Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control
Journal homepage: http://kinetik.umm.ac.id
ISSN: 2503-2267
Vol. 5, No. 3, August 2020, Pp. 187-194

SDN-honeypot integration for DDoS detection scheme using entropy

Irmawati Feren Kilwalaga[1], Fauzi Dwi Setiawan Sumadi*[2], Syaifuddin[3]
Universitas Muhammadiyah Malang, Indonesia[1,2,3]

Article Info
Keywords:
SDN, DDoS, MHN, Entropy, Detection

Article history:
Received 27 March 2020
Revised 15 July 2020
Accepted 18 July 2020
Published 31 August 2020

Cite:
Kilwalaga, I., Sumadi, F., & Syaifuddin, S. (2020). SDN-Honeypot Integration for DDoS Detection Scheme Using Entropy. Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, 5(3). doi:https://doi.org/10.22219/kinetik.v5i3.1058

*Corresponding author.
Fauzi Dwi Setiawan Sumadi
E-mail address:
fauzisumadi@umm.ac.id

Abstract
Limitations on traditional networks contributed to the development of a new paradigm called Software Defined Network (SDN). The separation of control and data plane provides an advantage as well as a security gap on the SDN network because all controls are centralized on the controller so when the compilation of attacks are directed the controller, the controller will be overburdened and eventually dropped. One of the attacks that can be used is the DDoS attack - ICMP Flood. ICMP Flood is an attack intended to overwhelm the target with a large number of ICMP requests. To overcome this problem, this paper proposes detection and mitigation using the Modern Honey Network (MHN) integration in SDN and then makes reactive applications outside the controller using the entropy method. Entropy is a statistical method used to calculate the randomness level of an incoming packet and use header information as a reference for its calculation. In this study, the variables used are the source of IP, the destination of IP and protocol. The results show that detection and mitigation were successfully carried out with an average value of entropy around 10.830. Moreover, CPU usage either in normal packet delivery or attacks showed insignificant impact from the use of entropy. In addition, it can be concluded that the best data collected in 30 seconds in term of the promptness of mitigation flow installation.

1. Introduction

SDN is a network paradigm that separates control planes and data planes to solve problems on traditional networks [1][2]. Control plane is responsible for configuring, including managing traffic flow, while the data plane is responsible for running all the rules defined by the controller [3]. In SDN there are three infrastructure layers, which in the control and data layer are connected using the southbound interface, namely OpenFlow. OpenFlow itself is one of the protocol standards in the SDN network that is used to communicate securely [4]. However, it does not prevent an occurrence of vulnerability. One of the most popular attacks in SDN is namely Distributed Denial of Service (DDoS). DDoS is a distributed attack that is used to attack targets by sending large numbers of packets with the goal of consuming resources and bandwidth so that the target is down [5][6][7][8] with several attack methods such as SYN Flood and ICMP Flood [9]. This can be a problem because in the networking management mechanism on the SDN when the packet arrives, the switch will match the information in the flow table to determine the right action. If no match is found, the switch will send a packet to the controller [10][11] and if a DDoS attack is launched with a target of an attack on the controller then the controller will go down because all controls are centralized on the controller.

In this research, the writer uses Modern Honey Network (MHN) to install a honeypot Suricata sensor which can later be used to trap the attacker and gather information. MHN is open-source software that supports the management and installation of several honeypot sensors easily [12]. Several honeypot sensors in MHN include Dionaea, Suricata, Kippo, Crown and others with MongoDB as sensor data's storage facilities. In addition to MHN, entropy is also used in this study and the combination of MHN-entropy can be a solution to DDoS attacks on SDN networks because entropy is a statistical method used to calculate the uncertainty or randomness of incoming packets [11][13][14].

Several studies have been conducted before to detect and mitigate attacks on the SDN network. In research [15], researchers conducted intrusion detection and analyzed the performance of the detection system using two honeypot sensors, namely Honeyd and Kfsensor. In [16], two methods are combined, namely the honeypot and the NICE model for the detection and prevention of DDoS attacks. Other research [17], proposes the IDS Honeypot method to overcome security threats such as MITM, DoS, DNS Spoofing and ARP Poisoning on wireless networks. Researchers in [10] used the entropy method with the destination IP variable to detect the first 250 attacks on the controller. Whereas in research [11], researchers detected UDP Flood using entropy with the destination IP variable to calculate the randomness level of the first 125 packets. Another study [12], proposed the detection of entropy-based UDP Flooding Attack based on source IP, destination IP, source port, destination port, protocol and generated rate limiting based mitigation. In research [19], researchers proposed a new approach, namely StateSec to increase reactivity and reduce controller load by

delegating local maintenance to the switch. StateSec will utilize entropy-based methods based on source IP, destination IP, source port, destination port variables to detect and mitigate port scans DDoS and flooding of 50000 packets.

Based on previous research, this paper proposes a new mechanism reactively by detecting and mitigating using MHN integration in SDN and then make reactive applications outside the controller using entropy. So, it does not overload the controller performance. The MHN sensor is operated as the security trap for the attacker and can directly analyzed the incoming malicious traffic (ICMP flood) and perform REST communication to the controller to generate OFPT_FLOW_MOD message for installing mitigation rule for the flood attack.

2. Research Method

The experiment was carried out in real environment using 5 PCs that had Ubuntu 18.04 installed and applied a tree topology to the network architecture design consisting of 1 RYU controller [20], 3 Mikrotik switches [21] and 4 hosts with Core i3 4GB RAM specifications. In addition, the protocol used to connect controllers and switches is OpenFlow [4]. The topology is depicted in Figure 1.

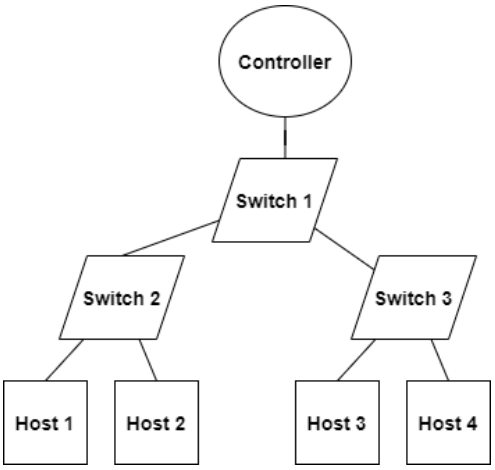

Figure 1. Network's Topology

The experiment scenario was divided into 2 parts, performed by sending 10000 normal packages and ICMP DDoS packages that have been generated using the Scapy tool [22] with each sending rate of 500 and 1000 packets per second (pps). Furthermore, the data was retrieved from the MHN every 5, 15, 30, 50 and 70 seconds. From Figure 1, host 1 acts as an attacker who sent attack packets using the TCPReply tool [23] with random IP and MAC addresses to host 4 which has the Suricata honeypot sensor installed [24].

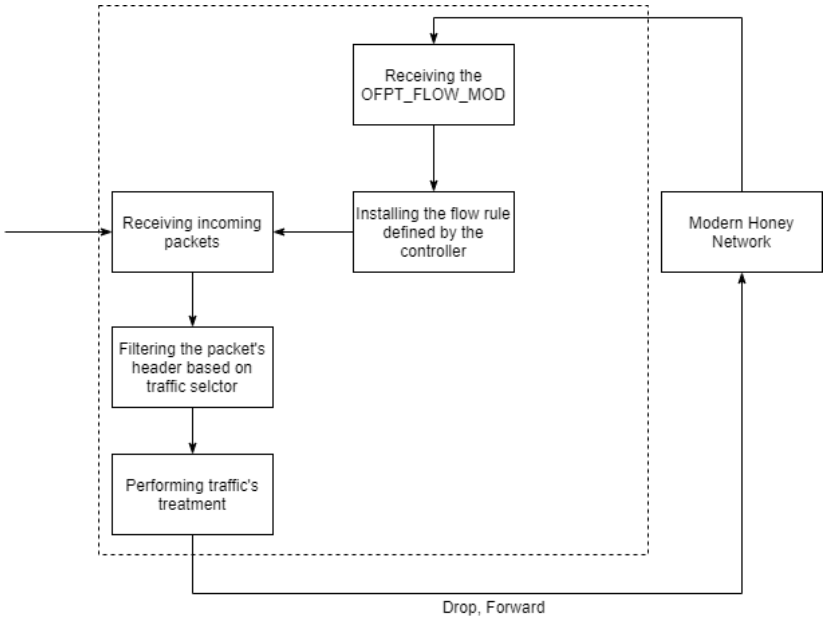

Figure 2. SDN Switches' Block Diagram

The packet was received by the switch and filtered to discover the compatibility of the packet's header with the installed flow rule. If no match was found, the switch would automatically consider the package as a new package. Packets that were considered as new packets would be sent to the controller because there was no mapping of the IP and MAC addresses of these packets. Detection and mitigation of DDoS attacks are shown in Figure 2, Figure 3, and Figure 4.

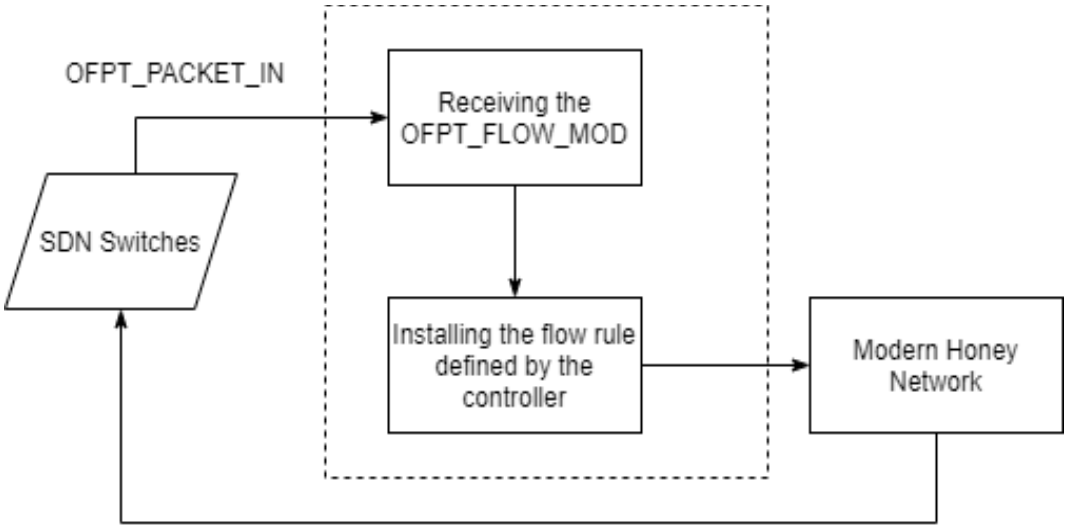

Figure 3. SDN Controller's Block Diagram

In Figure 2, the system flow starts when incoming packets are received by the switch and filtered based on the traffic selector. If a packet with a similar filter already exists, the action is according to traffic treatment. Conversely, if it does not exist then the packet will be sent to the controller in the form of OFPT_PACKET_IN message because it is considered as a new packet. The OFPT_PACKET_IN message will be re-encapsulated and broadcast in the form of OFPT_PACKET_OUT as shown in Figure 3. and the MHN that has installed the Suricata sensor will record and store packets that enter the Mongo database in accordance with the period of data retrieval mentioned in MHN. Then, the package will be exported in CSV format using an entropy application with complete information described in Table 1.

Table 1. Example of Normal and DDoS Data from MHN

Data					Category
Protocol	Source_ip	Destination_ip	Identifier	Honeypot	
ICMP	85.58.15.230	192.168.77.20	d8918b40-64dd-11ea-ac4c-fcaa14e7485f	suricata	DDoS
ICMP	58.47.14.122	192.168.77.20	d8918b40-64dd-11ea-ac4c-fcaa14e7485f	suricata	DDoS
ICMP	9.105.13.238	192.168.77.20	d8918b40-64dd-11ea-ac4c-fcaa14e7485f	suricata	DDoS
ICMP	36.245.6.95	192.168.77.20	d8918b40-64dd-11ea-ac4c-fcaa14e7485f	suricata	DDoS
ICMP	54.215.47.1	192.168.77.20	d8918b40-64dd-11ea-ac4c-fcaa14e7485f	suricata	DDoS
ICMP	192.168.77.15	192.168.77.20	d8918b40-64dd-11ea-ac4c-fcaa14e7485f	suricata	Normal
ICMP	192.168.77.65	192.168.77.20	d8918b40-64dd-11ea-ac4c-fcaa14e7485f	suricata	Normal
ICMP	192.168.77.15	192.168.77.20	d8918b40-64dd-11ea-ac4c-fcaa14e7485f	suricata	Normal
ICMP	192.168.77.65	192.168.77.20	d8918b40-64dd-11ea-ac4c-fcaa14e7485f	suricata	Normal
ICMP	192.168.77.15	192.168.77.20	d8918b40-64dd-11ea-ac4c-fcaa14e7485f	suricata	Normal

After that, the packet header is selected again to take three main variables including IP source, IP destination and protocol to be used as a reference in entropy calculation. This variable was chosen because it has a high level of randomness from the data set taken. The data shown in Table 1 describes the similar value of identifier because the Suricata sensor detects the ICMP flood by the term of 'd8918b40-64dd-11ea-ac4c-fcaa14e7485f'.

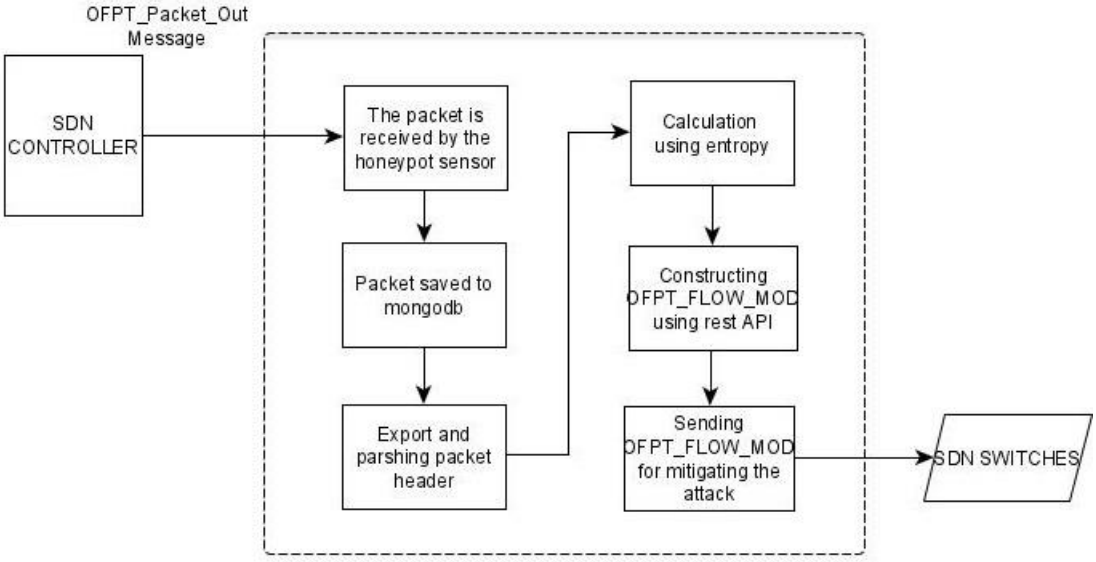

Figure 4. MHN's Block Diagram

Entropy calculation consists of two stages, including the distribution probability calculation of incoming packets using Equation 1 and calculating the entropy value using Equation 2.

$$Pi = \frac{xi}{n} \tag{1}$$

$$H = -\sum_{i=1}^{n} pi \; log_2 \, pi \tag{2}$$

Where $X = \{x_1, x_2, x_3, \dots, x_n\}$ and $Pi = \{p_1, p_2, p_3, \dots, p_n\}$ [10] [25] [26].

The results of entropy will be compared with a predetermined threshold. In this study, the threshold is 1. If entropy < threshold or is in the range of 0-1, the incoming packet is normal [5][11][25][27]. Conversely, if the entropy value is ≥ 1, then the packet is identified as DDoS and host 4 communicates via the application layer using RYU REST Application Programming Interface (API) to create and send OFPT_FLOW_MOD messages to the switch. The OFPT_FLOW_MOD message contains rules for the installation of mitigation-flow instructions with the drop-protocol feature that appears most frequently, and the installation of flow rules is done on all available switches.

Table 2. Flow Rule Components to Block the Attack

Attribute	Values
Dpid	keys (dpid of switch)
cookie	0
table_id	0
Idle_timeout	60
priority	11111
Flags	1
match	{in_port, eth_type, ip_proto}
actions	[] drop protocol

As described in Table 2, the attribute match shows the flow match and actions indicate the flow actions that are used to block or drop the protocol. The protocol drop was chosen because the attacker sent a flooding attack with the same protocol type and by dropping the protocol that appeared most frequently, the mitigation scheme was also immediate. In term of drop-protocol, the idle_timeout with a duration of 60 seconds is used because when no packet is filtered by the flow rule, idle_timeout will run, and the flow will be deleted if the timeout runs out. So, when a normal packet comes in, it can be processed again.

3. Results and Discussion

Based on the experiment conducted 5 times for each scenario with distinct window sizes or the size n, the results obtain the accumulation of entropy values of normal and DDoS flows, CPU usage, and installation time of mitigation flow. The data were extracted after sending 10000 DDoS and 10000 Normal Packet which have different packet component for every data accumulation period and packet sending rate.

3.1 Entropy Results

Based on Table 3, the entropy application for DDoS attacks obtained entropy values > threshold. This means that the flooding process of 10000 DDoS packets to host 4 has been successfully detected for all scenarios with an average entropy value of 10,830. Similar to DDoS attacks, the normal flow of 10000 ICMP packets from 2 normal (Host 2 and Host 3) hosts also show a constant value in the range 0.2 - 0.5 with an average value of entropy around 0.374.

Table 3. Entropy Value

Data Accumulation Period	Packet Rate (pps)	Entropy for Normal Flow	Entropy for DDoS Flow
5 second	500 pps	0.500	8.244
	1000 pps	0.205	7.519
15 second	500 pps	0.333	11.566
	1000 pps	0.400	11.294
30 second	500 pps	0.400	12.379
	1000 pps	0.500	10.971
50 second	500 pps	0.400	8.807
	1000 pps	0.333	12.353
70 second	500 pps	0.333	12.876
	1000 pps	0.333	12.289

3.2 CPU Usage Results

The CPU usage was analyzed by using proc/stat command in Linux based environment every 1 ms period before and after the packet was delivered. The data was retrieved from the Suricata sensor which has Core i3 with 4 GB of RAM. As shown in Figure 5, CPU usage for the normal and DDoS flow with a packet rate of 500pps indicates a fluctuating trend between normal and DDoS. Where the test with a duration of 5 seconds and 50 seconds points to a decrease in CPU usage during a DDoS attack. Meanwhile, the duration of 15, 30 and 70 seconds does not show a small difference in CPU usage. This table also shows that CPU / resource usage is not significant because the average is just under 1%.

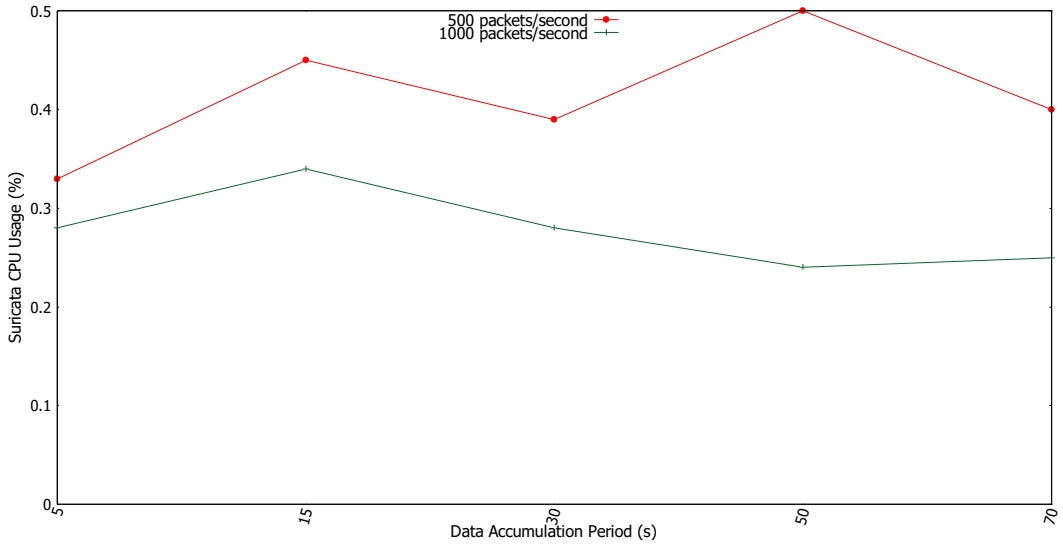

Figure 5. CPU Usage of Suricata during Normal Flow

Furthermore, Figure 6 debscribe CPU usage on a scenario with 1000 pps also shows inconsistent results and there is a decrease when sending DDoS compared to normal delivery. However, compared to the 500pps rate, the CPU

usage value of 1000pps shows a better value and the same as 500pps, CPU / resource usage is not significant because the average is just under 1% which indicates that the use of entropy is very efficient because of small resource utilization.

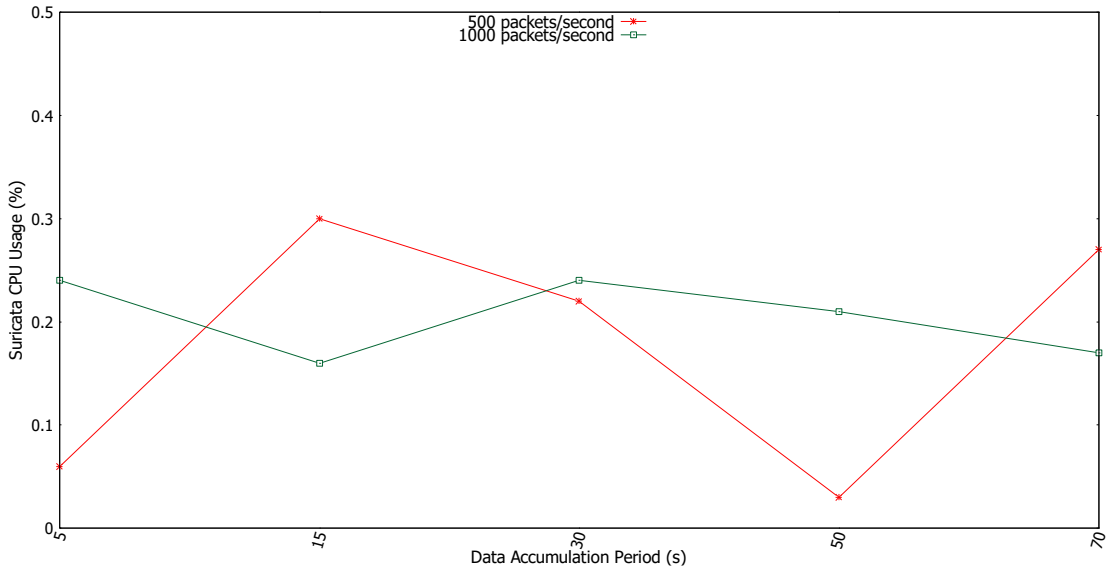

Figure 6. CPU Usage of Suricata during DDoS Flow

3.3 The Promptness of Flow Rule Mitigation

Based on Table 4, the entropy application successfully mitigates attacks with the fastest time duration of 7646ms in a 30-second period. This shows that the protocols of the most frequently occurring packets have been blocked by the mitigation flow rules that have been installed on the switch. It can also be concluded that the data accumulation 30s time is the most effective time for extracting the attack data.

Table 4. The Time Required to Install Mitigation Flow

Data Accumulation Period	Flow Rule Installation Time (ms)
5 second	8779
15 second	34264
30 second	7646
50 second	136904
70 second	37008

4. Conclusion

Based on the research that has been done, it can be concluded that the entropy application has succeeded in detecting and mitigating ICMP Flood DDoS attacks with an average value of entropy about 10.830 for the whole test. Not only that, CPU usage for normal testing and DDoS was also successfully obtained. The results show that the use of entropy has no significant impact on the use of CPU resources. In addition, from all tests, the best data accumulation time is in the span of 30 seconds. In the future, the authors will extend the method for detecting DDoS by comparing the implementation of supervised and unsupervised machine learning for classifying DDoS attack in MHN-SDN environment.

Notation

n : the number of packets
x_i : frequency of i-th packets
P_i : probability of i-th packets

Acknowledgement

The authors would like to express profound gratitude for Informatics Laboratory in Universitas Muhammadiyah Malang which provides a direct support for this research.

References

[1] Collaguazo Jaramillo, A., Alcivar, R., Pesantez, J., & Ponguillo, R. (2019). Cost Effective test-bed for Comparison of SDN Network and Traditional Network. 2018 IEEE 37th International Performance Computing and Communications Conference, IPCCC 2018, 1–2. https://doi.org/10.1109/PCCC.2018.8711223

[2] Sumadi, F., & Chandranegara, D. (2018). Controller Based Proxy for Handling NDP in OpenFlow Network. Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, 4(1), 55-62. https://doi.org/10.22219/kinetik.v4i1.720

[3] Deepa, V., Sudar, K. M., & Deepalakshmi, P. (2019). Detection of DDoS Attack on SDN Control plane using Hybrid Machine Learning Techniques. 2018 International Conference on Smart Systems and Inventive Technology (ICSSIT), Icssit, 299–303. https://doi.org/10.1109/ICSSIT.2018.8748836

[4] Thomas, R. M., & James, D. (2018). DDOS detection and denial using third party application in SDN. *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing, ICECDS 2017*, 3892–3897. https://doi.org/10.1109/ICECDS.2017.8390193

[5] Yan, R., Xu, G., & Qin, X. J. (2017). Detect and identify DDoS attacks from flash crowd based on self-similarity and Renyi entropy. *Proceedings - 2017 Chinese Automation Congress, CAC 2017*, *2017-January*, 7188–7194. https://doi.org/10.1109/CAC.2017.8244075

[6] Koay, A., Chen A., Welch I., &nK.G. Seah W. (2018). A New Multi Classifier System using Entropy-based Features in DDoS Attack Detection. (n.d.). http://10.1109/ICOIN.2018.8343104

[7] Daneshgadeh, S., Ahmed, T., Kemmerich, T., & Baykal, N. (2019). Detection of DDoS Attacks and Flash Events Using Shannon Entropy, KOAD and Mahalanobis Distance. *Proceedings of the 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops, ICIN 2019*, 222–229. https://doi.org/10.1109/ICIN.2019.8685891

[8] Zhang, N., Jaafar, F., & Malik, Y. (2019). Low-Rate DoS Attack Detection Using PSD Based Entropy and Machine Learning. *Proceedings - 6th IEEE International Conference on Cyber Security and Cloud Computing, CSCloud 2019 and 5th IEEE International Conference on Edge Computing and Scalable Cloud, EdgeCom 2019*, 59–62. https://doi.org/10.1109/CSCloud/EdgeCom.2019.00020

[9] Dharma, N. I. G., Muthohar, M. F., Prayuda, J. D. A., Priagung, K., & Choi, D. (2015). Time-based DDoS detection and mitigation for SDN controller. *17th Asia-Pacific Network Operations and Management Symposium: Managing a Very Connected World, APNOMS 2015*, 550–553. https://doi.org/10.1109/APNOMS.2015.7275389

[10] Mousavi, S. M., & St-Hilaire, M. (2015). Early detection of DDoS attacks against SDN controllers. *2015 International Conference on Computing, Networking and Communications, ICNC 2015*, 77–81. https://doi.org/10.1109/ICCNC.2015.7069319

[11] Dave, M. (2019). Defending DDoS against Software Defined Networks using Entropy. *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, 1–5. https://doi.org/10.1109/IoT-SIU.2019.8777688

[12] Wafi, H., Fiade, A., Hakiem, N., & Bahaweres, R. B. (2017). Implementation of a modern security systems honeypot Honey Network on wireless networks. *Proceedings - 2017 International Young Engineers Forum, YEF-ECE 2017*, *November*, 91–96. https://doi.org/10.1109/YEF-ECE.2017.7935647

[13] Divyasree I R., Selvamani K. (2018). Detection of High-Rate Distributed Denial of Service Attack using Entropy Metrics in Cloud Computing Environment. (n.d.). 53–59. https://csce.ucmss.com/cr/books/2018/LFS/CSREA2018/GCC4077.pdf

[14] Sahoo, K. S. (2017). Detection of Control Layer DDoS Attack using Entropy metrics in SDN : An Empirical Investigation. *2017 Ninth International Conference on Advanced Computing (ICoAC)*, 281–286. https://doi.org/10.1109/ICoAC.2017.8441392

[15] Bhagat, N., & Arora, B. (2018). Intrusion detection using honeypots. *PDGC 2018 - 2018 5th International Conference on Parallel, Distributed and Grid Computing*, 412–417. https://doi.org/10.1109/PDGC.2018.8745761

[16] Pandire, P. A., & Gaikwad, V. B. (2018). Attack Detection in Cloud Virtual Environment and Prevention Using Honeypot. *Proceedings of the International Conference on Inventive Research in Computing Applications, ICIRCA 2018*, *Icirca*, 515–520. https://doi.org/10.1109/ICIRCA.2018.8597359

[17] Agrawal, N., & Tapaswi, S. (2017). The Performance Analysis of Honeypot Based Intrusion Detection System for Wireless Network. *International Journal of Wireless Information Networks*, *24*(1), 14–26. https://doi.org/10.1007/s10776-016-0330-3

[18] Ahalawat, A., Dash, S. S., Panda, A., & Babu, K. S. (2019). Entropy Based DDoS Detection and Mitigation in OpenFlow Enabled SDN. *Proceedings - International Conference on Vision Towards Emerging Trends in Communication and Networking, ViTECoN 2019*, 1–5. https://doi.org/10.1109/ViTECoN.2019.8899721

[19] Rebecchi, F., Boite, J., Nardin, P. A., Bouet, M., & Conan, V. (2019). DDoS protection with stateful software-defined networking. *International Journal of Network Management*, *29*(1), 1–19. https://doi.org/10.1002/nem.2042

[20] https://osrg.github.io/ryu/

[21] https://mikrotik.com/

[22] https://scapy.net/

[23] https://tcpreplay.appneta.com/

[24] https://github.com/pwnlandia/mhn/wiki/Suricata-Sensor

[25] Wang, R., Jia, Z., & Ju, L. (2015). An entropy-based distributed DDoS detection mechanism in software-defined networking. *Proceedings - 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015*, *1*, 310–317. https://doi.org/10.1109/Trustcom.2015.389

[26] Oshima, S., Nakashima, T., & Sueyoshi, T. (2010). Early DoS/DDoS detection method using short-term statistics. *CISIS 2010 - The 4th International Conference on Complex, Intelligent and Software Intensive Systems*, 168–173. https://doi.org/10.1109/CISIS.2010.53

[27] Kalkan, K., Altay, L., Gür, G., & Alagöz, F. (2018). JESS: Joint Entropy-Based DDoS Defense Scheme in SDN. *IEEE Journal on Selected Areas in Communications*, *36*(10), 2358–2372. https://doi.org/10.1109/JSAC.2018.2869997