LETTER

A Collaborative Intrusion Detection System against DDoS for SDN

Xiaofan CHEN[†a)], ***Student Member and*** **Shunzheng YU**[†], ***Nonmember***

SUMMARY DDoS remains a major threat to Software Defined Networks. To keep SDN secure, effective detection techniques for DDoS are indispensable. Most of the newly proposed schemes for detecting such attacks on SDN make the SDN controller act as the IDS or the central server of a collaborative IDS. The controller consequently becomes a target of the attacks and a heavy loaded point of collecting traffic. A collaborative intrusion detection system is proposed in this paper without the need for the controller to play a central role. It is deployed as a modified artificial neural network distributed over the entire substrate of SDN. It disperses its computation power over the network that requires every participating switch to perform like a neuron. The system is robust without individual targets and has a global view on a large-scale distributed attack without aggregating traffic over the network. Emulation results demonstrate its effectiveness.

key words: *collaborative intrusion detection system (CIDS), distributed denial-of-service (DDoS), software defined networks (SDN), artificial neural network (ANN)*

1. Introduction

The existing IDSs and CIDSs with centralized structure suffer from the single-point-failure problem [1]. The centers may become a target of attacks. They may be overloaded in collecting and processing attack samples. One earlier work [2] uses three OpenFlow switches to collect flow-based features and send them to the controller. A SOM (Self-Organizing Maps) based IDS is built on the controller to do the detection job. The controller does not merge the information from different OpenFlow switches. It makes all the switches act as independent single point IDSs. As a result, the controller is less likely to discover the distributed attacks. So it is not a CIDS. Distributed attacks are difficult to detect since the evidence of them is spread across the network. Combining the evidence from different sources, CIDS is more likely to discover the stealthy and dispersed traffic of such attacks [1]. DefenseFlow [3], deployed on the controller, is an IDS which is mostly used as a counter measure against DoS/DDoS. It can redirect the suspicious flow to the DefensePro mitigation devices. In other words, it requires dedicated hardware beyond SDN controller and switches. AVANT-GUARD [4] is mainly designed for detecting and mitigating the TCP based attacks in SDN. In this solution, each switch keeps the information of every TCP session. It tries to block the TCP handshaking attack, like SYN flood, at the switch side. Giotis et al. [5] combine sFlow and OpenFlow in their CIDS. sFlow is a sampling technique. It may undersample [6] some attack packets when the attack flows are stealthy and dispersing. The CIDS proposed by Mousavi [7] extracts information from OpenFlow *packet-in* message and uses entropy of source IP addresses to detect DDoS. When the entropy is larger than a threshold for several successive times, the system knows that DDoS occurs. In DrawBridge [8], the controllers of ISPs provide traffic engineering services. The controllers of users and other ISPs can subscribe to the services to filter DDoS traffic.

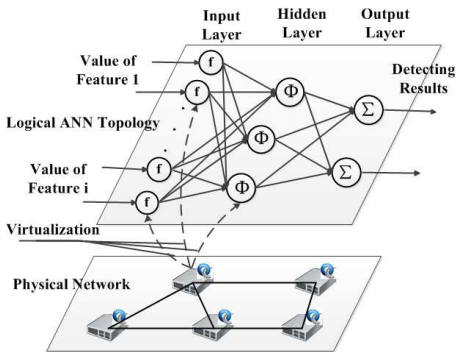

Fig. 1 The architecture of our CIDS for SDN. f is the preprocess function. Φ is the activation function. $\sum$ is the weighted sum up function.

We think that the effective way to deal with single-point-failure problem is to disperse limited computation power to switches without increasing their complexity and costs. As shown in Fig. 1, our CIDS is implemented as an ANN (artificial neural network) overlaid on the network. The switches play the role of neurons. The ANN is designed for monitoring network features and forming a global view on the network status. Based on the global view, the ANN can detect on-line attacks with high accuracy. In other words, the whole network can act as an integrated ANN-based CIDS. The SDN controller is responsible for establishing and maintaining the ANN-based CIDS. It does not participate in the online detection. Due to the property of ANN, the CIDS is capable of dealing with incomplete and distorted data. Therefore, even if some inputs of neuron(s) are delayed, missed or disturbed by noise, the CIDS can still function as usual.

2. System Analysis

When DDoS occurs, some network features may be quite

Manuscript received January 19, 2016.
Manuscript revised May 5, 2016.
Manuscript publicized June 1, 2016.

[†]The authors are with the School of Information Science and Technology, Sun Yat-Sen University, China.
a) E-mail: chxfanz@gmail.com

DOI: 10.1587/transinf.2016EDL8016

different from usual. By globally monitoring those features, our CIDS can capture the abnormal behavior over the network. A set of properly selected features can help the CIDS to capture more kinds of DDoS with less consumption of resources. Table 1 in our previous work [9] shows what types of attacks can be detected by using some corresponding features. In our previous work [9], we have proposed CIPA, a new Collaborative Intrusion Prevention Architecture. It consists of the ensemble of BP (Back Propagation) neural networks. We used a small real world network with simple topology to validate its effectiveness in SDN. Base on CIPA, we make some improvement and focus on DDoS detection for SDN in this paper. We utilize a modified 3-layer RBF (Radial Basis Function) neural network [10], shown in Fig. 1, as the ANN. The reasons are as follows: (a) Compared with the ensemble of ANNs, the detection ability of the modified single ANN is similar, while the complexity and overhead are much lower. (b) RBF net performs better than BP net at pattern recognition, which is more suitable for detecting abnormality [11]. This is the first time that the RBF net is deployed and distributed over an entire SDN. Every physical switch of SDN can be virtualized into one or several neurons. The connections among neurons can be logical connections defined by flow-tables of the physical switches. Therefore, the ANN-based CIDS is actually a virtual network. It can be overlaid over the entire SDN as required. An RBF net consists of an input layer, a hidden layer and an output layer. Neurons at the same layer function in the same way. We assume that the dimension of input layer of a traditional RBF net is n. If there are k features used in the CIDS, the dimension of input layer of the modified RBF net is $k * n$. Therefore, we can use one RBF net, instead of the ensemble of ANNs [9], to process a set of features.

Input neurons, i.e. neurons of the input layer, are in charge of monitoring the flows passing through them, extracting features from the flows, and sending the preprocessed results to the connected neurons of hidden layer. Each input neuron monitors a set of features. It updates their statistics during every sample cycle. At the end of a sample cycle, each neuron gets a vector of feature values in parallel. To reduce the communication overhead and operation complexity of ANN, all feature values are packed into one message and transferred to the hidden layer.

Hidden neurons receive the values from input neurons, operate them with activation functions, and then send the processed results to the output layer. The activation function of a hidden neuron is the radial basis function. In this study the Gaussian function [10] is employed for emulation. The emulation results show that it has good performance.

$$y_j = \Phi(\|x - c_j\|) = \exp(-\frac{\|x - c_j\|^2}{2\delta_j^2}) \quad (1)$$

The Gaussian function has been widely used as a radial basis function, where y_j is the output of the j^{th} hidden neuron, $\|\bullet\|$ the Euclidean distance, $x = [x_1, \ldots, x_m]$ the output vector of input layer, $c_j = [c_{1j}, \ldots, c_{mj}]$ the center of Gaussian function of the j^{th} hidden neuron, and δ_j the width parameter.

Output neurons accept the values from hidden neurons, do the linear mapping operation, and then send out the final results as the detection results of the CIDS. The detection results can tell us whether DDoS occurs now.

$$z_j = \sum_i \omega_{ij} \cdot y_i \quad (2)$$

where z_j is the output of the j^{th} output neuron, y_i the output of the i^{th} hidden neuron, and w_{ij} the weight between the i^{th} hidden neuron and the j^{th} output neuron.

The messages transmitted between neurons require the communication between switches in data plane. We build a neural forwarding table [9] for routing such messages in each switch. Every entry in the table is a key-value pair. The key is the DPID of a destination switch. The corresponding value is one output port of current switch. A neural message [9] can be encapsulated into an Ethernet frame. The type field of the frame is set to a special value to mark that the payload is a neural message. An application can be developed to take charge of the deployment and maintenance of the ANN. The application can be a module of the SDN controller. It can also be deployed on another PC and communicate with the SDN controller through the northbound interface. The application collects the information of available resources, like CPU and memory, of every switch. It will generate a value for the switch according to its resources. It ranks the switches according to their values. Then it chooses some top ranking switches to act as neurons of the ANN. We assume that all or most of the switches should act as input neurons. Then no matter which switches are chosen to be the hidden neurons and output neurons, our system still has a global sight of the whole network. Therefore, different choice of switches will have little or no effect on the detection ability of our system.

3. Emulation Results

Open vSwitch [12] is one of the most widely used OpenFlow switches in research field. POX [13] is also a widely used SDN controller. Mininet [14] is the most famous SDN emulation tool. Mininet allows us to use the real POX controller and the real code from Open vSwitch in emulation. Hence, we utilize Open vSwitch v1.9.3, POX and mininet to build the emulation environment.

We modify the source code of Open vSwitch to realize neural computing and communication. We define neural functions in individual *.c and *.h files. We set a hook in the packet receiving procedure. When receiving a packet, the Open vSwitch checks whether it is a normal packet or a neural message. Then it will process the packet in different ways. For a neural message, after checking the DPID in the message, Open vSwitch sends it to the neural function for further process or forwards it to other switch according to the neural forwarding table. For a normal packet, Open vSwitch makes a copy of it. It sends the copy to its input

neuron for features extraction and statistics. The original normal packet is processed as the usual forwarding procedure. We use BRITE [15] to generate network topologies. Two sizes of networks are used to test the performance of CIDS under different network scales. The sizes of networks are 50 and 100 switches. Each switch connects one host. For each size of networks, we generate 10 different topologies.

Scapy [16] is used to generate the normal traffic and/or attack traffic in each host. Without loss of generality, the ratios of different protocols in normal traffic are: TCP 85%, UDP 10% and ICMP 5% [2]. The ratios during some random time interval are changed so as to make the normal traffic more fluctuant. ICMP flood is used in the emulations. Two attack rates, low and high, are used to test the performance of our CIDS and other schemes under different attack rates. The ratio between the low-rate attack traffic volume aggregated at the victim and the normal traffic volume of each host is about 1 : 5. The ratio for high-rate attack is 1 : 1. In each topology, 30% of the hosts are randomly selected to be the attack sources in every round of emulation. One victim is randomly selected from 10% of the hosts in each round of emulation. The speed of normal traffic observed at each switch is about 6 Mbps. At each 100-switch network, the speed of low-rate DDoS traffic generated at each source is about 6 Mbps/5/30 = 40 Kbps while that of the high-rate one is 200 Kbps. At each 50-switch network, the volume of the aggregated DDoS traffic is not changed, so the speeds at each source are double, i.e. 80 Kbps and 400 Kbps.

The features we selected here are:

- ICMP_ratio: the ratio of ICMP packets to all packets.
- UDP_ratio: the ratio of UDP packets to all packets.
- (SYN_num - ACK_num)_ratio: the ratio of the numeric difference between packets with SYN flag set and packets with ACK flag set to all packets.

All values are monitored or calculated during the current sample cycle. These features are enough for detecting most kinds of network/transport-level DDoS [9], [17]. The number of features is 3 and the network sizes are 50 and 100 switches, respectively. Each switch has 3 input neurons for monitoring the 3 features. So there are 150 and 300 input neurons in the corresponding RBF nets. We use samples from normal traffic and OLS (Orthogonal Least Squares) [18] algorithm to train RBF nets offline. The structures of RBF nets are 150-16-2 and 300-22-2, where x-y-z means there are x input neurons, y hidden neurons and z output neurons. The hidden and output neurons are deployed on some randomly selected switches.

After selecting the topology, victim and attack rate, we can run a round of emulation. There are $10*10*2+10*5*2 = 300$ rounds in all. Each round lasts for $T = 1000s$. There is normal traffic all the time, while there are three waves of attack traffic in every scenario. They last for $t = 50s, 150s$ and $300s$ respectively. In the emulations of this section, the sample cycle of input neurons are set to be $2s$. The small value of sample cycle helps to capture the anomaly more quickly.

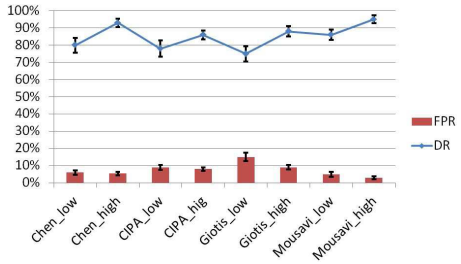

Fig. 2 Emulation results of 50-switch network. Launch ICMP flood with spoofed source IP addresses.

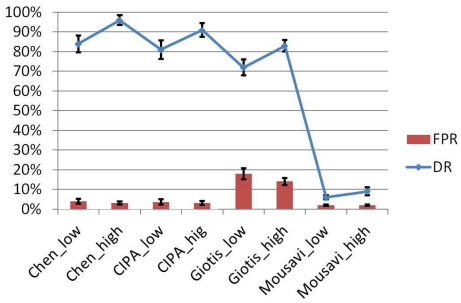

Fig. 3 Emulation results of 100-switch network. Launch ICMP flood with real source IP addresses.

Table 1 Detection results of different attacks.

	ICMP flood	SYN flood	UDP flood	DNS reflection
DR	96.3%	94.8%	95.8%	93.1%
FRP	3.4%	2.3%	2.8%	2.7%

Table 2 Communication overhead under high-rate DDoS.

Network Size	Chen	CIPA	Giotis	Mousavi
50	0.47%	0.88%	2.56%	0
100	0.68%	1.31%	2.64%	0

Then, we compare our ANN-based CIDS with CIPA [9], and two typical CIDS schemes proposed by Giotis et al. [5] and Mousavi [7], that can be found in the literature. The results are presented in Fig. 2, Fig. 3 and Table 2, where DR represents the detection rate, and FPR the false positive rate. The values in Table 2 represent the average ratio of communication traffic rate to normal traffic rate in each switch under high-rate DDoS. All results are the averages.

Some other DDoS attacks last for a very long time. But the detectable features of DDoS may not have significant difference no matter the attack duration is long or short. We also do some emulations which last for a long time. Each round last for $T = 3h$. Each waves of attack lasts for $t = 0.5h$. Other setting are the same. Since it cost too much time to run a round of such emulation, we just do several ones with high-rate DDoS attack. The network size is 100 switches. The attacks are ICMP flood, SYN flood, UDP flood and DNS reflection. The results are showed in Table 1. For SYN flood and UDP flood attack scenarios, 30% randomly selected hosts are attack sources. For DNS reflection attack scenario, 5 randomly selected hosts are DNS servers.

15% randomly selected hosts are attack sources. The results show that our CIDS is effective against different attacks.

Figure 2 shows that the performance of our CIDS is better than CIPA [9] and Giotis's CIDS [5], and comparable with Mousavi's CIDS [7]. However, Mousavi's CIDS is effective only if the source IP addresses of attackers are spoofed. If the attackers launch DDoS with real IPs, it will miss discovering much attack traffic. Recently, many botnets, which launch DDoS, do not rely on IP spoofing to confound source tracking technologies [19]. Figure 3 shows the results of such kind of emulations. The DR of Mousavi's CIDS drops to nearly zero while the DRs of other CIDS schemes are still high. Since a SDN controller can sample only a limited number of packets, the DR of Giotis's CIDS is not as high as the others. When the attack flow becomes more stealthy and dispersing, the DR of Giotis's CIDS drops more. To prove that our CIDS is better than Mousavi's CIDS in terms of detection performance, we conducts emulations with 100-switch network and source IP spoofing ICMP flood. The DR of Mousavi's CIDS is lower than that of our CIDS by about 3%. The aggregation volume of attack traffic does not change. As network becomes larger, the attack traffic is diluted. The ratio of DDoS related *packet-in* message decreases. So the DR of Mousavi's CIDS drops. Since more input neurons are affected by attack traffic, the hidden and output neurons are more likely to find out the abnormity [9]. So the DR of our ANN-based CIDS increases.

Table 2 shows that the communication overhead of our CIDS is better than CIPA and Giotis's CIDS. The data of Mousavi's CIDS is collected from *packet-in* messages, it does not have any extra communication overhead between controller and switches. Besides, we find that the traffic load on the controller of our CIDS and Mousavi's CIDS are almost the same, while that of Giotis's CIDS is much higher. When launching DDoS with spoofed source IP addresses, the traffic load on the controller of our CIDS and Mousavi's CIDS are high since there are more *packet-in* messages. The traffic load on the controller of Giotis's CIDS is about twice as much as that of our CIDS. When launching DDoS with real source IP addresses, the traffic load on the controller of our CIDS and Mousavi's CIDS are a little higher than usual. But traffic load on the controller of Giotis's CIDS is over ten times as much as that of our CIDS since sFlow brings about too much traffic.

All in all, our approach can relieve the heavy traffic on SDN controller while keeping high detection rate, low false positive rate and low communication overhead.

4. Conclusion

This paper proposes an ANN-based CIDS for SDN. The architecture and implementation of the ANN-based CIDS are described in detail in this paper. It disperses limited computational capability to the switches. Each of them acts as a neuron. It requires few resources for computation and communication. With a global view, the ANN-based CIDS is good at detecting large scale DDoS even if the attack volume observed at each node/link is weak compared with the normal traffic. Taking advantage of the virtualization capability and programmability of SDN switches, our ANN-based CIDS is easy to be deployed and adjusted to fit to different detection strategies. It is suitable to be deployed in any type of SDN-based networks, including SDN-based data center, enterprise and campus networks. Compared with other schemes [5], [7], [9], the emulation results with Mininet show that our ANN-based CIDS has good performance and low communication overhead.

Acknowledgments

This work was supported by the Natural Science Foundation of Guangdong Province, China (Grant No. 2014A030313130).

References

[1] C.V. Zhou, C. Leckie, and S. Karunasekera, "A survey of coordinated attacks and collaborative intrusion detection," Elsevier Computers & Security, vol.29, no.1, pp.124–140, 2010.
[2] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," Proc. 35th IEEE LCN, pp.408–415, IEEE, 2010.
[3] Radware, "DDoS Protection as a Native SDN Application by Radware & Big Switch Networks," https://www.radware.com/Solutions/SDN-Resources, 2013.
[4] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks," Proc. ACM CCS, pp.413–424, ACM, 2013.
[5] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments," Elsevier Computer Networks, vol.62, pp.122–136, 2014.
[6] A. Lazarevic, D. Pokrajac, and J. Nikolic, "Applications of neural networks in network intrusion detection," 8th Seminar on NNAEE, pp.59–64, IEEE, 2006.
[7] S.M. Mousavi, Early detection of ddos attacks in software defined networks controller, Master's Thesis, Carleton Univercity, 2014.
[8] J. Li, S. Berg, M. Zhang, P. Reiher, and T. Wei, "Drawbridge: software-defined ddos-resistant traffic engineering," Proc. ACM SIGCOMM, vol.44, no.4, pp.591–592, ACM, 2014.
[9] X.-F. Chen and S.-Z. Yu, "CIPA: a collaborative intrusion prevention architecture for programmable network and SDN," Elsevier Computers & Security, vol.58, pp.1–19, 2016.
[10] S. Haykin, Neural networks: a comprehensive foundation, 3rd ed., pp.230–263, Pearson Prentice Hall, Upper Saddle River, NJ, 2008.
[11] J. Bi, K. Zhang, and X. Cheng, "Intrusion detection based on RBF neural network," Proc. International Symposium on Information Engineering and Electronic Commerce, pp.357–360, IEEE, 2009.
[12] OVS, "Open vSwitch," http://openvswitch.org/, 2015.
[13] MurphyMc, "POX," https://github.com/noxrepo/pox/, 2015.
[14] Stanford, "Mininet," http://mininet.org/, 2015.
[15] Boston, "BRITE: Boston university Representative Internet Topology Generator," http://www.cs.bu.edu/brite/, 2015.
[16] P. Biondi, "Scapy," http://www.secdev.org/projects/scapy/, 2015.
[17] S.T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks," IEEE Comm. Surv. & Tutorials, vol.15, no.4, pp.2046–2069, 2013.
[18] S. Chen, C.F.N. Cowan, and P.M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," IEEE Trans. Neural Networks, vol.2, no.2, pp.302–309, 1991.

[19] S. Lim, J. Ha, H. Kim, Y. Kim, and S. Yang, "A SDN-oriented DDoS blocking scheme for botnet-based attacks," Ubiquitous and Future Networks (ICUFN), 2014 Sixth International Conf on, pp.63–68, IEEE, 2014.