The Evaluation of the Two Detection Algorithms for Distributed Denial of Service Attack

Vukosi Rikhotso and Mthulisi Velempini(✉)

University of Limpopo, Private Bag X1106, Sovenga 0727, South Africa
kingvuksy@gmail.com, mvelempini@gmail.com

Abstract. The study evaluates two Distributed Denial of Service (DDoS) attacks detection schemes, the Cloud based and the Netplumber. The schemes are evaluated in terms of CPU and memory utilization. The main objective is to identify the better algorithm with a view of enhancing the schemes. The related work on detection algorithms was reviewed. The schemes are evaluated in a Software defined and Cognitive Radio (SD-CRN) Network environment. An early detection and lightweight detection schemes is desirable.

The desirable algorithm detects the attack within the least number of packets. It also consumes less memory and the least amount of CPU time on average. The study uses a statistical approach with the covariance matrix to evaluate the effect of the attack on the SD-CRN controller. SD-CRN introduces a programmable, dynamic, adaptable, manageable and cost-effective network architecture.

DDoS attacks deplete the network bandwidth or exhausts the victim's resources. Researchers have proposed a number of defence mechanisms (such as attack prevention, trackback, reaction, detection, and characterization) in an endeavour to address the effects of the DDoS attacks. Unfortunately, the incidents of the attacks are on the rise. However, the results of this evaluation show that the Netplumber is the promising algorithm.

Keywords: Detection · Distributed Denial of Service · Performance evaluation

1 Introduction

Technological advances in a number of fields have created a cyber-world in which things are done electronically with speed. The advances are increasing complexity and information overload while drastically reducing decision making durations.

Software Defined Networking (SD-CRN) introduces a programmable, dynamic, adaptable, well-managed and cost-effective network architecture. Network administrators have a global view of the network topology and can manage the behavior of the network through abstraction of higher-level functionality by separating the control plane from the data plane. This emerging architecture is dynamic and is able to deal with dynamic applications. Software Defined Networking (SDN) employs an open standard-based and vendor-neutral approach using the Open Flow protocol for the realization of the SD-CRN architecture. Its fundamental feature is to manage the flow of traffic on the

W. Bao et al. (Eds.): ADHOCNETS 2021/TridentCom 2021, LNICST 428, pp. 63–71, 2022.
https://doi.org/10.1007/978-3-030-98005-4_5

network through defining flow table entries. Network administrators can manage easily an SDN network [1].

SDN attempt to build a computer network by separating it into two systems. The first system is the control plane, which provide performance and fault management, and network flow and management of network topology. Controller can process connection request, link management between devices. The second system is a data plane, where switches may rely on the controller to make decisions or can make decisions on their own. The controller can determine connection a path, if the switch request for one from the controller. The controller could do so by using the control protocol. The controller decides whether the flow could be granted [1].

CRN is an adaptive, intelligent radio and network technology that can detect available channels in a wireless spectrum and change transmission parameters to adapt to the spectrum environment. It also enables a number of communication sessions to run concurrently. It's also an intelligent, reconfigurable and dynamic radio technology [2].

2 Related Work

Several studies have been done with the aim of determining the performance of SDN controllers. The work done in [3], provides a comparative study of the SDN controllers, considered a limited number of controllers such as the NOX-MT, Beacon and Maestro. It focused on the performance of controllers. However, these controllers have since been replaced by other controllers such as the POX, Ryu, Floodlight and OpenDaylight which are the most used controllers in the network environment.

The study conducted provided a set of requirements that are the basis of the comparison between the controllers: TLS Support, virtualization, open source, interfaces, GUI, RESTful API, productivity, documentation, modularity, platform support, age, OpenFlow support, and OpenStack Neutron support. The comparison was done using a Multi Criteria Decision Making (MCDM) method named the Analytic Hierarchy Process (AHP) adapted by a monotonic interpolation/extrapolation mechanism which maps the values of the properties to a value in a pre-defined scale. By using the adapted AHP, five controllers (POX, Ryu, Trema, Floodlight, and OpenDaylight) have been compared, and "Ryu" was observed as to be the best controller based on metrics used. However, the assumed scale is subjective and changing the scale would result in a different conclusion. In this study, we compare four schemes using only one controller [4].

In [4], the malicious nodes are detected through the unique identifier of nodes. The algorithm is effective in detecting malicious nodes however; it does not address the effects of the attacks. In [5], the effects of the distributed denial of service attack are examined and proposed some detection techniques that can be implemented in an effort to prevent the DDoS attack.

3 Simulation Environment

This study employed a statistical approach which was used to collect data using mathematical and statistical equations or computational simulations such as discrete event and Matlab simulators. This study depends on network simulations to collect data.

This approach refers to the systematic empirical investigation of social phenomena using statistical, mathematical, or computational techniques. This approach is useful when a study involves the large scale networks such as WiMAX.

This study uses a cluster sampling to sample the nodes of the network. In cluster sampling, the population is divided into groups usually, geographically. These groups are called clusters. The clusters are randomly selected, and each element in the selected cluster is used. In this study, the population of 150 nodes was divided into four clusters. The four scenarios had the following number of nodes: 30, 50, 80,150, for each scenario 30% to 90% of attacking nodes were considered. The packets were sent to the controller randomly using simple random sampling (SRS).

4 Results and Analysis

4.1 The Effect of DDoS Attack on the Controller

The flooding of the controller by the DDoS is one scenario which can directly affect the controller with a stream of packets. Packets that do not have a match in the flow table were sent to the controller for processing. Most DDoS attacks use spoofed source address, which translates into new incoming packets at the switch. When the number of new incoming packets exhaust the bandwidth of the channel and the processing power controller, the attack overwhelms the controller. The DDoS uses a number of compromised distributed streaming nodes designed to overwhelm the target, the controller.

4.2 Controller Usage Without the DDoS Attack

During this experiments, we monitored the bandwidth usage, the consumption of the CPU time of the controller, the response rate and also the memory utilization of the controller. The study also evaluated the usage of the resources on the controller under normal conditions when there is no attack. To ensure that the results are not distorted, all the processes on the machine were disabled, in order to effectively monitor the usage resources of the controller. The bandwidth on the controller was set to 1000 Mbps, the memory of the controller was set to 1.7 GB, and the CPU was set to 1.80 GHz. The study also evaluated the response rate of the controller. The CPU and bandwidth results are show in Figs. 1 and 2 respectively.

Attacking nodes generate large amount of traffic that are forwarded to the controller. The streams of generated traffic degrade the performance of the controller to the extent that it may not be able to provide service to legitimate nodes. When there were 100 nodes in the network, the traffic rate was observed to be 548.88 Mbps. This is a large traffic that cannot be handled by one controller. Increasing the number of nodes is likely to degrades the controller's response time. It also consumes a lot of CPU time while few nodes require less CPU processing time. It also consumes a lot of memory when the number of nodes is increased. When the number of nodes was set to 80, 36% of the memory was utilized while for 100 nodes, 40% of the memory was used as shown in Fig. 1. If the number of nodes were kept increasing, the controller is likely to crush when 100% of the CPU time is utilized. This may also cause some bottlenecks in the connection.

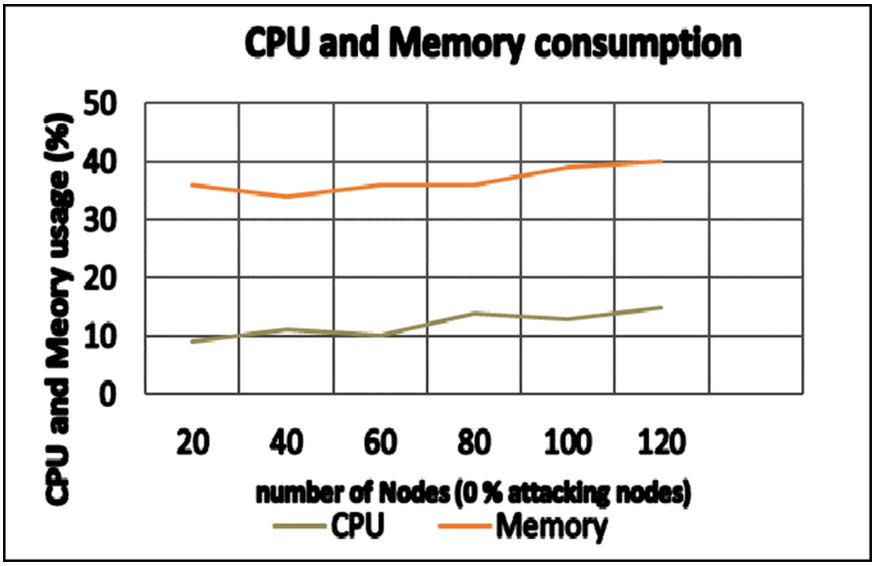

Fig. 1. CPU and Memory usage with 0% of attacking nodes

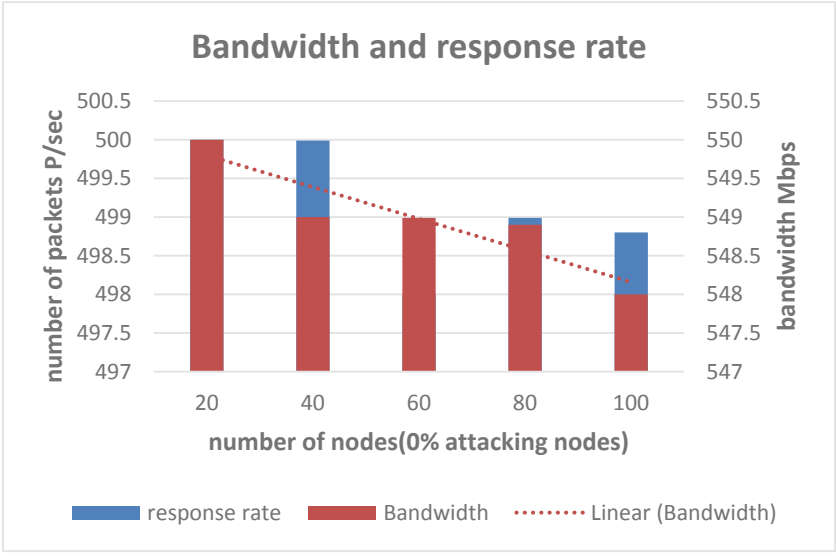

Fig. 2. Bandwidth utilization in Mbps and response rate of the controller in p/sec with 0% of attacking nodes results

4.3 Comparison of DDoS Attack Detection Algorithms

In Fig. 3, we compared the performance of cloud-based and NetPlumber detection algorithms to find out which of the two schemes detects earlier the DDoS attack. Cloud-based scheme detects the DDoS attack within the first 650 packets received by the controller in the network with 20 nodes. The number of received packets decreased for all the detection schemes as the number of nodes in the network increased as shown in Fig. 3.

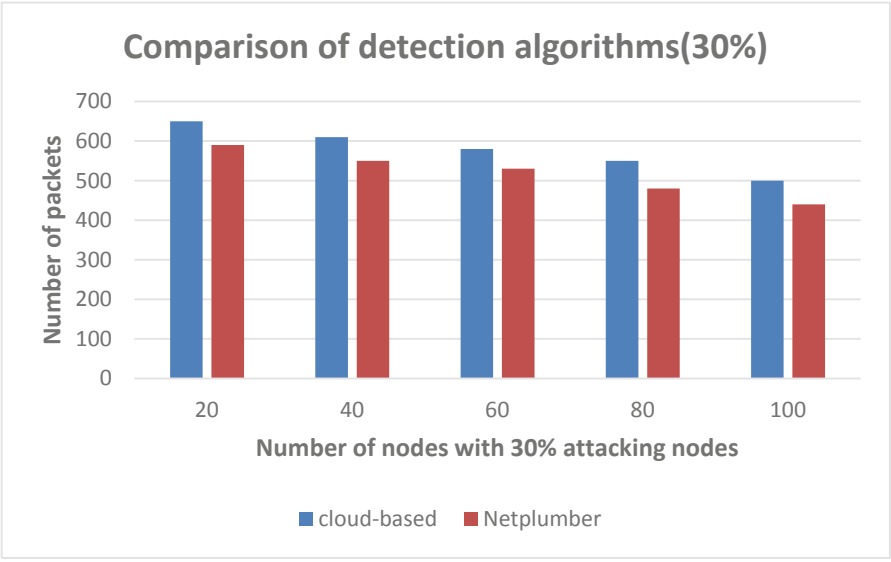

Fig. 3. Comparison of two DDoS attack detection algorithms with 30% of the attacking node

Figure 3 also presents the performance of the algorithms as the number of nodes were increased in the network. This increased the amount of traffic in the network. Suppose that if a specific node can produce 100 packets per millisecond, when the number of nodes in the network is 100. Therefore, if there are 40 malicious nodes in the network, 700 packets per millisecond would be generated. In this case, the NetPlumber detection scheme outperformed the cloud-based algorithm. It detected the DDoS attack within the first 590 packets.

In Fig. 4, we present the results of the scenario populated with 60% of malicious nodes. The results show a change in the detection rate.

The results in Fig. 4 show that NetPlumber and cloud based schemes were still performing better compared to a scenario with 30% attacking nodes. They were able to detect the attack within the least number of packets received compared to when the scenario with 30% attacking nodes. However, the NetPlumber detection scheme performed better than the cloud-based algorithm. The experiment was run three times to verify the results. We observed that as the number of nodes in the network increased, the detection rate improved. The NetPlumber detection algorithm managed to detect the attack within the first 435 received packets in a network scenario with 100 nodes and

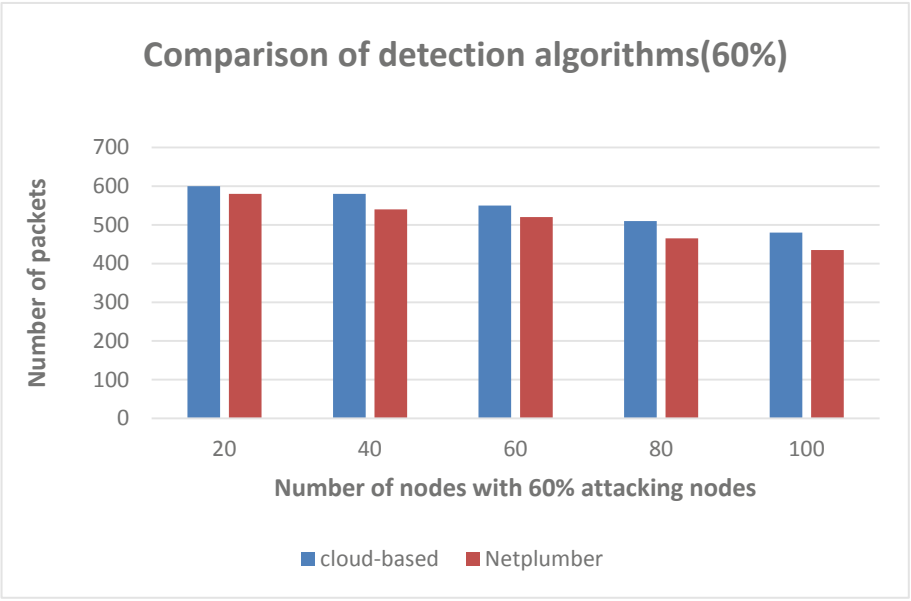

Fig. 4. Comparison of two DDoS attack detection algorithms with 60% of the attacking nodes

60 malicious nodes, which produced 2000 stream of packets per millisecond worth of traffic.

Figure 5 depicts the results of the third scenario. The NetPlumber detection algorithm was observed as the better performing algorithm.

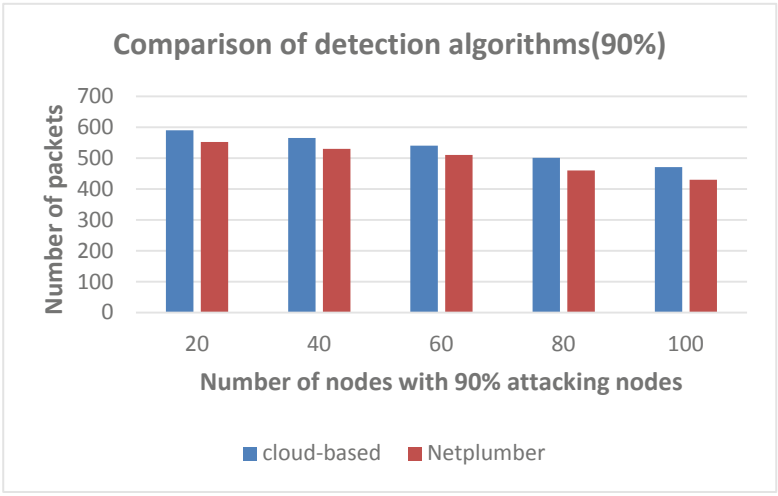

Fig. 5. Comparison of two DDoS attack detection algorithms with 90% of the attacking nodes

A fast detecting scheme that is also lightweight in terms of CPU and memory usage is desirable. The cloud-based at some point performed better than NetPlumber detection algorithm, however it consumed a higher percentage of the CPU and memory of the controller as compared to the NetPlumber algorithm. The NetPlumber consumed less memory and CPU time. However, it takes less time and fewer packets to detect the attack as compared to cloud-based algorithm. Figure 6 depicts the memory utilization results.

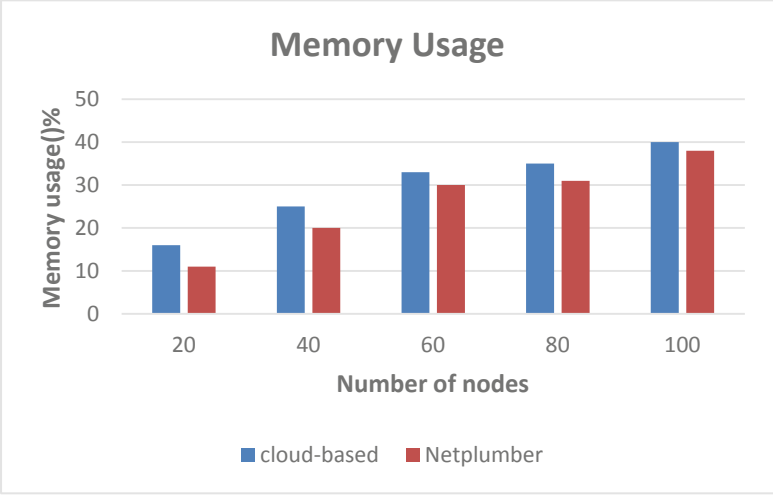

Fig. 6. Memory usage for experiments

The results presented in Fig. 6 shows that NetPlumber detection algorithm is the best performing scheme in terms of the CPU consumption. It consumed 26% of the memory while cloud-based consumed 30% on average. The cloud-based is also poor in the detection of the DoS attack. Figure 7 presents the CPU utilization results.

Figure 7 presents the results of the CPU usage of the two algorithms. The NetPlumber detection algorithm achieved the better performance as compared to the cloud based algorithm. It consumed the least amount of CPU time. It consumed 47% on average the controller's CPU time. Cloud based detection algorithm is the worst performing scheme, which consumed the most processing power of the controller, which is 48.2% on average.

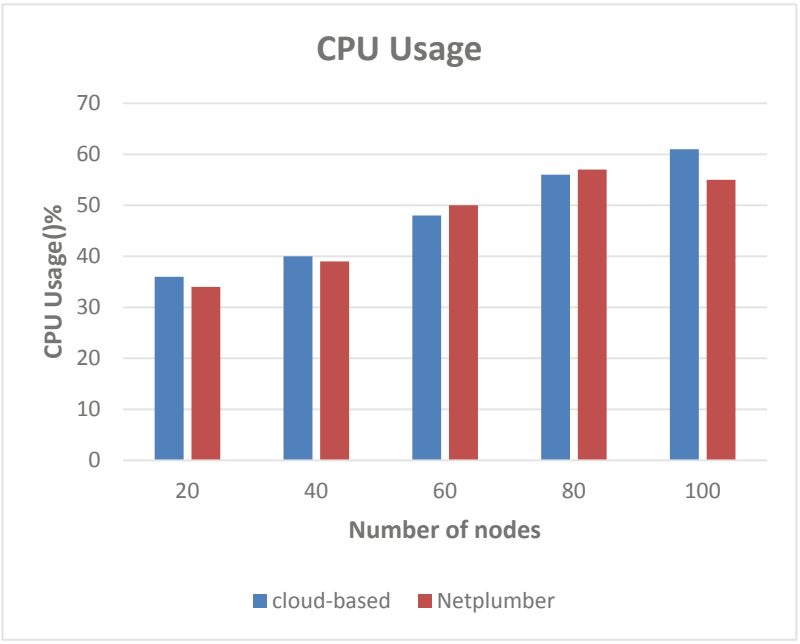

Fig. 7. CPU usage for experiments

5 Conclusion

In our study we investigated two algorithms designed to detect the DDoS attack. The results show that the Netplumber is a lightweight and early detection algorithm which can detect high traffic rate of DDoS attack. We also observed that the controller can withstand the attack for few seconds when the number of malicious packets is less than 300Mbps. Network scenarios with many controllers may be considered in the future.

References

1. Bany Salameh, H., Krunz, M.: Channel access protocols for multihop opportunistic networks: challenges and recent developments. In: IEEE Network-Special Issue on Networking over Multi-hop Cognitive Networks, July 2017
2. Stevenson, C., Chouinard, G., Lei, Z., Wendong, H., Shellhammer, S., Caldwell, W.: IEEE 802.22: the first cognitive radio wireless regional area network standard. IEEE Commun. Magaz. **47**(1), 130–138 (2009). https://doi.org/10.1109/MCOM.2009.4752688
3. Marx, S.E., Luck, J.D., Hoy, R.M., Pitla, S.K., Darr, M.J., Blankenship, E.: Validation of machine CAN bus J1939 fuel rate accuracy using Ne-braska Tractor Test Laboratory fuel rate data. Comput. Electron. Agric. **118**, 179–185 (2015)
4. Bera, S., Misra, S., Vasilakos, A.V.: Software-defined networking for Internet of Things: a survey. IEEE Internet Things J. **4**(6), 1994–2008 (2017)
5. wiki. MATLAB (2017). https://en.wikipedia.org/wiki/MATLAB. Accessed 27 Feb 2017
6. Shin, S., Gu, G.: Attacking Software-Defined Networks: A First Feasibility Study (2013)

7. Feamste, N., Hyojoo, K.: Improving Network Management with Software Defined Networking (2013)
8. Mousavi, S.M., St-Hilaire, M.: Early detection of DDoS attacks against SDN controller. In: International Conference on Computing, Networking and Communications, Communications and Information Security Symposium (2015)
9. Mousavi, S.M., St-Hilaire, M.: Early detection of DDoS attacks against SDN controller. In: International Conference on Computing, Networking and Communications, Communications and Information Security Symposium (2017)
10. Giotis, K., Androulidakis, G., Maglaris, V.: Leveraging SDN for efficient anomaly detection and mitigation on legacy networks. In: Third European Workshop on Software-Defined Network (2014)
11. Kokila, R., Selvi, T., Govindaraja, K.: DDoS detection and analysis in SDN-based environment using support vector machine classifier. In: Sixth International Conference on Advanced Computing (ICoAC) (2014)
12. Khurshid, A., Zou, X., Zhou, W., Caesar, M., Godfrey, P.: VeriFlow: Verifying Network-wide Invariants in Real Time (2013)
13. Kazemian, P., Chang, M., Zeng, H., Varghese, G., McKeown, N., Whyte, S.: Real Time Network Policy Checking Using Header Space (2013)
14. Dhawan, M., Poddar, R., Mahajan, K., Mann, V.: SPHINX: Detecting Security Attacks in Software-Defined Networks (2015)
15. Gkounis, D., Kotronis, V., Dimitropoulos, X.: Towards Defeating the Crossfire Attack using SDN (2013)
16. Krishnan, R., Durrani, M., Phaal, P.: Real-time SDN Analytics for DDoS mitigation (2014)
17. Anderson, C.J., et al.: NetKAT: Semantic foundations for networks. InPOPL (2014)
18. Bailis, P., Andkingsbury, K.: The Network is reliable. Queue **12**(7), 20:20–20:32 (2014)
19. Beckett, R., Zou, X.K., Zhang, S., Malik, S., Rexford, J., Andwalker, D.: An assertion language for debugging SDN applications. In: HotSDN (2014)
20. Radware. DefenseFlow: The SDN Application that Programs Networks for DoS Security. Network Applications in Software Defined Networking (SDN) (2017)
21. Morales, L., Murillo, A., Rueda, S.: Extending the floodlight controller. In: 2015 IEEE 14th International Symposium on Network Computing and Applications (2015)
22. Floodlight. Floodlight Project. http://www.projectfloodlight.org/floodlight/. Accessed 15 July 2016
23. Bhuyan, M., Kashyap, H., Bhattacharyya, D., Kalita, J.: Detecting Distributed Denial of Service Attacks: Methods, Tools and Future Directions (2017)
24. Nhu-Ngoc, D., Junho, P., Minho, P., Sungrae, C.: A Feasible Method to combat against DDoS Attack in SDN Network
25. Aggarwal, A., Gupta, A.: Survey on data mining and IP traceback technique in DDoS attack. Int. J. Eng. Comput. Sci. **4**(6), 12595–12598 (2015). ISSN:2319-7242