IJE TRANSACTIONS B: Applications Vol. 30, No. 11, (November 2017) 1714-1722

International Journal of Engineering

Journal Homepage: www.ije.ir

Neural Network Based Protection of Software Defined Network Controller against Distributed Denial of Service Attacks

F. Gharvirian, A. Bohlooli*

Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran

PAPER INFO

Paper history:
Received 27 March 2017
Received in revised form 06 September 2017
Accepted 08 September 2017

Keywords:
Software Defined Network
Neural Network
Distributed Denial of Service Attack
Fast Entropy

ABSTRACT

Software Defined Network (SDN) is a new architecture for network management and its main concept is centralizing network management in the network control level that has an overview of the network and determines the forwarding rules for switches and routers (the data level). Although this centralized control is the main advantage of SDN, it is also a single point of failure. If this main control is made unreachable for any reason, the architecture of the network is crashed. A distributed denial of service (DDoS) attack is a threat for the SDN controller which can make it unreachable. In the previous researches in DDoS detection in SDN, not enough work has been done on improvement of accuracy in detection. The proposed solution of this research can detect DDoS attack on SDN controller with a noticeable accuracy and prevents serious damage to the controller. For this purpose, fast entropy of each flow is computed at certain time intervals. Then, by the use of adaptive threshold, the possibility of a DDoS attack is investigated. In order to achieve more accuracy, another method, computing flow initiation rate, is used alongside. After observation of the results of this two methods, according to the described conditions, the existence of an attack is confirmed or rejected, or this decision is made at the next step of the algorithm, with further study of flow statistics of network switches by the perceptron neural network. The evaluation results show that the proposed algorithm has been able to make a significant improvement in detection rate and a reduction in false alarm rate compared to closest previous work, besides maintaining the average detection time on an acceptable level.

***doi**: 10.5829/ije.2017.30.11b.12*

NOMENCLATURE

f_i	Activation functions in MLP	TN	True Negative
b_i	Bias vectors in MLP	TP	True Positive
w_i	Weight matrices in MLP	FN	False Negative
H'	Fast Entropy	FP	False Positive
m	The number of packtes of a flow	**Greek Symbols**	
n	Total number of packets received in a certain time interval	τ	Entropy calibration factor
D_i	Difference between fast entropy value and the average (D_i=| μ_i-H_n |)	μ_i	The average of fast entropy in a time interval
FR	Flow initiation rate	σ	The standard deviation of fast entropy values with the average
t	Duration of a time interval	β	Threshold multiplication factor
DR	Detection Rate	ω	Threshold ($\omega = \beta * \sigma$)
FA	False Alarm Rate		

1. INTRODUCTION

The architecture of software defined network is based on abstraction of network control from data level. The data level consists of simple packet forwarding devices, that do not do any process and contains flow tables populated with localized flow rules [1]. When a new packet comes to a SDN switch, it looks for a match in its flow tables. If no matching rules were found, the packet will be forwarded to the controller [2].

*Corresponding Author's Email: *bohlooli@eng.ui.ac.ir* (A. Bohlooli)

Please cite this article as: F. Gharvirian, A. Bohlooli, Neural Network Based Protection of Software Defined Network Controller against Distributed Denial of Service Attacks, International Journal of Engineering (IJE), TRANSACTIONS B: Applications Vol. 30, No. 11, (November 2017) 1714-1722

Controller, as the operating system of the network compiles network policies into forwarding rules, and installs them at switches through a standard channel e.g., Open Flow [2, 3]. This centralized architecture suffers from a drawback: The controller can become a single-point of failure under attack. An adversary can flood flow requests towards the controller, in order to saturate its processing capacity [3].

Denial of service (DoS) attack, means sending aggressive traffic to the network with the aim of breaking it down, making another user in the network can't receive services from the network services (e.g. servers), making network peripheral (e.g. switch, router) overloads, or at least decreasing the legitimate throughput of the network [4]. Distributed DoS (DDoS) is a type of DoS attack where multiple compromised systems are used to target one or more victims. An attacker may be a real person or a group of zombies which are controlled by an attacker. It can send a large volume of packets to the victim with spoofed source IP addresses [5, 6]. In this research, it is tried to improve the accuracy in detection of this attacks with an acceptable detection time.

In the following, some previous works in DDoS attack detection in SDN are presented. Braga et al. [7] use a Self-Organizing Maps (SOM) for detecting DDoS attacks in SDN using NOX controller. In certain time intervals flow tables of network switches are received and six features of the network flow extracted from the tables are sent to the SOM to determine whether they correspond to a DDoS attack or a normal traffic. Periodic query of all switches, especially in large scale networks is an overhead on the system. A SOM is a competitive neural network useful for reducing the amount of data by clustering [8].

In order to reduce computation time of detecting DDoS attacks, No et al. [9] and David et al. [10] have used fast entropy approach in non-software defined networks. Lim et al. [11] proposed a scheduling-based architecture for the SDN controller protection against DoS attacks. It modifies the controller model so that single request processing queue at the controller is logically subdivided into k queues, each of which corresponds to a flow switch. The controller serves these logical queues with a scheduling discipline.

Mousavi et al. [12] used Entropy for early detection of DDoS attack in SDN using POX controller. In his work, entropy of destination IP address is computed for every 50 packets coming to the controller. If the calculated value is less than a defined threshold continuously for 5 times, an attack will be reported. Although computing entropy variations is a common method of DDoS detection, it has some limitations [13]. When the number of victim hosts increase, this is not a successful method. In addition, when we have a burst legitimate traffic, this method will result in false positive attack detections. it has some limitations. When the number of victim hosts increase, this is not a successful method. In addition, when we have a burst legitimate traffic, this method will result in false positive attack detections.

Kia [14] also has proposed a method for early detection and mitigation of DDoS attacks in SDN using POX controller. In the first step of the solution, Entropy of destination IP address is calculated for every 50 packets coming to the controller. If the computed value is less than a defined threshold, and this is repeated for 5 uninterrupted periods, an attack is suspected and further analysis will be done. However, if entropy method did not detect any attack, the second step, will be performed. Similar to entropy calculation, flow initiation rate for every 50 packets coming to the controller, is computed. If the computed value is above the threshold for 5 uninterrupted times, an attack is suspected and further analysis will be performed in next step. The controller will extract flow statistics of the switches that are suspected in the paths of the attack and confirm or reject the attack. After attack detection, the algorithm tries to mitigate it by reducing value of flow idle timer to prevent the breakdown of switches.

Relying on static thresholds for analyzing flow statistics causes the accuracy to decrease.

The main contributions of our proposed method are as follows:

- In this research, fast entropy is used instead of entropy and therefore, computational time of attack detection is reduced, and since the threshold value is adapted based on traffic condition, the accuracy of detection is improved.
- Using the perceptron neural network for analyzing flow statistics is another good solution for improving the accuracy of the attack detection, which is utilized in this research.
- Querying switches for their flow tables is done only when an attack is suspected and this prevents putting an overhead on the system.
- The effectiveness of the proposed method is demonstrated through different scenarios.

This paper is organized as follows: In section 2, the proposed algorithm for DDoS detection is described. In section 3, the detailed simulation results and analysis is presented. Section 4 covers the conclusion and future works.

2. PROPOSED METHOD

The proposed method consists of five main steps which are presented in Figure 1. This section describes each step in detail.

2. 1. Counting Total Number of Packets In A Flow

First of all, packets received in controller, with identical

source and destination IP/port and of the same protocol, within an identified time interval are counted and classified into flows.

2. 2. Fast Enropy Computation In Determined Time Intervals Entropy is known as a measure of randomness and uncertainty, and a common method of DDoS detection. In this research, in order to reduce computational time of attack detection, while maintaining the accuracy, fast entropy is used instead of entropy. For each flow within a defined time interval, it is computed as Equation (1) [9].

$$H' = -\log\frac{m}{n} + \tau \tag{1}$$

where, m is the number of packets of a flow, n is total number of packets received in a certain time interval. τ is entropy calibration factor, and is added to minimize the false negatives. It is represented as Equation (2):

$$\tau = \begin{cases} \left|\log\frac{n_{i-1}}{n_i}\right| & if\ n_i \geq n_{i-1} \\ \left|\log\frac{n_i}{n_{i-1}}\right| & if\ n_i < n_{i-1} \end{cases} \tag{2}$$

n_i is total number of packets in time interval t_i.

In case of an attack, the number of packets of a flow increases and the entropy drops, but in a normal state the entropy will be in a constant range [10].

2. 3. Updating Adaptive Threshold Defining a suitable threshold plays an important role in attack detection. If it has a small value, the system is very sensitive; this will result in false positive errors when we have legitimate traffic in its peak time. On the other hand, with a threshold with high value, system cannot detect an attack with small changes, so we will have false negative errors.

Adaptive threshold updates according to channel variations.

In each time interval, the fast entropy value is monitored, and the threshold is updated based on following definitions:

- μ_i : The average of fast entropy in a time interval
- σ : The standard deviation of fast entropy values with the average
- D_i :Difference between fast entropy value and the average (D_i=| μ_i- H_n |)
- β : Threshold multiplication factor
- ω : threshold ($\omega = \beta.\sigma$)

After many simulation runs, it was observed that the initial value 3 for β will result in less false positive/ negative errors.

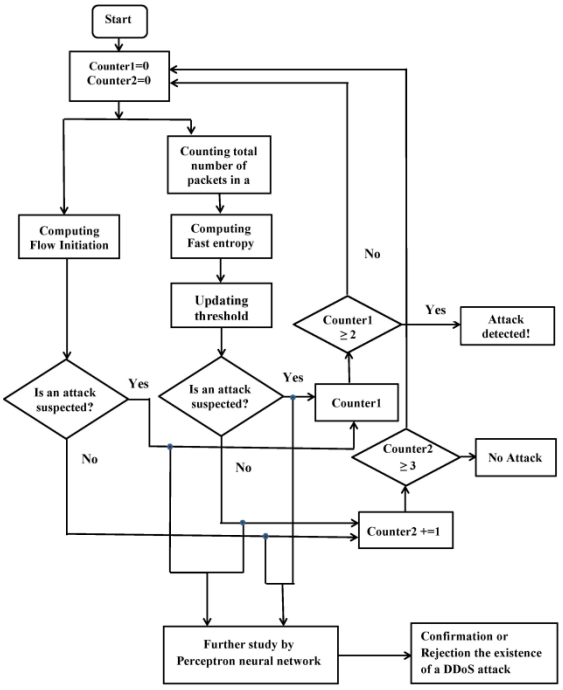


Figure 1. Algorithm flowchart

After computing μ, it is compared with fast entropy values and β will be updated as Equation (3):

$$\beta = \begin{cases} \beta + 1 & H_n > 1.5\mu_i \\ \beta & 0.5\mu_i \leq H_n \leq 1.5\mu_i \\ \beta - 1 & H_n < 0.5\mu_i \end{cases} \tag{3}$$

If the difference between μ_i and H_n (D_i) is above the threshold, there is a possibility of an attack. Figure illustrates this step.

2. 4. Flow Initiation Rate Computation In Determined Time Intervals During a DDoS attack a large volume of packets are sent to a victim, so the flow initiation rate follows a linear growth [7].

In this research, flow initiation rate for each time interval is computed as Equation (4). If it is above a defined threshold, there is a possibility of an attack.

$$FR = \frac{n}{t} \tag{4}$$

where, FR is the flow initiation rate, n is number of packets received in controller within a time interval, t is duration of a time interval.

With running the legitimate traffic for several times, value 60 has been selected for the threshold.

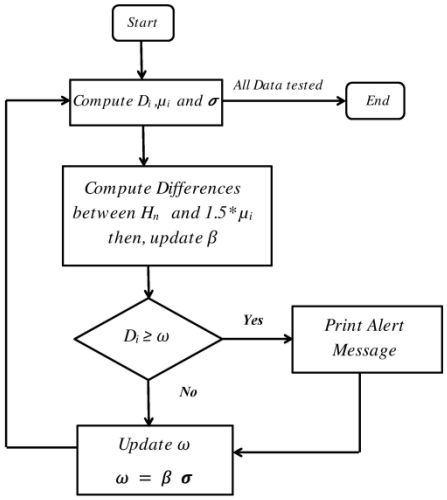


Figure 2. Attack detector with Fast entropy and an adaptive threshold [9]

A sudden rise in legitimate traffic also can cause the flow initiations to increase, so confirming an attack just based on the results of flow initiation rate can result in false positive errors.

2. 5. Examine Flow Statistics by Neural Network to Confirm or Reject Existence of an Attack After multiple simulation runs and observing the accuracy of fast entropy and flow initiation rate functions, four states are considered:

- If both functions report an attack for at least two consecutive time intervals, the attack can be confirmed.
- If both functions reject the existence of an attack, and this result repeated at least for three uninterrupted interval times, it can be declared that there is no attack.
- If one of the two functions report an attack and the other rejected it, flow statistics will be further studied to confirm the attack.

Since query of all switches will put an overhead on the system, in this research it is performed only when an attack is suspected. In this step, a request is sent to network switches to send their flow tables to the controller. Three following per-flow features are extracted from the tables:

- Packet count per flow
- Byte count per flow
- Flow duration

Then, using the perceptron neural network [15-17], existence of a DDoS attack is confirmed or rejected. The parameter values of the applied perceptron are listed in Table 1.

TABLE 1. Parameter values of the NN

Parameter	Value
Number of hidden layers	2
Number of neurons in each layer	5,2
Activation function	f(x)= max(0,x)
Learning rate	0.001
momentum	0.9
alpha	1e-05
beta_1	0.9
beta_2	0.999
epsilon	1e-08
Max_iter	200

3. SIMULATION AND EVALUATION

The algorithm discussed above is implemented on POX, a python-based SDN controller. It is a popular, fast and lightweight controller and is suitable for research and academic field [18].

3. 1. Network Emulator Mininet, the standard emulation tool for SDN, is used for this experiment[2]. The topology used in this research is a tree of depth two with nine switches and 64 hosts. The Open Virtual Switch (OVS) is selected for the network switches. OVS is a virtual multi-layer switch that a lot of protocols and network interfaces is implemented on it[3].

3. 2. Traffic Generation Packet generation is done by Scapy. It is powerful and interactive tool for creating and management of network packets [19]. Scapy can be run in two ways: interactively from a terminal window or by writing programs on python scripts. In this research three python scripts are used for generating legitimate, single victim and multiple victim attack.

3. 3. Training The Perceptron Neural Network Before the main simulation run, in a learning phase about 1 minute, legitimate and attack traffics are run separately and network switches are queried for their flow tables. Three per-flow features mentioned before, are extracted from the tables and are saved for each traffic type. Then, these features are learned for legitimate and attack traffic by Perceptron neural network.

3. 4. Test Case As the main goal of this research is to improve the accuracy of DDoS attack detection, test cases is done the same as one of the recent related

[2] http://mininet.org
[3] http: //openvswitch.org

works [14]. In addition, three more experiments are performed.

Attack traffic used in this research has flows with short duration, small number of packets and no payload. Three legitimate traffics with different rates are considered: In type A, legitimate traffic has noticeable difference with attack traffic. It has flows with longer duration, more packet and byte counts. In type B, legitimate and attack traffic are less different. It has shorter duration flows and less number of packets. Legitimate traffic type C, is a combination of type A and B. Half of the hosts generating legitimate traffic, use type A, and the other hosts use type B. Characteristics of different traffics used in this research are described in Table 2.

The number of hosts generating traffic is fixed at 20. If there is no attackers, the number of legitimate traffic sources is 20-n. Attack Traffic Ratio is defined as Equation (5) [14]:

$$Attack\ Traffic\ Ratio = \frac{Attack\ Traffic\ Rate}{Total\ Traffic\ Rate} \quad (5)$$

For example, in the single victim attack, if one host is sending attack traffic with the rate of 12.5 packets/second, 19 host will be sending legitimate traffic with the rate of 5 packets/second (in case of traffic type A) and this will result in 13% attack traffic ratio. With two, three and four attackers, we have 28, 45 and 63% attack traffic ratio.

In multiple victim attack, if one host sends attack traffic with 33.3 packets/second to four victims and 19 other hosts generate legitimate traffic type A (5 packets/second), we will have 26% attack traffic ratio. With 2 hosts sending attack traffic to 8 victims we have 42.5% and with 3 attacking hosts to 12 victims we have 54% attack traffic ratio. Four hosts attacking 16 victims will result in 62% attack traffic ratio.

TABLE 2. Different types of traffic used in this research

	Type A	**Type B**	**Single-victim attack-Type1**	**Single-victim attack-Type2**	**Multiple-victim attack-Type1**	**Multiple-victim attack-Type2**
Packet type	UDP	UDP	UDP	UDP	UDP	UDP
Packet payload(Byte)	21	21	-	-	-	-
No. of packets to be sent	7	4	1	1	1	1
Traffic Interval	0.2	0.1	0.08	0.01	0.03	0.01
Flow rate(p/s)	5	10	12.5	100	33.3	100

Simulation for each combination of two types of attacks and three types of legitimate traffic is run for 10 times. Each simulation run lasts for 20 minutes and four 3-minutes attacks are performed in this period. In case of single-victim attack, four attacks with one/two/three or four sources to one destination is done. In case of multiple victim attack, one/two/three or four attackers will attack to four/eight/twelve or sixteen victims.

Three more experiments are done in addition to test cases performed in [14]: In pattern D, only legitimate traffic type C is run on the network and for 10 minutes no attack are executed. In pattern E, simulation is similar to previous ones, but legitimate traffic is type C, and attacking hosts execute the attack according to single victim attack-Type 2. In pattern F, simulation is run for 10 minutes and two attacks are performed using multiple victim attack-Type-2: one host attacks four victims and two hosts attack eight victims.

In the proposed algorithm, at each time interval, existence of an attack is confirmed or rejected. When an attack is performed, it is detected with a delay of a few intervals, and is written to a log file with a time stamp. When the attack is finished, it is reported as well.

3. 5. Performance Evaluation In this section, the simulation results of the proposed solution in terms of false positive/ false negative errors and detection rate and false alarm rate of the algorithm are presented. In each simulation, during the execution of an attack, if the attack existence is rejected at a time interval, it is considered as false negative error. In addition, when no attack is running, if an attack is reported at a time interval, it is considered as false positive error.

In order to evaluate the performance of the detection algorithm, detection rate and false alarm rate, those are also used in Ref. [7] and are calculated as Equations (6) and (7) are utilized.

$$DR = \frac{TP}{TP + FN} \quad (6)$$

$$FA = \frac{FP}{FP + TN} \quad (7)$$

where, *DR*= Detection Rate, *FA*= False Alarm Rate, *TP*= True Positive, attack flow classified as attack, *FN*= False Negative, attack traffic classified as legitimate, *FP*= False Positive, legitimate traffic classified as attack, *TN*= True Negative, legitimate traffic classified as legitimate.

Table 3 shows the results of running single victim attack-Type1 alongside three different legitimate traffics. Table 4 shows the results for multiple victim attack-Type1. Results of three more experiments are presented in Table 5.

TABLE 3. Results for single victim attack

	Type A	Type B	Type C
Total number of attacks	40	40	40
	Avg. No. of time intervals with false report		
FP	9.6	10.3	9.7
FN	12.1	10.2	11.4
Detection Rate (%)	91.36	91.84	91.92
False Alarm (%)	2.59	2.46	3.06

TABLE 4. Results for multiple victim attack

	Type A	Type B	Type C
Total number of attacks	40	40	40
	Avg. No. of time intervals with false report		
FP	7.6	10.1	9.7
FN	11	9.4	9.1
Detection Rate (%)	91.92	92.15	93.04
False Alarm (%)	2.19	2.86	3.38

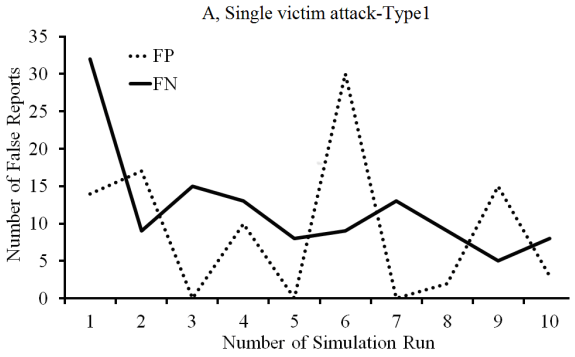
TABLE 5. Results for other patterns

	Pattern D	Pattern E	Pattern F
Characteristics	Without performing any attacks	Legitimate traffic type C + Single-victim attack-Type2	Legitimate traffic type C + Multiple-victim attack-Type2
Total No. of runs	5	5	5
	Avg. No. of time intervals with false report		
FP	0	14	3.6
FN	0	10.8	5.4
Detection Rate (%)	0.00	90.30	95.23
False Alarm (%)	0.00	4.06	2.00

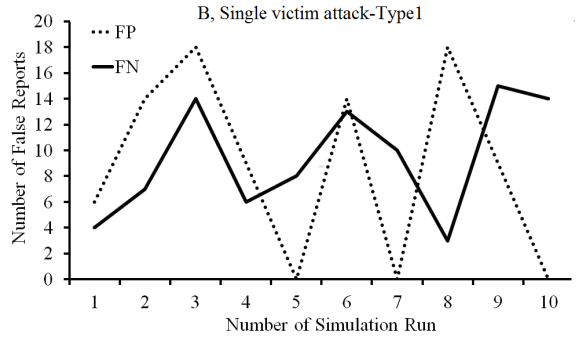
As explained before, the false reports mentioned in above tables are based on the assessment at each time interval. Hence, false negative reports in the results are due to a few time intervals delay of attack detection. It does not mean that the algorithm had been unable to detect the attacks at all. For example, if the false negative is 12.1, it means that within 10 times of simulation runs, it lasted 12.1 time intervals on average to detect the attack. Thus, at this number of intervals no attack was reported. The results indicate that even in conditions that attack detection is expected to be more difficult, for example when the characteristics of the legitimate and attack traffics are close to each other (Type C), or when the attack is distributed among multiple victims, the proposed solution shows a few false reports and has a high detection rate.

Figures 3 to 5 present number of false positive and false negative errors for single-victim and multiple-victim attacks.

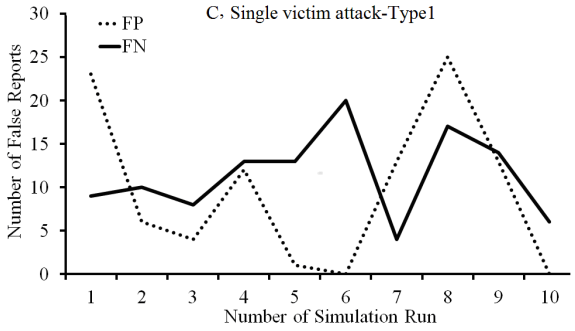
The results show that there is a noticeable improvement in detection rate of the algorithm when legitimate traffic type C is running, for both single victim and multiple victim attacks of type 1 (Tables 6 and 7). Also, in these two cases a significant decrease in false alarm rate is observed, and in type B for multiple-victim it is improved.



a) False reports in traffic Type A, Single victim attack-Type1

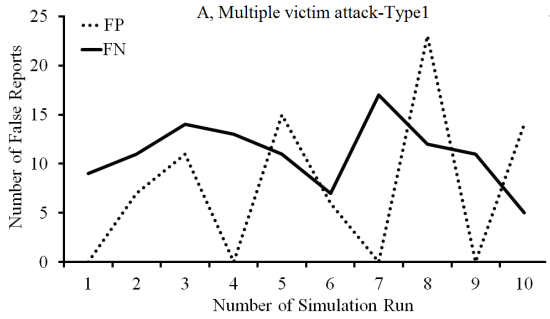


b) False reports in traffic Type B, Single victim attack-Type1

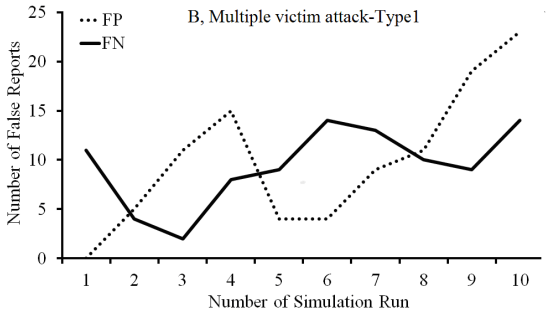


c) False reports in traffic Type C, Single victim attack-Type1

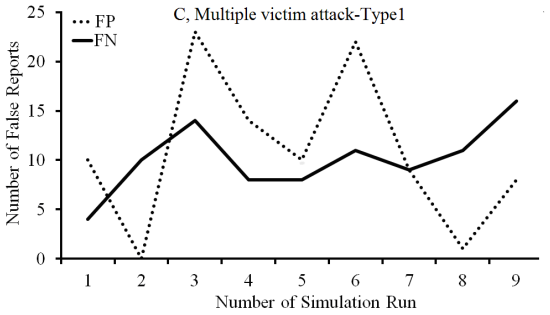
Figure 3. False reports for Single victim attack-Type1



a) False reports in traffic Type A, Multiple victim attack-Type1



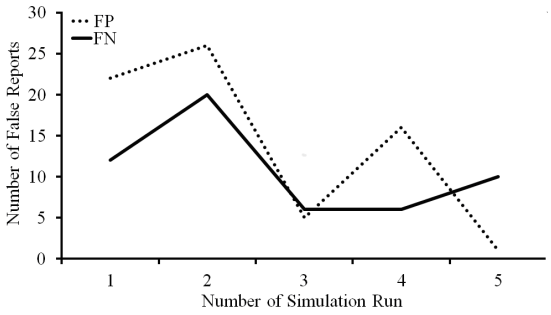
b) False reports in traffic Type B, Multiple victim-Type1



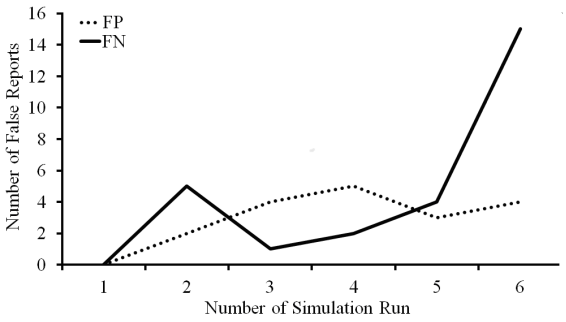
c) False reports in traffic Type C, Multiple victim-Type1

Figure 4. False reports for Multiple victim attack-Type1

It is emphasized that the false negative reports which are effective in computing detection rate, are due to delay in attack detections.



a) False reports in traffic Type C, Single victim attack-Type2



b) False reports in traffic Type C, Multiple victim attack-Type2

Figure 5. False reports for victim attack-Type2

TABLE 6. Performance comparison of the two algorithms for Single victim attack-Type1

	Type A (%)		**Type B** (%)		**Type C** (%)	
	DR	**FA**	**DR**	**FA**	**DR**	**FA**
Algorithm used in this research	91.36	2.59	91.84	2.46	91.92	3.06
Algorithm used in Kia [14]	100	0.0	98.75	1.25	90.00	10.0

TABLE 7. Performance comparison of the two algorithms for multiple victim attack-Type1

	Type A (%)		**Type B** (%)		**Type C** (%)	
	DR	**FA**	**DR**	**FA**	**DR**	**FA**
Algorithm used in this research	91.92	2.19	92.15	2.86	93.04	3.38
Algorithm used in Kia [14]	100	0.00	96.25	3.75	88.75	11.25

If this delay is not considered in the computations, then the detection rate of the proposed algorithm in all cases above will be 100%.

Figure 6 displays the average detection time for different traffic rates of type A for both methods.

It is observed that our proposed solution has been able to maintain the detection time close to the previous work or even decrease it in some cases.

Another parameter considered in the performance evaluation is delay in attack detection. The average detection time of the proposed algorithm for single victim attack is 10.57 seconds and for multiple victim attack is 23.55 seconds. Table 8 is a comparison between the average detection time of the two algorithms for single victim and multiple victim attacks, Type_1, in legitimate traffic type A.

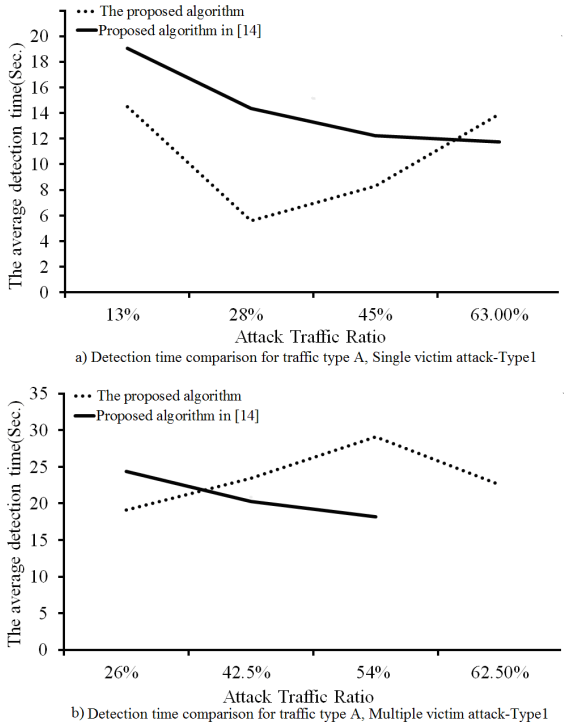


Figure 6. Detection time comparison for traffic type A, Single/Multiple victim attack-Type1

TABLE 8. Detection time comparison of the two algorithms

	Type A, Single-victim attack	Type A, Multiple-victim attack
Algorithm used in this research	10.57^s	23.55^s
Algorithm used in Kia [14]	14.86^s	20.97^s

4. CONCLUSION AND FUTURE WORK

The main goal of this research was to present a method to improve the accuracy in detection of DDoS attacks in software defined networks controller. Also, the delay in detecting the attacks must be small so that there is enough time to mitigate the attack before the controller is made unreachable or slowed down. In this research a particular method for software defined networks and based on their main characteristics, the centralized control unit is presented for DDoS attack detection. In operations like counting packets of a flow and computing fast entropy, flow initiation rate calculation and using flow tables of network switches, this feature has been utilized. The ability of the proposed algorithm in detecting single and multiple victim attacks indicates its high performance. In addition, its high detection rate and low false alarm rate even when the characteristics of legitimate and attack traffics are close to each other, and the short detection time show its high performance. It is observed that for legitimate traffic type A, the algorithm detect single victim attacks with an average delay of 10.57 seconds and false alarm rate for this kind of attack is 2.59%.

For traffic type A and multiple victim attack, the delay is 23.55 seconds and false alarm rate is 2.19%.

In future works, we will investigate an efficient method to mitigate the attack. A possible solution might be tracking the attacker and blocking its traffics.

5. REFERENCES

1. Ali, S.T., Sivaraman, V., Radford, A. and Jha, S., "A survey of securing networks using software defined networking", ***IEEE Transactions on Reliability***, Vol. 64, No. 3, (2015), 1086-1097.
2. Vizváry, M. and Vykopal, J., "Future of ddos attacks mitigation in software defined networks", in IFIP International Conference on Autonomous Infrastructure, Management and Security, Springer., (2014), 123-127.
3. Wang, H.-z., Zhang, P., Xiong, L., Liu, X. and Hu, C.-c., "A secure and high-performance multi-controller architecture for software-defined networking", ***Frontiers of Information Technology & Electronic Engineering***, Vol. 17, No. 7, (2016), 634-646.
4. Oktian, Y.E., Lee, S. and Lee, H., "Mitigating denial of service (dos) attacks in openflow networks", in Information and Communication Technology Convergence (ICTC), 2014 International Conference on, IEEE., (2014), 325-330.
5. Jeyanthi, N., Shabeeb, H., Durai, M.S. and Thandeeswaran, R., "Rescue: Reputation based service for cloud user environment", ***International Journal of Engineering-Transactions B: Applications***, Vol. 27, No. 8, (2014), 1179-1185.
6. Yan, Q. and Yu, F.R., "Distributed denial of service attacks in software-defined networking with cloud computing", ***IEEE Communications Magazine***, Vol. 53, No. 4, (2015), 52-59.
7. Braga, R., Mota, E. and Passito, A., "Lightweight ddos flooding attack detection using nox/openflow", in Local Computer Networks (LCN), 2010 IEEE 35th Conference on, IEEE., (2010), 408-415.
8. Bohlooli, A. and Jamshidi, K., "A gps-free method for vehicle future movement directions prediction using som for vanet", ***Applied Intelligence***, Vol. 36, No. 3, (2012), 685-697.
9. No, G. and Ra, I., "Adaptive ddos detector design using fast entropy computation method", in Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on, IEEE., (2011), 86-93.
10. David, J. and Thomas, C., "Ddos attack detection using fast entropy approach on flow-based network traffic", ***Procedia Computer Science***, Vol. 50, (2015), 30-36.
11. Lim, S., Yang, S., Kim, Y., Yang, S. and Kim, H., "Controller scheduling for continued sdn operation under ddos attacks", ***Electronics Letters***, Vol. 51, No. 16, (2015), 1259-1261.
12. Mousavi, S.M. and St-Hilaire, M., "Early detection of ddos attacks against sdn controllers", in Computing, Networking and Communications (ICNC), International Conference on, IEEE., (2015), 77-81.
13. G., V., N., S.S. and Manikandan MSK., "Navie bayes intrusion classification system for voice over internet protocol network

using honeypot", ***International Journal of Engineering Transaction A: Basics***, Vol. 28, No. 1, (2015), 44-51.

14. Kia, M., "Early detection and mitigation of ddos attackin software defined networks", Ryerson University, Toronto, Ontario, Canada, Ms.c. (2015),
15. Khozani, Z.S., Bonakdari, H. and Zaji, A., "Comparison of three soft computing methods in estimating apparent shear stress in compound channels" ***International Journal of Engineering Transaction C: Aspects***, Vol. 29, No. 9, (2016) , 1219-1226..
16. Pradeep, J., Srinivasan, E. and Himavathi, S., "Neural network based recognition system integrating feature extraction and classification for english handwritten", ***International Journal of Engineering-Transactions B: Applications***, Vol. 25, No. 2, (2012), 99-107.
17. Shamaei, E. and Kaedi, M., "Suspended sediment concentration estimation by stacking the genetic programming and neuro-fuzzy predictions", ***Applied Soft Computing***, Vol. 45, (2016), 187-196.
18. Prete, L.R., Schweitzer, C.M., Shinoda, A.A. and de Oliveira, R.L.S., "Simulation in an sdn network scenario using the pox controller", in Communications and Computing (COLCOM), IEEE Colombian Conference on, IEEE., (2014), 1-6.
19. Bohlooli A., Jamshidi K., "Profile based routing in vehicular ad-hoc networks", ***Science China Information Sciences***, Vol. 57, No. 6, (2014). 1-11.

Neural Network Based Protection of Software Defined Network Controller against Distributed Denial of Service Attacks

F. Gharvirian, A. Bohlooli

Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran

P A P E R I N F O

Paper history:
Received 27 March 2017
Received in revised form 06 September 2017
Accepted 08 September 2017

Keywords:
Software Defined Network
Neural Network
Distributed Denial of Service Attack
Fast Entropy

چکیده

شبکه نرم‌افزار محور یک معماری جدید برای مدیریت شبکه‌ها است که ایده‌ی اصلی آن متمرکزکردن منطق کنترل در سطح کنترل‌کننده‌ی شبکه است که یک دید کلی از شبکه داشته و قوانین ارسال را به تمام سوییچ‌ها و مسیریاب‌های شبکه(سطح داده) صادر می‌کند. این کنترل‌کننده‌ی مرکزی اگرچه یک مزیت بزرگ است، اما اگر به هر دلیلی از دسترس خارج شود، شبکه سطح پردازش خود را از دست داده و معماری آن از بین می‌رود. حملات محروم‌سازی از سرویس توزیع‌یافته می‌توانند کنترل‌کننده‌ی شبکه را از دسترس خارج نمایند. بیشتر کارهای انجام شده در زمینه‌ی تشخیص این حملات در شبکه نرم‌افزارمحور روی تشخیص زودهنگام تمرکز داشته و کار کافی روی بهبود دقت در تشخیص انجام نگرفته است. راه حل پیشنهادی این پژوهش می‌تواند حمله محروم‌سازی از سرویس توزیع شده به کنترلر شبکه‌ی نرم افزار را با دقت قابل توجهی تشخیص دهد و از وارد آمدن آسیب جدی به کنترلر جلوگیری نماید. برای این منظور، آنتروپی سریع برای هر جریان در وقفه‌های زمانی مشخص محاسبه می‌شود. سپس با استفاده از حد آستانه‌ی تطبیق‌پذیر، احتمال یک حمله محروم‌سازی از سرویس توزیع‌یافته بررسی می‌شود. برای دستیابی به دقت بیشتر در کنار این روش، از یک روش دیگر، یعنی محاسبه‌ی نرخ آغاز جریان هم استفاده می‌گردد. پس از مشاهده‌ی نتایج این دو روش، بر اساس شرایطی که در متن توضیح داده خواهد شد، وجود یک حمله تایید یا رد می‌شود و یا اینکه این تصمیم در مرحله‌ی بعدی با بررسی آمارهای جریان سوییچ‌های شبکه توسط شبکه عصبی پرسپترون، انجام می‌گیرد.نتایج ارزیابی نشان می‌دهد که الگوریتم پیشنهادی قادر است یک بهبود مهم در نرخ تشخیص و یک کاهش در نرخ اعلام خطا، نسبت به مرتبط‌ترین کار قبلی ایجاد نماید و علاوه‌براین میانگین زمان تشخیص الگوریتم را نیز در یک سطح قابل قبول نگه دارد.

***doi**: 10.5829/ije.2017.30.11b.12*