dedicated to specify the operation to be performed). [20] is aimed at individuating a subset of opcodes suitable for malware detection through SVM (Support Vector Machine). [23] proposes a method that uses single-class learning to detect unknown malware families. Among signature-based approaches:

[4] 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. [9] 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): [5] proposes a hybrid IDS combining packet header anomaly detection (PHAD) and network traffic anomaly detection (NETAD). [15] 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. [21] introduces a hybrid intrusion detection system that combines k-Means and two classifiers: K-nearest neighbor and Naive Bayes for anomaly detection. [13] 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.

[12] and [3] are aimed at detecting application-layer tunnels, which are the considered anomalies, throughout Statistical Fingerprints. [12] presents a statistical classification mechanism called Tunnel Hunter devoted to recognize a generic application protocol tunneled on top of HTTP or of SSH. [3] aims at detecting DNS tunnels. Another important paper that uses techniques similar concerning 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. [16] introduces a scheme for intrusion detection operating in WEKA. [28] 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. [17] extracts a long list of features from the used dataset [1] and compares, different machine learning classifiers such as DTNB, JRIP, PART, Ridor. [24] uses classifier J48, Random Forest and Random Tree in the same operating environment by using the same dataset and list of features presented in [17] and proposes to use a combination of classifiers to enhance the performance. [14] proposes 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 [1].

3 SDN vs Non-SDN approach

Our previous work [7] describes in detail the architecture of an Intrusion Detection System based on Statistical Fingerprint that aims at distinguishing malicious traffic from normal one. The model of the system is based on the classical TCP/IP architecture, composed of a flow analyzer and a "filter". The former analyzes all the flows traversing the interface, IP and TCP/UDP headers of packets are checked in order to gather the necessary features for each flow. The features that are used in [7] are reported in Table 2. The system has to compute these parameters in order to characterize the flows traversing the network. For each flow the second part of the system takes as input all the parameters showed in Table 2 and then applies a machine learning technique to the purpose of detecting if the flow is affected by malware or not.

The work presented in this paper is focusing 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 well separated entity:

- **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 so-called flow tables that are 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 [2]. Embedding a malware detector IDS within SDN would be a clear step forward in the service provided by SDN and would allow to simplify the IDS design being each action left to the SDN controller. Of course the malware detection implementation 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 2, which leads to a reduction of the features involved for the malware detection. For this reason we have selected a limited number of features, in this way we can be compliant to the SDN-OpenFlow standard and also with the features that most switches already available in the market can really measure.

The new set of features that can be collected using the Software Defined Network architecture are reported in Table 3. As one can note their number is drastically reduced, starting from 14, using the SDN paradigm, only 7 features can be used to detect if a flow is affected by malware or not.

Features	**Description**
Num_Pack	Number of packets
Tot_Byte_Flux	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 2: Non SDN features for each flow as Statistical Fingerprint.

Features	Description
Num_Pack	Number of packets
Tot_Byte_Flux	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

Table 3: SDN features for each flow as Statistical Fingerprint.

As mentioned before, the features limitations 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 and we cannot compute the Delta parameter as well as LE, DPL, Max, Min and Std. Referring to Table 3: 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 the Average packet length may be computed by the Controller on the basis of the received information.

4 ADENOIDS Architecture

The architecture of the entire system is shown in Figure 1. The system is composed of an SDN switch that is responsible for the routing of packets coming from the external interface 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 stop 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 3 and, based on the Malware Database, applies a configurable machine learning scheme that classifies the traffic as malware or normal traffic.

In more detail ADENOIDS works as described in the following: packets from the Internet traverse an SDN switch under the control of the SDN controller. If the switch doesn't 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, TCP/UDP header 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 elaborates this information to the purpose of extracting the other features suitable for the analysis: Byte rate, Packet rate, and Average packet length.

[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.

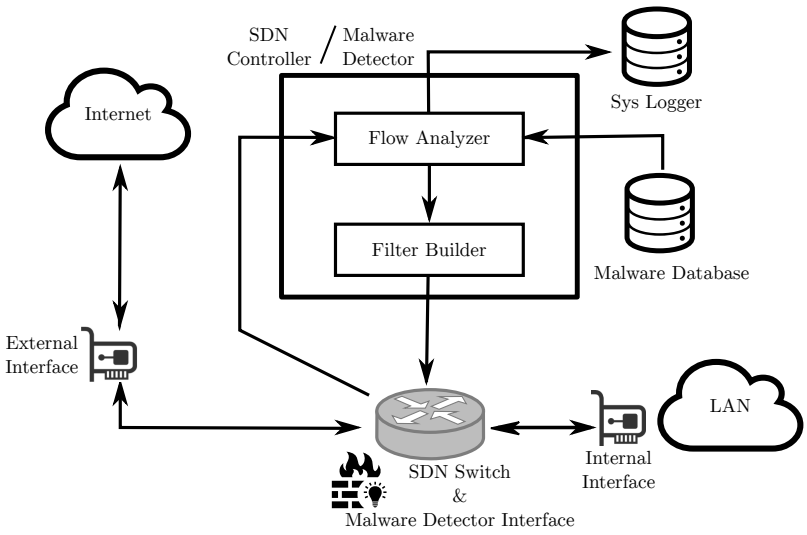

Figure 1: ADENOIDS Architecture

The length of the first packet of the flow was already stored in the Controller. After that the Controller classifies each flow as malware affected or not.

Malware	**Flows**	**Packets**
Cutwail	2347	35674
Purple Haze	7349	324709
Ramnit	25141	155973
Tbot	223	13048
Zeus	202	7443
ZeroAccess	350	2535
AlienspyRAT	1214	9010
Kuluoz	16894	179607

Table 4: Used malware.

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. It takes as input the model of the selected machine learning scheme, previously trained with the use of the Malware Database composed of the 50% of the packets shown in Table 4 and applies the model on the extracted features. The algorithm output is the distinction if the traffic is malware affected or not.

5 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 do this we have used as malware the 50% of the Malware Database not used for the training phase and we

have mixed these flows with captured regular traffic which we know it doesn't contain any malware. The number of captured packets and flows is reported in Table 5. From the traces

Normal Traffic	Flows	Packets
Normal Traffic 1	4969	833368
Normal Traffic 2	12552	3533925
Normal Traffic 3	23351	4428188

Table 5: Captured normal traffic.

reported above the Controller extracts the Statistical Fingerprint (see Table 3) of all the flows and forwards it as input to a suitable machine learning scheme whose selection is the aim of this performance evaluation. The considered classification techniques are [11]: Linear SVM - the frontier between regions is a linear function; Quadratic SVM - the frontier between regions is a quadratic function; Cubic SVM - the frontier between regions is a cubic function; Radial Basis Functions (RBF) SVM; K-Nearest Neighbors - K-NN with $K = 1$, and $K = 3$; JRIP; Random Forest; DTNB; PART; Ridor; SMO; J48; Random Tree; and RBF Network. The performance of the each classifier reported above has been evaluated by comparing the results of the classification with the ground truth. Under this perspective, 4 cases, can occur:

- True Negative (TN) - A flow is normal traffic, i.e., it is not malware affected and it is correctly classified as normal traffic.
- False Positive (FP) - A flow is normal traffic, i.e., it is not malware affected but it is wrongly classified as malware. This case is also called False Alarm.
- True Positive (TP) - A flow is malware affected and it is correctly classified as malware.
- False Negative (FN) - A flow is malware affected but it is wrongly classified as normal traffic. This case is also called Missed Detection.

The results for each single classifier are reported in Table 6 that includes also the boundaries of the 95% confidence interval for the measure of the Accuracy, LINT for the lower bound and HINT for the upper bound. The evaluation parameter Accuracy is computed by summing the number of flows marked by the algorithm as True Negative and number of flows marked as True Positive and by dividing the obtained quantity by the total number of flows.

The results show that the Flow Analyzer can individuate with a satisfactory precision the tested malware. The tree based classifiers achieve better performances with respect to Support Vector Machine ones: Random Forest and J48 show the best results in terms of Accuracy. All the algorithms used for the tests show good results in terms of True Positives and False Negatives, i.e. the flow is affected by malware and the applied machine learning technique recognizes it as malware. Support Vector Machine-based schemes as well as NaiveBayes and RBF Network and SMO fails in the recognition of normal traffic which is often wrongly classified as a malware.

6 Conclusions

The paper tries to combine the advantage of a Statistical Fingerprint IDS with the potentiality of a Software Defined Networking architecture. In this approach the brain of the system is decoupled from the nodes that compose the network and is located in a centralized and well separated entity. This entity has the control of the entire network and can act at higher

Classifier	Accuracy	TP	FN	TN	FP	LINT	HINT
1NN	96.5665	96.7	3.3	96.5	3.5	96.4126	96.7204
3NN	96.9311	97.6	2.4	96.2	3.8	96.7853	97.0769
Cubic SVM	50.6594	100	0	1.9	98.1	50.2368	51.082
DTNB	97.1059	98.4	1.6	95.9	4.1	96.9642	97.2476
J48	97.9280	98.7	1.3	97.2	2.8	97.8076	98.0484
JRIP	97.9038	99	1	96.8	3.2	97.7827	98.0249
Linear SVM	86.6679	94.1	5.9	79.3	20.7	86.3806	86.9552
NaiveBayes	72.0506	99.6	0.4	44.8	55.2	71.6713	72.4299
PART	97.8518	99.4	0.6	96.3	3.7	97.7292	97.9744
Quadratic SVM	50.3729	97.8	2.2	3.5	96.5	49.9503	50.7955
Random Forest	97.9727	98.7	1.3	97.3	2.7	97.8536	98.0918
Random Tree	97.4816	97.7	2.3	97.3	2.7	97.3492	97.614
RBF Network	69.2477	92.9	7.1	45.9	54.1	68.8576	69.6378
RBF SVM	79.8010	91.7	8.3	68	32	79.4616	80.1404
Ridor	97.1915	99.5	0.5	94.9	5.1	97.0518	97.3312
SMO	86.6679	94.1	5.9	79.3	20.7	86.3806	86.9552

Table 6: Evaluation parameters and 95% confidence interval.

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 is called ADENOIDS (softwAre DEfined NetwOrk Intrusion Detection System). It is composed essentially of two different entities: the network node also called Switch, responsible for the collection of the features needed to infer information from the flows traversing the network, and the Controller that acts as the brain of the network and contains a configurable machine learning module that, starting from the features extracted by the switch, 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.

References

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

[2] Openflow switch specification - version 1.4.0. https://goo.gl/GEXWQc, October 14, 2013. Last view at October, 2016.

[3] M Aiello, M Mongelli, and G Papaleo. Dns tunneling detection through statistical fingerprints of protocol messages and machine learning. *International Journal of Communication Systems*, 28(14):1987–2002, 2015.

[4] Adeeb Alhomoud, Rashid Munir, Jules Pagna Disso, Irfan Awan, and A. Al-Dhelaan. Performance evaluation study of intrusion detection systems. *Procedia Computer Science*, 5:173 – 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).

[5] M Ali Aydın, A Halim Zaim, and K Gökhan Ceylan. A hybrid intrusion detection system design for computer network security. *Computers & Electrical Engineering*, 35(3):517–526, 2009.

[6] Jonathan J Blount, Daniel R Tauritz, and Samuel A Mulder. Adaptive rule-based malware detection employing learning classifier systems: a proof of concept. In *Computer Software and Appli-*

cations Conference Workshops (COMPSACW), 2011 IEEE 35th Annual, pages 110–115. IEEE, 2011.

[7] Luca Boero, Marco Cello, Mario Marchese, Enrico Mariconti, Talha Naqash, and Sandro Zappatore. Statistical Fingerprint - Based Intrusion Detection System (SF-IDS). *International Journal of Communication Systems*, 2016. Accepted for Publication.

[8] Vinton G Cerf. Defense against the dark arts. *Internet Computing, IEEE*, 16(1):96–96, 2012.

[9] S. Cesare, Y. Xiang, and W. Zhou. Malwise - an effective and efficient classification system for packed and polymorphic malware. *IEEE Transactions on Computers*, 62(6):1193–1206, June 2013.

[10] Sang Kil Cha, Iulian Moraru, Jiyong Jang, John Truelove, David Brumley, and David G Andersen. Splitscreen: Enabling efficient, distributed malware detection. *Communications and Networks, Journal of*, 13(2):187–200, 2011.

[11] Ranjita Kumari Dash. Selection of the best classifier from different datasets using weka. In *International Journal of Engineering Research and Technology*, volume 2. ESRSA Publications, 2013.

[12] Maurizio Dusi, Manuel Crotti, Francesco Gringoli, and Luca Salgarelli. Tunnel hunter: Detecting application-layer tunnels with statistical fingerprinting. *Computer Networks*, 53(1):81–97, 2009.

[13] Reda M Elbasiony, Elsayed A Sallam, Tarek E Eltobely, and Mahmoud M Fahmy. A hybrid network intrusion detection framework based on random forests and weighted k-means. *Ain Shams Engineering Journal*, 4(4):753–762, 2013.

[14] 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*, pages 153–158, May 2014.

[15] R Rangadurai Karthick, Vipul P Hattiwale, and Balaraman Ravindran. Adaptive network intrusion detection system using a hybrid approach. In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, pages 1–7. IEEE, 2012.

[16] Muamer N Mohammad, Norrozila Sulaiman, and Osama Abdulkarim Muhsin. A novel intrusion detection system by using intelligent data mining in weka environment. *Procedia Computer Science*, 3:1237–1242, 2011.

[17] 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*, pages 1–7, July 2012.

[18] H. Nakayama, A. Jamalipour, and N. Kato. Network-based traitor-tracing technique using traffic pattern. *IEEE Transactions on Information Forensics and Security*, 5(2):300–313, June 2010.

[19] 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*, 16(3):1617–1634, Third 2014.

[20] Philip O'Kane, Sakir Sezer, Keiran McLaughlin, and Eul Gyu Im. Svm training phase reduction using dataset feature filtering for malware detection. *Information Forensics and Security, IEEE Transactions on*, 8(3):500–509, 2013.

[21] Hari Om and Aritra 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*, pages 131–136. IEEE, 2012.

[22] Farzad Sabahi and Ali Movaghar. Intrusion detection: A survey. In *Systems and Networks Communications, 2008. ICSNC'08. 3rd International Conference on*, pages 23–26. IEEE, 2008.

[23] Igor Santos, Felix Brezo, Borja Sanz, Carlos Laorden, and Pablo G Bringas. Using opcode sequences in single-class learning to detect unknown malware. *Information Security, IET*, 5(4):220–227, 2011.

[24] Subramanian Saravanan, Srinivasan Vijay Bhanu, and Ramasamy Chandrasekaran. Study on classification algorithms for network intrusion systems. *Journal of Communication and Computer*,

9(11):1242–1246, 2012.

[25] Zhiyong Shan and Xin Wang. Growing grapes in your computer to defend against malware. *Information Forensics and Security, IEEE Transactions on*, 9(2):196–207, 2014.

[26] William Stallings. Software-defined networks and openflow. *The internet protocol Journal*, 16(1):2–14, 2013.

[27] Yanfang Ye, Tao Li, Qingshan Jiang, and Youyu Wang. Cimds: adapting postprocessing techniques of associative classification for malware detection. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(3):298–307, 2010.

[28] Mahdi Zamani and Mahnush Movahedi. Machine learning techniques for intrusion detection. *arXiv preprint arXiv:1312.2177*, 2013.

[29] Weiwei Zhuang, Yanfang Ye, Yong Chen, and Tao Li. Ensemble clustering for internet security applications. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42(6):1784–1796, 2012.