Distributed Security Network Functions against Botnet Attacks in Software-defined Networks

Younghee Park
Computer Engineering Department
San Jose State University
San Jose, CA
younghee.park@sjsu.edu

Nikhil Nikhil Vijayakumar Kengalahalli
Computer Engineering Department
San Jose State University
San Jose, CA
nikhilvijayakumar.kengalahalli@sjsu.edu

Sang-Yoon Chang
Computer Science Department
University of Colorado Colorado Springs
Colorado Springs, CO USA
schang2@uccs.edu

***Abstract*— For the past decade, botnets have dominated network attacks in spite of significant research advances in defending against them. The distributed attack sources, the network size, and the diverse botnet attack techniques challenge the effectiveness of a single-point centralized security solution. This paper proposes a distributed security system against large-scale disruptive botnet attacks by using SDN/NFV and machine-learning. In our system, a set of distributed network functions detect network attacks for each protocol and to collect real-time traffic information, which also gets relayed to the SDN controller for more sophisticated analyses. The SDN controller then analyzes the real-time traffic with the only forwarded information using machine learning and updates the flow rule or take routing/bandwidth-control measures, which get executed on the nodes implementing the security network functions. Our evaluations show the proposed system to be an efficient and effective defense method against botnet attacks. The evaluation results demonstrated that the proposed system detects large-scale distributed network attacks from botnets at the SDN controller while the network functions locally detect known attacks across different networking protocols.**

Keywords—Software-defined networking, network Function Virtualization, Botnets, machine-learning

I. INTRODUCTION

Massive compromised distributed hosts called botnets continue to interrupt and launch various network attacks on the Internet and other networked system applications, such as the Internet-of-Things (IoT), Smart Grid, and health systems [2, 3, 4]. With the proliferation of botnets, attackers through botnets illegitimately access networked devices and launch various network attacks. For example, Mirai and its variants have attacked the Internet infrastructure to disrupt communications and operations [5, 6], launching more than several thousand attacks in 2017 [6]. The complexity of diverse heterogeneous networked devices and applications along with diverse network protocols makes it difficult to develop a generalized and integrated security framework against botnet-related attacks.

Software-defined networking (SDN), providing the flexibility and the visibility of networks through programmable forwarding decisions, has changed the direction of defense methods against various network-based attacks [25, 28, 29, 30]. With Network Function Virtualization (NFV), replacing hardware middleboxes with software-based network appliances in virtualization, SDN enables us to develop a scalable and elastic defense against increasingly disruptive distributed denial-of-service attacks [7, 9, 27]. Many security solutions using SDN/NFV to defend against network attacks have been proposed for the past decade [6, 8, 9, 10, 11, 20, 26, 27, 13, 14, 16, 29]. They have mostly focused on network security issues related to the infrastructure of SDN/NFV itself [20, 27]. Some previous research has addressed network intrusion detection systems by implementing security applications on top of the SDN controller [9, 14, 29]. These solutions assume that all the traffic must first be transferred to the controller, which creates a bottleneck hampering timely intrusion detection. The overhead of having the traffic go through the controller only gets exacerbated with botnets because of the diversity of the routing paths that the attack traffic will take, further challenging the practicality of the solutions in the literature. In addition, the large scale of distributed botnet attacks and their various malicious actions create challenges in achieving scalability and flexibility as network size explodes, low latency to respond to network dynamics, and easy integration with different network types and applications.

In this paper, we propose an SDN/NFV-based security framework to detect and mitigate botnet attacks by using SDN/NFV with a machine-learning technique. The proposed system implements virtualized network functions to immediately detect network attacks in the data plane by using NFV and the SDN controller to detect distributed stealthy botnet attacks using machine-learning in conjunction with NFV. In other words, the defense framework is divided in two parts: the data plane and the control plane. NFV in the data plane manages virtual network functions (VNF) in order to detect network attacks and to make a service chain of the VNFs according to traffic types. In addition, one of the VNFs keeps monitoring traffic and collecting the feature set information to be transferred into an SDN controller for machine-learning-based intrusion detection. The SDN controller in the control plane makes a routing path decision and orchestrates VNF security services into the Internet with network programmability. With a machine-learning-based network traffic analysis, the controller detects and defends against abnormal network behavior with the real-time feature sets from the VNF.

In the proposed system, the SDN controller works with the virtualization manager to monitor network behavior and to

collect data from source to destination. It performs two steps to detect botnet attacks. The first detection action is performed by each network function, each targeting a specific attack related to each network protocol. The second detection action is conducted by machine-learning analysis in the SDN controller by extracting real-time traffic characteristics. Through this two-level defense method, the SDN controller detects, or in the alternative mitigates, potential botnet attacks and its variants by dropping malicious traffic. Therefore, our system not only detects and analyzes an attack but also prevents the propagation of the attack impact via defensive routing. Compared with the previous works, which forward all the traffic to the controller for intrusion detection, the proposed system needs to send only feature set information to the SDN controller. The controller can easily obtain real-time traffic information through one of the virtualized security network functions, called feature extractor in the data plane. In addition, well-known attacks can be immediately detected by individual VNFs to prevent a potential protocol vulnerability in a network.

This paper contributes to botnet security in the following aspects. First, we proposed an integrated multi-dimensional network intrusion detection system (NIDS) against various botnet attacks based on the emerging network technique of SDN and NFV. Second, we achieve a real-time machine-learning-based NIDS in the SDN since our designed VNF keeps extracting feature sets from real-time traffic and sends them to the SDN controller for further detection of distributed network attacks. Third, to the best of our knowledge, we are the first to integrate machine-learning with both SDN and NFV while VNFs keep extracting traffic feature sets for real-time data analysis. Most machine-learning-based intrusion detection systems have limitations to analyze real-time traffic. Fourth, by using NFV and its flexibility, we implemented many security network functions for various network protocols to detect network attacks. Our system design provides customizable IDS with high cost-efficiency, and high flexibility and elasticity to deploy VNFs. Finally, we evaluate the proposed system and compare it to historical malicious data sets and simulated botnet traffic to demonstrate the effectiveness and the efficiency of the proposed system.

The paper is organized as follows. Section II summarizes related work. Section III presents our proposed system and explains each component of the proposed system in detail. Section IV explains our implementation and presents our experimental results to show our system's effectiveness. Finally, we discuss our method and our future plans in Section VI and present our conclusion in Section V.

II. Related Work

A botnet consists of compromised hosts (i.e. bot agents), a botmaster, and command and control (C&C) servers to carry out malicious activities on the Internet. Most of botnets utilize DNS to host their command and control (C&C) servers and they keep communicating with each other via HTTP to receive commands and to send their results to the remote servers managed by botmasters. Intensive traffic analysis for HTTP and DNS has been performed to detect and mitigate botnet-related attacks [5, 6, 28, 52]. Maryman and Alireza summarized all possible botnet detection methods: signature-based, anomaly-based, DNS-based, mining-based botnet detection approaches [28]. However, most of these rely on botnet protocols and, except for one DNS-based botnet detection method, none of them allow us to detect botnets in real-time [28]. In spite of extensive efforts, botnets still proliferate to negatively impact networks. The anonymity of botmasters and various communication protocols over a wide range of network topologies make it difficult to defend against botnet attacks [28]. Therefore, to effectively and efficiently monitor and defend against such large-scale distributed botnets, a new security framework is needed for real-time monitoring and detection system.

The widespread use of SDN in recent years has brought with it a number of new security issues involving the three main layers of SDN: the application layer, the control plane (controller), and the data plane [12, 17, 25, 30]. One problem is that network applications in the controller threaten the controller's resilience because fallacious applications can cause fatal instabilities, unexpected vulnerabilities, and loss of network control [20, 26, 27]. Network intrusion detection systems in SDN using machine-learning have been proposed to detect network attacks [13, 14, 16, 29]. They are limited to monitor real-time traffic and focused on data analysis without real-time flow information and NFV. Hardware-based NFs present significant drawbacks including high costs, management complexity, slow time to market, and non-scalability, to name a few [18, 19]. Different control frameworks for virtualized NFs have recently been proposed to address the safe scaling of virtual network functions [21, 22, 23, 24]. Fayaz et al. proposed Bohatei, a flexible and elastic virtual DDoS defense system to effectively defend against DDoS attacks [7]. While Bohatei focused on only DDoS attacks by using only SDN/NFV with Bro, we developed our own security network functions to detect various network attacks and botnet attacks based on machine-learning with NFV/SDN. To achieve our ultimate goal, NFV monitors and collects real-time traffic information under the control of SDN.

III. Our Approach

A. System Overview

To defend against network attacks generated by botnets, our paper proposes a multi-dimensional intrusion detection system by using SDN and NFV. Our proposed system is scalable and elastic, enabling it to monitor large networks by analyzing incoming traffic and collecting traffic information at various entry points on the Internet. Figure 1 shows the overall system architecture using SDN and NFV to defend against botnet attacks, utilizing the full functionalities of SDN and NFV to monitor malicious traffic in real time.

As in Figure 1, the SDN controller has three main functions; (1) making routing path decisions to maximize network utilization and fast virtual NIDS orchestration services, as described in Section III C; (2) performing machine-learning based network monitoring to detect malicious traffic by using the Random Forest algorithm, which might be missed by the VNF, and thus improve attack accuracy as explained in Section Section III D; and (3) managing flow

Identify applicable funding agency here. If none, delete this text box.

rules to change network behavior, e.g., bandwidth control, through the programmable data plane in switches, as presented in Section III E.

The NFV in Figure 1 plays an important role in defending against common protocol-specific attacks by using different network functions (NF) for different protocols. First, it provides the intrusion detection VNF management to maintain a pool of NFs with a virtual machine manager, as described in Section III B. Second, it dynamically spawns a chain of VNF services according to traffic characteristics instead of monolithic bulky virtual instances, as in Section III C. Lastly, one of the VNFs collects various information from real-time incoming traffic to be provided for machine-learning-based NIDS in the SDN controller. Figure 2 presents detailed system components to achieve our ultimate goal. In the following subsections, we now describe each component.

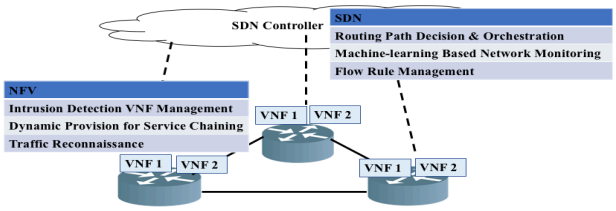

Fig. 1. An integrated multi-dimensional network intrusion detection system based on SDN/NFV

B. *Virtual Network Functions for Intrusion Detection*

We first implement individual intrusion detection functions for protocol-specific network appliances for virtual network functions, which we refer to as the micro intrusion detection network function (*mVNF*). Figure 2 shows its components: a traffic classifier, protocol-dependent VNFs, and a feature extractor with service-chaining by a virtual machine manager. The virtual machine manager virtualizes a set of mNFs to provide micro-services in NFV with unique features: cost-efficiency in resources, customization, scalability, and reusability since the micro-services architecture breaks down a single large application into smaller components to individually utilize a particular service in a scalable and flexible way [32, 33]. Instead of using the existing bulky monolithic IDS, we design each intrusion detection component for each communication protocol from the application layer to the network layer to include various protocols popularly used in botnets [34], such as HTTP and DNS.

The set of micro-NF (each of which are denoted by $mVNF_i$ where i distinguishes the virtual network security functions) is implemented to defend against well-known network attacks related to each network protocol, such as HTTP, DNS, TCP, UDP, ARP and IP. For example, if botnets are using a DNS protocol to identify their C&C servers, our system needs to examine traffic with the VNFs of DNS and UDP. In addition, when bots deliver malware into the Internet, the host needs to be compromised, for example by spoofing based on the ARP protocol. Thus, the first defense method should detect such ARP-related attacks to prevent bots from compromising hosts. The proposed system implements an individual $mVNF_i$ function for each network protocol while considering various attack techniques. In addition, the traffic classifier is another $mVNF_i$ function to identify traffic types according to each protocol, as shown in Figure 3. The traffic classifier classifies incoming traffic types to form the correct service chain of VNFs for each protocol. Depending on the traffic type, a set of VNFs needs to be spawned to monitor traffic according to traffic characteristics.

The most important function of VNFs is a feature extractor ($mVNF_i$) in the data plane for the machine-learning-based intrusion detection in the control plane. The feature extractor keeps monitoring incoming traffic and extracting information for feature sets that must be used for machine-learning analysis for intrusion detection in the controller. The number of feature sets ranges from ten to several hundreds, depending on different machine-learning algorithms. Because of performance bottlenecks in the switches, a lot of data for feature sets cannot be extracted inside switches even though the statistical manager in the SDN controller keeps polling the traffic statistical information for simple traffic monitoring. Therefore, it would be critical to implement a separate VNF dedicated to feature extraction for machine-learning for timely detection. The feature extractor function keeps extracting real-time traffic features, such as IP addresses, port numbers, service types, packet sizes, which are transferred into the SDN controller for machine-learning analysis for intrusion detection. Botnets usually launch distributed attacks by coordinating with a group of bots remotely. Stealthy distributed botnet attacks can be identified by machine-learning analysis in the control plane with real-time traffic information. Thus, our proposed system implements a special purpose virtual network function which mainly extracts all packet information to be used for feature sets across the Internet. The machine-learning method for IDS in the controller is described in detail in Section III D. For example, if the traffic classifier identifies incoming traffic as HTTP traffic, our system makes a chain of at least three *mVNF* functions such as, HTTP, TCP, and the feature extractor. Therefore, to monitor network attacks related to HTTP traffic, we have a virtual intrusion detection network function (VNIDS) with these three network functions expressed as $VNIDS_i = \{mVNF_1, mVNF_2, mVNF_3\}$ (Note that i is the number of $mVNF_i$). Any set of VNIDS can be distributed throughout the Internet.

Each micro-NF ($mVNF_i$) examines standard protocol specifications according to each RFC for each protocol in order to validate packet header information compliance and also includes a detection method to identify well-known network attacks related to each protocol, such as HTTP attacks [48, 55], DNS [40, 41], TCP attacks [42, 43, 44, 45], UDP attacks [34], ARP attacks [15], and IP attacks [31]. Due to space limitations, we have omitted a detailed explanation of each attack related to each protocol, and explain only HTTP and DNS protocols, which are the main protocols used by botnets. First, we verify all HTTP and DNS header information according to RFC 7231 and RFC 1035 and related standard documents. We implement a detection method to identify known DNS attacks such as DNS bomb attacks, DoS attacks using recursive DNS queries, and DNS amplification attacks. The micro-NF ($mVNF_i$) for HTTP protocols can detect HTTP flooding attacks by using GET or POST commands. To detect such attacks, we check a

specific value to detect a specific action, for example setting a recursive bit in DNS or setting a threshold to count the number of particular packets to detect DoS attacks. In addition, we compare source IP addresses with destination IP addresses for the same origin security policy to detect a man-in-the-middle attack by using ARP. Details of implementation are explained in Section IV.

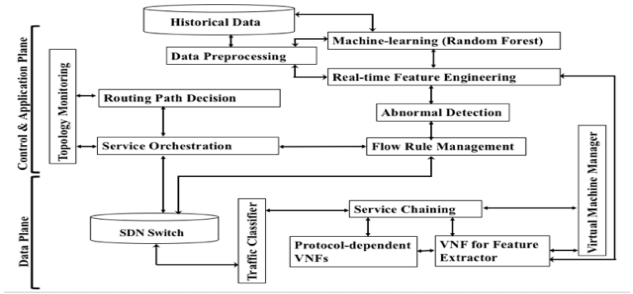

Fig. 2. The procedure of each componet for the proposed distributed security framework based on SDN/NFV.

C. Routing Path Decision and Orchestration

The SDN control plane keeps monitoring current network topology and network bandwidth usage to determine the best path from source to destination instead of the shortest path routing algorithm popularly used for routing in SDN. To measure effective bandwidth, the controller polls byte rates (i.e. transmission rates) for each port in the switches. The controller computes the maximum bandwidth by evaluating all possible paths (P_{ij} = {P_{12}, P_{13}, ... , P_{ij}}, Note that i,j= {1, ... , n}) between a source (i) and a destination (j). The controller then selects the highest available bandwidth path after considering all paths. The total bandwidth is $B=\Sigma(b_{ij})$ (i.e. i,j= {1, ... , n}) for all paths between two nodes after summing each link bandwidth b_{ij} *for each path* P_{ij}. Note that b_{ij} is the link bandwidth between a node N_i and its neighboring node N_j (not a destination node). Therefore, the controller chooses the most available bandwidth (i.e. the maximum total bandwidth (B)) from source to destination.

With the maximum available bandwidth, our proposed system maximizes network utilization based on high transmission rates and fast traffic processing times to efficiently provide our security services. When the controller determines a routing path, the controller orchestrates security services in the designated path by making a chain of security services with the virtual machine manager in NFV on switches. Figure 3 shows an example of service chaining for NFV orchestration in the proposed system. The SDN controller keeps communicating with the virtual machine manager to provide VNF services for network intrusion detection on designated paths from source to destination. Figure 3 shows security services for HTTP, DNS, TCP, UDP and IP along with the traffic classifier used to recognize specific traffic types. The service chain of VNFs can be deployed linearly or in parallel. When the traffic classifier recognizes traffic types, other VNFs can examine traffic in parallel. For example, after classifying HTTP traffic using the traffic classifier, the VNF for HTTP, the VNF for TCP, and the VNF for the feature extractor can examine traffic in parallel for fast traffic processing time.

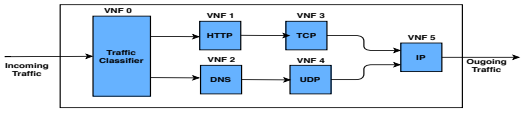

Fig. 3. VNF orchestration services

D. Machine-learning-based Network Monitoring

As shown in Figure 2, the SDN controller continues to monitor real-time traffic using a machine-learning algorithm to detect malicious traffic. First, with the public botnet dataset (CTU-13 [53]) as described in Section IV, the proposed system has an attack model generated by the Random Forest algorithm [37]. Our proposed system does not focus on machine-learning techniques themselves, so we will not detail the machine-learning techniques used to generate an attack model by using the Random Forest algorithm. We used eleven feature sets to generate an attack model by using the public dataset. The feature sets included traffic information including source/destination addresses and port numbers, flags, type of services, protocols, and bytes. The SDN controller collects the feature set information of real-time traffic using the feature extractor network function in the data plane, retains the attack model with the public dataset using the Random Forest algorithm in the control plane, and utilizes the attack model to detect real-time botnet attacks.

Different from typical machine-learning-based intrusion detection systems, which rely solely on historical data analysis for network intrusion, the proposed system also analyzes and collects real-time traffic information through the feature extractor as a virtual network function in NFV. With the help of NFV, the controller effectively uses the generated attack models to monitor real-time traffic. In other words, the attack model is generated with well-defined historical datasets. The distributed feature extraction network function on the Internet continues to monitor and collect the feature information in the data plane to be used by the machine-learning algorithm in the controller. Most machine-learning techniques require a huge set of feature information, which overwhelms the switches, slowing transfer to the statistical manager in the SDN controller; hence, our designed feature extraction virtual network function in NFV plays an important role in addressing this problem and providing real-time traffic analysis to improve machine-learning-based intrusion detection in SDN with NFV. The controller makes a decision as to whether incoming traffic in SDN is malicious or not within a given confidence score by matching incoming traffic with the generated attack models. The controller drops incoming traffic when the match with existing attack models exceeds that threshold.

E. Flow Rule Management

As a threat response module, the flow rule manager in the SDN controller in Figure 2 reacts to intrusion detection by controlling current bandwidth and installing new flow rules. When abnormal behavior is detected, the SDN controller

Identify applicable funding agency here. If none, delete this text box.

pushes new flow rules to drop malicious packets, such as DoS/DDoS and other network attacks, or to reduce bandwidth to slow down massive traffic, such as elephant flows or unknown traffic, which degrades network performance. There are many network attacks that can be countered by just modifying the network bandwidth to some extent [35, 36]. The value of the bandwidth is determined by the controller, making it possible for the system to scale up and scale down the bandwidth as needed. The virtual security network functions immediately drop malicious packets when attacks are detected by virtual security NFs in NFV, e.g., deterministic attack patterns. Based on these threat response methods in the SDN controller, the proposed framework autonomously mitigates network abnormalities while improving overall network utilization.

IV. Evaluation

A. Implementation and Setup

We implemented our proposed system by using ClickOS [38] with Click version 2.0.1 and Xen hypervisor version 4.4.1 and Floodlight version 1.2 [39]. ClickOS is an open source network function virtualization platform, consisting of a set of click elements to develop various network functions [38]. We implemented eight separate click files (HTTP, DNS, TCP, UDP, IP, ARP, a traffic classifier, and a feature extractor) into ClickOS by adding many new click elements to cover various network attacks for each protocol. For example, the traffic classifier was extended by utilizing IPFilter and IPClassifier in ClickOS to classify various protocols including the application protocols like HTTP and DNS. The DNS click file includes a function to examine the DNS header information and to detect DNS-related attacks, such as DNS bomb attacks [40], DNS recursive query attacks [41], and DNS flooding attacks. The TCP click file includes a detection function for TCP SYN flooding attacks, TCP RST attacks, and attacks related to TCP sequence numbers [42, 43, 44, 45]. The machine-learning system in Floodlight was implemented in Python and the flow rule management was implemented in REST APIs to push commands to change flow tables through the SDN controller.

Figure 3 shows our testbest in CloudLab, an open cloud-based laboratory supported by NSF [1]. We created six hosts in different servers in CloudLab and three hosts were installed with Open vSwitch version 2.3, one SDN controller and two end hosts. Each host had Ubuntu 14.0 with two Intel core 2.9 GHz with 8GB RAM. For malicious traffic, we used BoNeSi [46], the DDoS botnet simulator to generate HTTP flooding and DNS attacks. As described in Section III D, to generate an attack models for botnets, we used CTU-13 datasets [53] which were botnet traffic datasets collected by CTU University in Czech Republic. The CTU-13 datasets included various botnet attacks like spam and DDoS attacks with IRC, P2P or HTTP protocols.

Fig. 4. Testbed in CloudLab

Identify applicable funding agency here. If none, delete this text box.

B. Experimental Results

We evaluated various aspects of our proposed system, including processing time, CPU usage, throughput, and network latency. In addition, we tested the performance of our feature extractor with machine-learning analysis in the SDN controller.

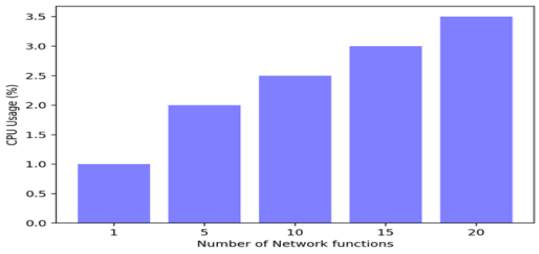

Fig. 5. CPU Usage

CPU Usage: Figure 5 shows the CPU usage according to the number of network functions. As the number of each VNF increased, the CPU usage also increased. However, up to 20 VNFs, the proposed system used around 3.5% of CPU in total. For memory usage, the entire ClickOS only required 5MB with quick booting time as presented in [38]. Regardless of the number of VNFs, our proposed system also required somewhat less than 5MB.

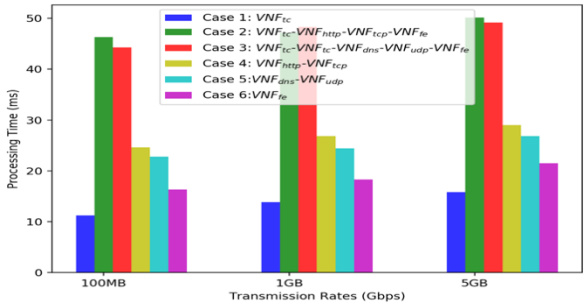

Fig. 6. Processing Time (Note that the results shows the combination of different by $mVNF_i$. VNF_{tc} is a traffic classifier. VNF_{fe} is a feature extractor. The other *VNF* is related to each protocol.)

Processing Time: Figure 6 presents traffic processing time at different settings of various VNFs in a single switch. We generated different transmission rates from source (Host 1) to destination (Host 2) with different combinations of VNFs in linear or in parallel. Both Case 2 and Case 3, which have linear service chains of four or five VNFs, showed the slowest processing times since the traffic had to be linearly and individually examined. Except for these two cases, other test cases demonstrated that each VNF is able to nearly double the processing speed of analyzing all the traffic. As traffic size increased, the processing time also increased only slightly in all cases, even when many NFV functions were running in parallel.

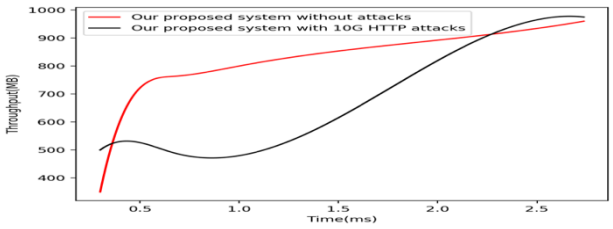

Fig. 7. The througput of our system

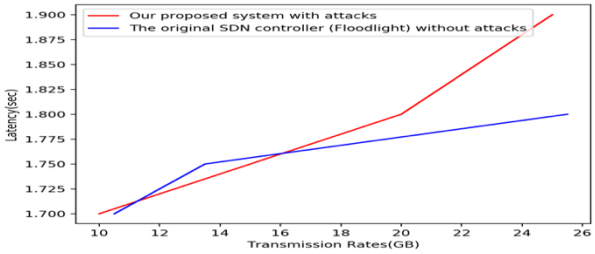

Fig. 8. The network lantency of our system

Network Performance: Figures 7 and Figure 8 show throughput and network latency of our proposed system with or without attacks to demonstrate the effectiveness of our system by showing the throughput increase/recovery as the network controller adapts the network flow control. 1GB of benign traffic, excluding the attack traffic, was sent from Host 1 to Host 2. In Figure 7, to evaluate our proposed system, we generated 10GB of attack traffic by using BoNeSi. The experimental results demonstrated that with attack traffic, our system using SDN/NFV can rapidly recover the original throughput and small network latency. Figure 8 shows the network latency at different transmission rates. We generated the same amount of attack traffic with each transmission rate to measure the network latency. For example, when we sent 10GB traffic, we also generated 10GB DNS attack traffic by using BoNeSi. Under serious attacks, our system showed a little network latency which was comparable to the original SDN system without attacks. Therefore, these experiments demonstrated the effectiveness of the proposed system to detect attacks using SDN and NFV.

Performance of machine-learning analysis: We tested the botnet attack datasets from CTU-13 datasets [53] to generate an attack model in the SDN controller. The dataset had a total of 10,48,576 network flows with a total data size of around 24MB. The dataset included 500 botnet flows and 500 normal flows, as described in [52], we used the ten features, such as duration, source and destination addresses, source and destination port numbers, service types, total packets and bytes, protocol, flags. 70% of the datasets used were for training sets, and the rest of the datasets were used for testing sets by randomly selecting data. When the confidence socre is 0.5, our results showed 100% accuracy along with a 92% true positive rate and a 10% false positive rate.

To test real-time botnet attack detection with the generated attack models of the CTU-13 datasets by using SDN/NFV, we evaluated synthetic traffic generated by the BoNeSi simulator, and real Mirai traffic generated by the public Mirai code [49]. To capture the Mirai traffic, we ran the Mirai code in Raspberry Pi and captured traffic using WireShark. In addition, we generated BoNeSi traffic to evaluate our proposed system. The feature extractor in ClickOS collected feature information for the captured synthetic and real botnet traffic (i.e. Mirai) for machine-learning analysis in the controller. Table I shows the results of our machine-learning analysis with real-time traffic. For BoNeSi, we collected 34,915 flows in total with 14MB. We obtained 6,930 Mirai flows in total with 831KB. Our results for Mirai traces showed 100% accuracy along with a 98% true positive rate and an 11% false positive rate. BoNeSi traces resulted in 98% accuracy with a 92% true positive rate and a 12% false positive rate. As expected, with the historical archived known attack data sets, the machine-learning algorithm gave good results to detect botnet attacks. As future work, we will reduce false positive rates for real-time malicious traffic with the trained attack model through a sophisticated feature engineering technique [50, 51].

TABLE I. THE RESULTS OF THE RANDOM FOREST ALGORITHM

	CTU-13 Dataset	BoNeSi	Mirai
True Positive	**0.92**	**0.92**	**0.98**
False Positive	**0.10**	**0.12**	**0.11**
Accuracy	**1.00**	**0.98**	**1.00**

V. DISCUSSION AND CONCLUSION

This paper proposes a distributed security system using virtual security functions in NFV with an SDN controller to defend against botnet attacks. The security network functions detect well-known protocol-specific attacks while continuing to extract traffic feature set information in the data plane for further machine-learning analysis in the controller. By combining SDN with NFV, the proposed system can monitor incoming traffic on the switches and detect malicious behavior in a network in real time, while matching real-time traffic information with an attack model generated by machine-learning. This work produced an initial integrated security system based on SDN/NFV by utilizing the main functionalities of those new technologies combined with machine-learning. To improve our proposed system, we first need to cover more protocols and attacks to defend against general attacks and botnet attacks. Second, attackers can generate false datasets to defeat machine-learning analysis. To improve our machine-learning analysis, we need to distinguish fake datasets and authentic datasets through more sophisticated virtual security network functions against adversarial machine-learning problems. Last, we can embed more threat response methods instead of controlling only flow rules.

ACKNOWLEDGMENT

The work is supported by NSF SaTC # 1723804.

REFERENCES

[1] CloudLab. https://www.cloudlab.us

[2] "A smarter grid with the Internet of Things". Texas Instruments.

[3] Hwang, Jong-Sung; Choe, Young Han "Smart Cities Seoul: a case study" (PDF). ITU-T Technology Watch. Retrieved 23 October 2016.

[4] Zanella, Andrea; Bui, Nicola; Castellani, Angelo; Vangelista, Lorenzo; Zorzi, Michele "Internet of Things for Smart Cities". IEEE Internet of Things Journal. **1** (1): 22–32. Retrieved 26 June 2015.

[5] C. Kolias, G. Kambourakis, A. Stavrou and J. Voas, "DDoS in the IoT: Mirai and Other Botnets," in Computer, vol. 50, no. 7, pp. 80-84, 2017.

[6] Manos Antonakakis, Tim April and et. al., "Understanding the Mirai Botnet," in USENIX Security Symposium, Vancourver, BC 2017.

[7] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: flexible and elastic ddos defense," in 24th USENIX Security, 2015.

[8] Younghee Park, Chungsik Song, et. al., "Machine-learning based Threat-aware System in Software Defined Networks," In Proc. the 26th International Conference on Computer Communications and Networks (ICCCN), July 31-August 3, 2017, Vancouver, Canada, 2017.

[9] Juan Deng, Hongxin Hu, Hongda Li, et. al., "VNGuard: An NFV/SDN Combination Framework for Provisioning and Managing Virtual Firewalls," IEEE Conference on NFV-SDN, USA, November 2015.

[10] Younghee Park, Pritesh Chandaliya, et. al., "Dynamic Defense Provision via Network Functions Virtualization," ACM International Workshop on SDN-NFV Security, Scottsdale, Arizona, USA 2017.

[11] Nuyun Zhang, Hongda Li, Hongxin Hu, Younghee Park, "Towards Effective Virtualization of Intrusion Detection Systems," ACM International Workshop on SDN-NFV Security, Arizona, USA 2017.

[12] D. Kreutz, F. M. Ramos, et. al. "Software-defined networking: A comprehensive survey," In Proceedings of the IEEE, vol. 103, 2015.

[13] R. Sommer, V. Paxson, Outside the closed world: On using machine learning for network intrusion detection, in Proceedings of IEEE Symposium on Security and Privacy (SP), 2010.

[14] T. Xing, D. Huang, L. Xu, C.-J. Chung, P. Khatkar, Snortflow: A open flow-based intrusion prevention system in cloud environment, in IEEE GENI Research and Educational Experiment Workshop (GREE), 2013

[15] Lockhart, Andrew, "Network security hacks," O'Reilly. p. 186. ISBN 978-0-596-52763-1, 2007.

[16] T. Xing, Z. Xiong, D. Huang, D. Medhi, "Sdnips: Enabling software-defined networking based intrusion prevention system in clouds," in Proceedings of 10th International Conference on Network and Service Management (CNSM), 2014.

[17] S. Shin and G. Gu, "Attacking software-defined networks: A first feasibility study," in Proceedings of ACM SIGCOMM workshop on Hot topics in software defined networking (HotSDN'13), 2013.

[18] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, et. al., "Making middleboxes someone else's problem: network processing as a cloud service," ACM SIGCOMM Computer Communication Review, 2012.

[19] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, "Design and implementation of a consolidated middlebox architecture," in Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, 2012.

[20] S. Hong, L. Xu, et. al., "Poisoning network visibility in software-defined networks: New attacks and countermeasures," in Proc. of the Network and Distributed System Security Symposium (NDSS), 2015.

[21] S. Rajagopalan, D. Williams, and H. Jamjoom, "Pico replication: A high availability framework for middleboxes," in Proceedings of the 4th annual Symposium on Cloud Computing, 2013.

[22] S. Rajagopalan, D. Williams, H. Jamjoom, and A. Warfield, "Split/merge: System support for elastic execution in virtual middleboxes," in Proceedings of NDSS, February 2013.

[23] D A. Gember-Jacobson, R. Viswanathan, et. al., "Opennf: Enabling innovation in network function control," in Proc. of the 2014 ACM Conferenceon SIGCOMM, 2014.

[24] A. Gember-Jacobson and A. Akella, "Improving the safety, scalability, and efficiency of network function state transfers," in Proc. of the ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization, 2015.

[25] S. Sezer, S. Scott-Hayward, et. al., "Are we ready for sdn? implementation challenges for software-defined networks," Communications Magazine, IEEE, July 2013.

[26] A. Khurshid, W. Zhou, M. Caesar, and P. Godfrey, "Veriflow: verifying network-wide invariants in real time," ACM SIGCOMM Computer Communication Review, 2012.

[27] H. Wang, L. Xu, and G. Gu, "Floodguard: A dos attack prevention extension in software-defined networks," in Proc. of IEEE International Conference on Dependable Systems and Networks (DSN), 2015.

[28] Maryam Feily and Alireza Shahrestani, "A survey of botnet and botnet detection," In Proceedings of International Conference on Emerging Security Information, Systems and Technologies, 2009.

[29] C. Yoon, T. Park, S. Lee, H. Kang, S. Shin, Z. Zhang, Enabling security functions with sdn: A feasibility study, Computer Networks, 2015.

[30] A. Khurshid, X. Zou, W. Zhou, et. al., "Veriflow: Verifying network-wide invariants in real time," in the 10th USENIX NSDI, 2013

[31] S. Kumar, "Smurf-based Distributed Denial of Service (DDoS) Attack Amplification in Internet," Second International Conference on Internet Monitoring and Protection (ICIMP 2007), San Jose, CA, 2007.

[32] SDX Central. What Are Microservices in NFV? Definition. https://www.sdxcentral.com/nfv/definitions/microservices-in-nfv-definition/

[33] James Lewis and Martin Fowler. Microservices: A definition of this new architectural term. Retrieved March 25, 2014 from https://www.martinfowler. com/articles/microservices.html

[34] H. C. Wei, Y. H. Tung and C. M. Yu, "Counteracting UDP flooding attacks in SDN," 2016 IEEE NetSoft, Seoul, 2016.

[35] L. H. Newman, "What we know about friday's massive east coast internet outage," Oct 2016, https://www.wired.com/2016/10/internet-outageddos- dns-dyn , as accessed on 04/05/2017

[36] D. Bisson, "The 5 most significant ddos attacks of 2016," Nov 2016, https://www.tripwire.com/state-of-security/securitydata-protection/cyber-security/5-significant-ddos-attacks-2016, as accessed on 04/05/2017

[37] L. Breiman, "Random forests," Machine Learning, 45(1), pp 5-32, 2001.

[38] Dddd Joao Martins, Mohamed Ahmed, et. al., "ClickOS and the art of network function virtualization." In Proceedings of the 11th USENIX NSDI, Berkeley, CA USA 2014.

[39] Floodlight, http://www.projectfloodlight.org/floodlight/

[40] Incapsula.com. (2018). [online] Available at: https://www.incapsula.com/ddos/attack-glossary/dns-flood.html

[41] What risks are associated with recursive DNS queries?, Garage [Online]. Available: https://www.godaddy.com/help/what-risks-are-associated-withrecursive-dns-queries-1184.

[42] Incapsula, https://www.incapsula.com/ddos/attack-glossary/syn-flood.html

[43] Wesley M. Eddy, "Defenses Against TCP SYN Flooding Attacks, " in the Internet Protocol Journal, vol. 9, num. 4, December 2006.

[44] Steven M. B., "Defending against sequence number attacks," RFC 6528.

[45] A Weakness in the 4.2 BSD Unix TCP/IP Software.

[46] BoNeSi. https://github.com/Markus-Go/bonesi

[47] Kyoto data set. http://www.takakura.com/kyoto_data

[48] https://www.incapsula.com/ddos/attack-glossary/http-flood.html

[49] Mirai. https://github.com/jgamblin/Mirai-Source-Code

[50] Fatemeh Nargesian, Horst Samulowitz, et. al. "Learning Feature Engineering for Classification," in Proceedings of 26th Internation Joint Conference on Artificial Intelligence (IJCAI), 2017.

[51] Udayan Khurana, Horst Samulowitz, and Deepak Turaga, "Feature Engineering for Predictive Modeling using Reinforcement Learning," in Proc. of 32 AAAI Conference on Artificial Intelligence (AAAI), 2018.

[52] P. Kalaivani and M.S. Vijaya, "Mining Based Detection of Botnet Traffic in Network Flow," in international Journal of Computer Science and Information Technology & Security (IJCSITS), Feb., 2016.

[53] CTU-13. http://mcfp.weebly.com/ctu-malware-capture-botnet-42.html

[54] H. C. Wei, Y. H. Tung and C. M. Yu, "Counteracting UDP flooding attacks in SDN," 2016 IEEE NetSoft, Seoul, 2016.

[55] http://www-inst.cs.berkeley.edu/~cs161/fa16/slides/monitoring1.key.pdf