A Botnet Detection Method on SDN using Deep Learning

Shogo Maeda
Science and Engineering
Hosei University
Tokyo, Japan

Atsushi Kanai
Science and Engineering
Hosei University
Tokyo, Japan

Shigeaki Tanimoto
Social Systems Science
Chiba Institute of Technology
Chiba, Japan

Takashi Hatashima
NTT Secure Platform Laboratories
Nippon Telegraph and Telephone Corporation
Tokyo, Japan

Kazuhiko Ohkubo
NTT Secure Platform Laboratories
Nippon Telegraph and Telephone Corporation
Tokyo, Japan

***Abstract*—The increased amount of user damage caused by malware is becoming a serious problem. Network software has also been advancing, and potentially it can be utilized to counter malware. In this work, we analyze the traffic data of malware collected from an existing network and classify normal traffic and malicious traffic by using deep learning. This makes it possible to combat unknown malware that cannot be dealt with using rule-based anti-virus software. By manipulating the network through programing, we can flexibly deal with malware. We propose running deep learning on a software-defined network to prevent host detection after infection and secondary damage detection of an infected host. In the proposed method, we add a connection block to the external network and perform network isolation to prevent internal infection.**

***Keywords*—deep learning, software-defined network, malware detection, botnet, neural network**

I. Introduction

The increased amount of user damage caused by malware is becoming a serious problem. Of particular concern are the botnets, which consist of three components: an attacker called a bot master, an infected machine called a bot, and command-and-control(C2C) server. The bot master sends instructions to the C2C server and the bot performs malicious actions by executing commands received from the C2C server. Examples of malicious behaviors include DDoS attacks, SPAM mail and data theft and concealment. The bot also communicates with the C2C server, updates the malware code, and propagates the commands to the bot. The bot regularly communicates with the C2C server and reports on the status. Botnets generally have three phases in a lifecycle: infection, C2C communication, and the attack phase [17]. They can centralize and decentralize C2C channels(HTTP P2P IRC etc.) according to communication protocols. IRC and HTTP are mainly used in the centralized type, and P2P communication is used in the distributed type.

For the centralized type, it is IRC and HTTPHTTPS are used. Bot masters have extensively used IRC-based C2C and HTTP-based C2C to introduce botnets from both spread and management aspects. This type of botnet centrally manages the C2C network architecture, and all bots connect to one or more C2C servers. The main weakness of a centralized system is that it is vulnerable to a single point of failure. Due to the enormous connection between the bot and the C2C server, it is easy to identify and disable centralized botnets. C2C servers are detected and disabled, IRC-based or HTTP-based botnets will be stopped.

In order to prevent a single point of failure, the distributed type deploys a botnet architecture with a peer-to-peer(P2P) communication protocol. Distributed botnets can issue commands to other peers or obtain useful information using either bots or P2P nodes. Any bot or P2P node may start malicious activity by communicating with the bot master and other nodes, even though some P2P botnets may be stopped.

Essentially, the evolving nature of the botnets means they cannot be dealt with by existing detection technology, and improvements are required every day simply to keep up with them.

Botnet detection techniques can be categorized as host-based and network-based. With the host-based technique, it identifies abnormal computer usage (e.g. increased CPU usage and memory usage) is identified. This technique is not affected by encrypted communication channels. However, the drawback is that the resource usage of all end hosts needs to be monitored which can be quite costly in terms of time. In contrast, with the network-based technique, the operation of the network connection is checked and the network traffic during the bot lifecycle is identified. In our opinion, it is easier to manage network-based detection because it is sufficient to simply see the status of the network to be monitored.

In terms of network-based botnet detection, there are signature-based methods and anomaly-based ones. The signature-based methods perform analyzes according to rules set by applying Deep Packet Inspection(DPI) on the contents of TCP/UDP packets. This method has a low false positive rate and is useful for detecting known botnets. However, it can only detect only known threats: i.e., it cannot deal with new varieties and subspecies. Therefore, frequent updat-

ing of signatures is necessary. Another difficulty with using signature-based methods is that signatures can be hidden by encryption and obfuscation of the C2C channel, Fast-Flex, DGA, etc. [14]. With anomaly-based methods, abnormalities such as packet payload and bot group behavior are detected. These methods are performed using various algorithms such as machine learning techniques and graph analysis and are extremely useful for extracting unexpected network patterns [8]. Anomaly-based methods are generally the preferred way to detect unknown botnets. However, these methods have a high false positive rate, which leads to the problem of increased detection errors [14].

In this work, we aim not only to detect by deep learning but also to separate infected machines using software-defined network(SDN). SDN is based on the idea of having a programmable network [10]. It is an architecture that eliminates the complexity of each conventional network device by separating the network control unit and the data transfer unit. The network control of SDN is performed by a controller, and the flow entry of the flow table of the SDN switch (such as OpenFlow switch) is managed using Southbound API. This process is handled by the flow entry in the flow table in response to the packet. When the switch receives a new flow from the host, the SDN processing flow forwards the traffic to the controller and establishes the communication path. The controller decides how to forward it to the destination of that traffic and sets the flow rules at the source and the destination. In the case of the same flow, it is processed according to the set flow. By using SDN like this, control network operations can be flexibly and dynamically controlled on a program basis. Our idea is to isolate infected terminals from other terminals by using this technology.

This paper aims at detecting botnets using machine learning and deep learning and isolating a bot-infected machine. For botnet detection, learning is performed by using the data collected on flow-based from the botnet traffic captured on a conventional network and then evaluating the detection accuracy. For isolating a bot-infected machine in SDN, botnet traffic is retransmitted and communication with the source IP address determined by the machine learning classifier is blocked and quarantined.

Related work is discussed in section II of this paper. Section III presents the architecture and operation of the proposed system. The evaluation results are discussed in section IV. We conclude in section V with a brief summary and mention of future work.

II. RELATED WORK

Research on botnet detection is moving from signature-based technology to the network flow-based approach. For the detection of botnets, statistical features are extracted from the flow and then a machine learning algorithm of the classification is applied. In these approaches, the C4.5 decision tree algorithm of the supervised algorithm in machine learning is commonly used.

The approach proposed by Beigi et al. [3] is to reexamine the flow-based function and its relative effectiveness by including 40% or more of the botnet traffic data in the test data using a published data set rated by the author. In this approach, the feature quantities are divided into four groups in a time window of 60 seconds, feature values are selected with a greedy algorithm, and the algorithm of C4.5 decision tree is applied. The results showed a detection accuracy of 75% and a false detection rate of 2.3%. The authors stated that the generality, realism, and reproduction of the data set are important and that paying attention to the creation of the data set is crucial. However, as yet there have been no comparisons with the algorithms of other machine learning or concrete methods after detection.

Wijesinghe et al. [18] created variable length data by using IPFIX to make a data set. This allows for the removal of unnecessary attributes from the dataset, the addition of important attributes, etc., which is helpful when it comes to botnet detection. The decision tree classifier is effective for P2P botnet detection because it analyzes flow intervals and detects the C2C communication of centralized botnets such as HTTP and IRC botnets in SVM and Bayesian networks. However, the study does not mention any specific values of detection accuracy or concrete methods of dynamic quarantine of infected machines.

Kirubavathi et al. [8] generated many small packets to look for infectable machines. However, ideally we want to be using large packets. A key feature of the botnets is that the main role of a botnet's C2C channel is to establish a connection between the bot and the C2C server. This channel is reliable and its appearance should also be relatively harmless. Therefore, it is desirable that the packet carrying botnet commands or information gathered from an infected host be as small as possible to minimize the impact on the network. At the same time, C2C communication activities are constant throughout the lifecycle of a bot, as the bot needs to maintain connections with other bots in order to receive commands and updates. Therefore, it maintains a strict packet size uniformity throughout the lifecycle. In addition, there are outbound packets to the same IP address and port number with constant response time during bot communication of all incoming packets sent from several IP addresses and port numbers. Normal traffic is characterized by packet size randomness and communication pattern inhomogeneities. The results of a survey using the statistical characteristics of the traffic flow behavior collected over a certain time interval showed that the Bayesian network had the highest accuracy (99.14%) due to the flow characteristics at 180-second intervals. This demonstrates that the accuracy is the highest with few feature quantities and that the method is independent of botnet structure, command, and protocol.

Tariq et al. [17] proposed a flow-based method to detect botnets by applying a C4.5 decision tree-based machine learning algorithm under the SDN environment. In this method, flow streams are collected using OpenDayLight's Time Series Data Repository (TSDR) function. This task uses the key ¡source IP, destination IP, destination port, protocol¿ to identify the flows

of the two endpoints of the same application. Flows with the same key are grouped together and a batch is formed. Each flow in the batch is counted and fetched from the float race with every ten flows along with the history of the last 60 minutes of the source and destination IP windows. A data set of the features from these data is created and then applied to a machine learning algorithm. The results showed that the average detection rate was 97.1% for known botnets and that the accuracy was 90.4% for unknown botnets.

Letteri et al. [9] built a system that detects botnets using the southbound and the northbound APIs of SDN. It sends an OpenFlow message containing statistical information from the switch to the controller at a fixed time interval of the time window as with citeTariq2017, and then the controller sends to the classifier created in deep learning of five relu hidden layers and then added the block rule of the flow rule via the REST API message and isolated the infected host. The detection accuracy exceeded 99%, demonstrating that it is just as effective as other machine learning algorithms. These processes were automated and showed high detection accuracy.

Su et al. [15] detected a P2P botnet under the SDN environment. A machine learning algorithm based on the statistics of 600 seconds of flow from the OpenFlow switch was applied. Two classification modules were then applied, where they were classified into two or more types, and then ambiguous instances were classified correctly with the second module if they were unknown. Classification accuracy was 98.7% and 99.7% for the first and second modules, respectively. Block communication from the infected terminal was performed by transmitting the detection result with the REST API and setting a new block rule to the OpenFlow controller. These processes are automated, thus reducing the burden on network administrators. However, no other botnets than the P2P botnet are supported.

In light of the issues with the existing research highlighted above, in this research we do the following.

1) Extract features from session behavior flow patterns for each session.
2) Detect using bore net structure and protocol independent of deep learning.
3) Use OpenFlow to block external communication and isolate infected terminals using VLANs.
4) Test whether it can be used in a real environment.

Our approach differs from the existing research in that it extracts the feature amount in the flow for each session, not the time interval of the flow, performs detection by depth learning, and quarantines by VLAN segmentation using OpenFlow.

III. PROPOSED METHOD

A. Architectural Overview

The architecture of the proposed system is shown in the Fig. 1. The system has a feature extraction module, a botnet detection module, and a module for creating OpenFlow rules in the application layer. The process flow of the work is shown in the Fig. 2. The feature extraction module performs processing

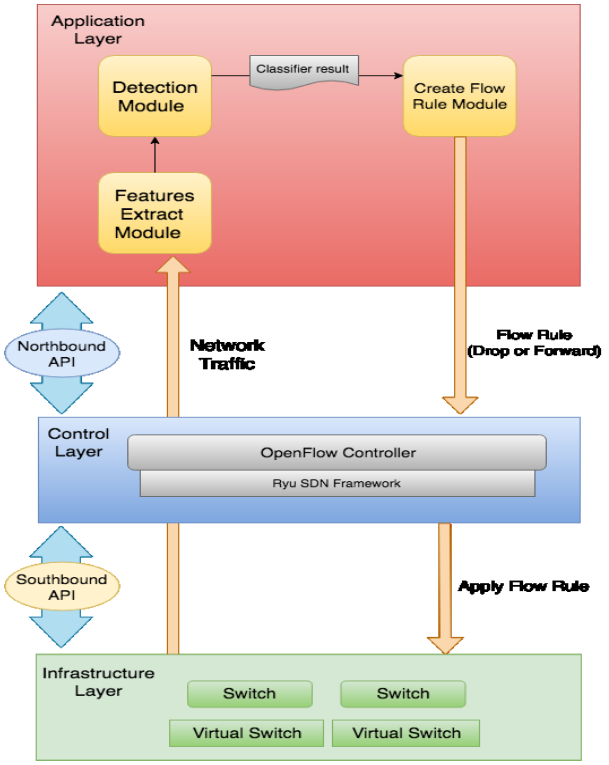

Fig. 1. Overview of system architecture.

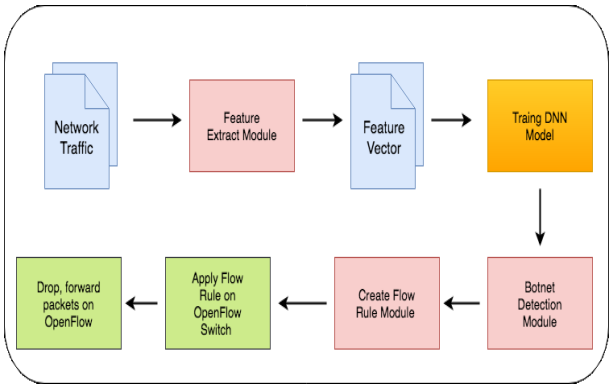

Fig. 2. Flow diagram of experiment system.

for each session, which is one unit for managing from the start to the end of communication, from the traffic data collected by the mirror port under the SDN environment, and then stores the header of the packets and statistical information of the session. The botnet detection module applies the classifier learned by MLP according to the feature set provided by the feature extraction module and classifies malicious traffic or normal traffic using pre-assigned labels. When there is botnet traffic classified by the botnet detection module, the module that creates an OpenFlow rule obtains a source IP address and a destination IP address from the session, along with a corresponding IP address for preventing external communication blocks and internal infection. Two rules are created to block a communication from a machine to a machine in the

TABLE I
FEATURES IN TRAFFIC SESSION

Statistics	Packet size (each direction) [SUM, MEAN, STD]
	Byte size (each direction) [SUM, MEAN, STD]
	Packet count (each direction) [SUM, MEAN, STD]
	Packet rate (each direction) [SUM, MEAN, STD]
	Byte rate (each direction) [SUM, MEAN, STD]
	Packet interval time [SUM, MEAN, STD]
	Session time
IP	IP address
	TTL [SUM, MEAN, STD]
	IP length [SUM, MEAN, STD]
TCP	Port number
	Number of flags (SYN, ACK, FIN, PSH, RST, URG, ECE, CWR)
	Window size [SUM, MEAN, STD]
	TCP payload size [SUM, MEAN, STD]
UDP	UDP payload size
	Port number
RAW	RAW length [SUM, MEAN, STD]

same network and the block communication is then sent to the OpenFlow controller. Details are given in the following subsections.

B. Deep Learning Approach

1) Feature Extraction Module: The proposed feature extraction model is collected programmatically from the pcap file. In the program, batch processing is performed for each communication session. However, since the size of the packet captured pcap file increases with time, when extracting a classifier model for detection, feature quantity extraction is done by parallel processing from data. The acquired features are mainly header information of the IP/TCP/UDP packet, session time, and statistical characteristics within the session time. Details of the features are shown in Table I. Learning is performed by using the features of the training data and the test data for the features described in the table. In packet capture, feature quantity is extracted for each session with this created program, and after extraction, these features pass data to the detection module using the REST API.

2) Detection Module: The proposed detection module classifies malicious traffic and normal traffic using MLP because the feature data has a numerical value. Moreover, we aim to detect early. During training, learning is done by assigning different labels to botnet traffic and normal traffic to the features described in the table. Deep learning is performed with five sigmoid hidden layers because many neurons in the MLP lead to longer computation time and can adversely affect the real-time performance of the detection module [9]. We use this model to classify test data. The test data is classified by a label assigned during training using the model of this classifier. When capturing packets, we receive the feature data of flows for each session and classify them using the classifier model.

C. SDN Approach

1) Flow Rule Module: In the module that creates a flow rule for OpenFlow, we first extract the IP address from the features of traffic judged as botnet traffic by the detection module. Second, we implement blocking of external communication using a firewall created with OpenFlow. In order to comply with the FW policy, we create a rule to block communication from the extracted source IP address to the outside. Third, the bot searches for the presence of other vulnerable machines in the same network. Therefore, to prevent internal infection, we create a flow rule that blocks internal communication. We also create a rule to flow the flow to the quarantine network for a bot quarantine created by VLAN for quarantining and investigating bots. After that, in order to apply the flow rule to the OpenFlow switch, the flow rule is sent to the OpenFlow controller with REST API, and it is then applied to the OpenFlow switch using the OpenFlow protocol.

IV. EVALUATION

In order to evaluate the proposed system, we first perform experiments to detect botnet traffic from malicious traffic and normal traffic and then test whether the rules for quarantine of infected terminals by SDN have been created and applied. The experimental environment used KVM bridge OpenvSwitch and the VLAN made access ports 10, 20, 30. The mirroring port setting was turned on and FW as an OpenFlow rule was set in advance. The configuration of the experimental environment is shown in Fig. 3.

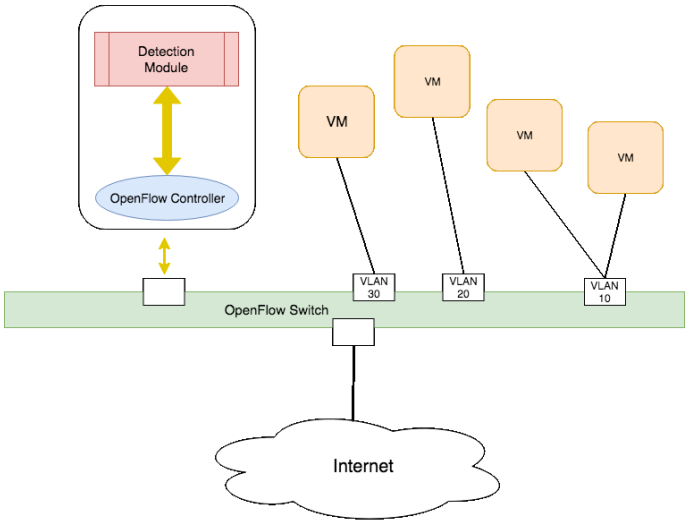

Fig. 3. Experimental configuration.

1) Detection: Published data set were used in the detection experiment. Specifically, botnet traffic used the CTU-13 data set [6] and normal traffic used data collected in our laboratory and the ISOT dataset [13]. These data are collected in the pcap file format and were used as training and test data. During training, the number of neurons, dropout rate, learning rate, batch size, and epoch number were adjusted and the most accurate model was used. For highly accurate models, we determined whether accurate models could be created using cross validation. Accuracy, Precision, Recall, F-measure using TP (true positives), FP (false positives), TN (true negatives), and FN (false negatives) were the parameters used to determine whether predictions and actual data labels were correct

or incorrect. The formula of each derivation is shown below. From these evaluations, we used classifier models with good results to classify from test data and feature quantities of data at packet capture.

$$Accuracy = \frac{TP}{TP + FP + TN + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$\frac{1}{F - measure} = \frac{1}{2}(\frac{1}{Recall} + \frac{1}{Precision})$$

2) SDN: For the quarantine experiment with SDN, we used the Ryu SDN Framework for the OpenFlow controller and OpenvSwitch for the OpenFlow switch. Many sample codes are published in Ryu, we can operate with the REST API's FW code and the modified switch code. The flow after application is shown in Fig. 4. First, we passed the data of the botnet traffic and detected session traffic in the detection experiment to the module that created the flow rule. The IP address was then extracted from the data. On the basis of that address, the FW policy was generated. In order to block external communication, a flow rule to block was created by specifying the source IP address extracted as the source address of the policy (1). Next, in order to block communication to an address in the same network, a rule to block internal communication was set by specifying the source address extracted in the source address of the policy and the network address of the same network as the destination address of the policy (2). Furthermore, by creating a flow rule that sends all communications of a bot-infected machine in VLAN 10 to VLAN 30, it is possible to investigate whether they are acting as an internal infection or are really infected (3).

3) Result: In this work, we used MLP's deep learning algorithm to detect hosts executing sessions of malicious traffic. Detection accuracy was 99.2%. The average detection accuracy rate was 98.7%. The confusion matrix as shown in Table II.The number, accuracy, and loss of training epoch (Fig. 5) reveals high accuracy and low loss. This detection accuracy showed higher results than [17] and [9], but it was lower than [15]. Also, the packet rate, byte rate, packet size, byte size and packet interval time are important in that the feature quantity is considered to be highly effective. In the SDN experiment, by changing the FW policy and the flow of VLAN, we were able to separate an external connection block from the network by using the REST API. By setting, the connection to the external network from the IP address of the infected host was blocked, and the connection to other hosts on the same network was also blocked.

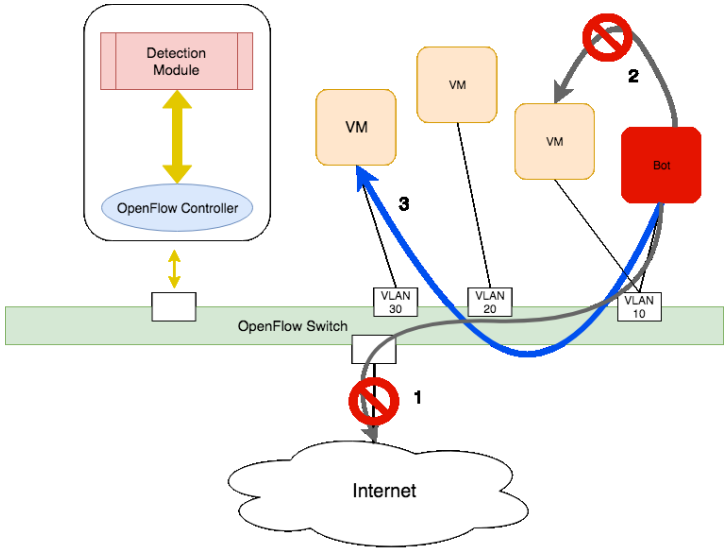

Fig. 4. Diagram of flow rule application after infection.

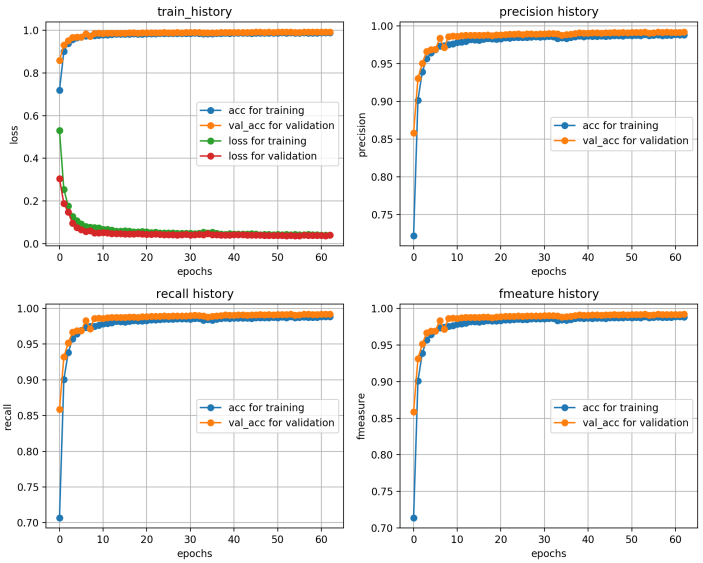

Fig. 5. Accuracy, loss, precision, recall, and F-measure for each epoch.

TABLE II
CONFUSION MATRIX

	Normal	Botnet
Normal	7310	198
Botnet	58	19517

V. CONCLUSION AND FUTURE WORK

In this work, we proposed a method that extracts feature quantity for each session from traffic data and isolates the infected machine after detection using MLP model of deep learning. As a result the detection rate was 99.2%. The detection rate was higher than that of existing methods. The isolation of an infected terminal using SDN could be achieved using a FW and VLAN. By using SDN, network control(FW and VLAN) could be dynamically enabled, and botnet traffic was detected by detecting the behavior of flow operation using machine learning. A limitation of this study is that we did not experiment on terminals that were actually infected with bots. In addition, although we extracted feature quantity for each session, we should also consider combination with feature extraction at time intervals to examine the tendency of small packet bursts as movements of internal bot infection of bot. For future work, it will be necessary to explore whether the network isolation can be performed on an actually infected host.

REFERENCES

[1] Dilara Acarali, Muttukrishnan Rajarajan, Nikos Komninos, and Ian Herwono. Survey of approaches and features for the identification of HTTP-based botnet traffic. *Journal of Network and Computer Applications*, 76(September):1–15, 2016.

[2] Francisco Villegas Alejandre, Nareli Cruz Cortes, and Eleazar Aguirre Anaya. Feature selection to detect botnets using machine learning algorithms. In *2017 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, pages 1–7. IEEE, 2017.

[3] Elaheh Biglar Beigi, Hossein Hadian Jazi, Natalia Stakhanova, and Ali A. Ghorbani. Towards effective feature selection in machine learning-based botnet detection approaches. In *2014 IEEE Conference on Communications and Network Security, CNS 2014*, 2014.

[4] Tanay Bhattacharya. Selected botnet detection techniques using flow data. Technical report.

[5] Yun-Chun Chen, Yu-Jhe Li, Aragorn Tseng, and Tsungnan Lin. Deep Learning for Malicious Flow Detection.

[6] S. García, M. Grill, J. Stiborek, and A. Zunino. An empirical comparison of botnet detection methods. *Computers and Security*, 45:100–123, 2014.

[7] Rahmadani Hadianto and Tito Waluyo Purboyo. A Survey Paper on Botnet Attacks and Defenses in Software Defined Networking. *International Journal of Applied Engineering Research*, 13(1):483–489, 2018.

[8] G. Kirubavathi and R. Anitha. Botnet detection via mining of traffic flow characteristics. *Computers and Electrical Engineering*, 50:91–101, 2016.

[9] Ivan Letteri, Massimo Del Rosso, Pasquale Caianiello, and Dajana Cassioli. Performance of Botnet Detection by Neural Networks in Software-Defined Networks.

[10] Nick Mckeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: Enabling Innovation in Campus Networks. Technical report.

[11] Roberto Perdisci, Wenke Lee, and Nick Feamster. Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces. Technical report.

[12] Paul Prasse, Lukas Machlica, Tomas Pevny, Jiri Havelka, and Tobias Scheffer. Malware Detection by Analysing Network Traffic with Neural Networks. *2017 IEEE Security and Privacy Workshops (SPW)*, i:205–210, 2017.

[13] Sherif Saad, Issa Traore, Ali Ghorbani, Bassam Sayed, David Zhao, Wei Lu, John Felix, and Payman Hakimian. Detecting P2P botnets through network behavior analysis and machine learning. *2011 Ninth Annual International Conference on Privacy Security and Trust*, pages 174–180, 2011.

[14] Matija Stevanovic, Jens Myrup Pedersen, M Stevanovic, and J M Pedersen. On the Use of Machine Learning for Identifying Botnet Network Traffic. *Journal of Cyber Security*, 4:1–32, 2016.

[15] Shang-Chiuan Su, Yi-Ren Chen, Shi-Chun Tsai, and Yi-Bing Lin. Detecting P2P Botnet in Software Defined Networks. *Security and Communication Networks*, 2018:1–13, jan 2018.

[16] Tuan A Tang, Lotfi Mhamdi, Des Mclernon, Syed Ali, Raza Zaidi, and Mounir Ghogho. Deep Learning Approach for Network Intrusion Detection in Software Defined Networking. Technical report.

[17] Farhan Tariq and Shamim Baig. Machine Learning Based Botnet Detection in Software Defined Networks. *International Journal of Security and Its Applications*, 11(11):1–12, 2017.

[18] Udaya Wijesinghe, Udaya Tupakula, and Vijay Varadharajan. Botnet detection using software defined networking. *2015 22nd International Conference on Telecommunications (ICT)*, (June):219–224, 2015.