Date of publication SEP-30, 2022, date of current version AUG-08, 2022.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.21.3.2691

Real-time DDoS Detection and Mitigation in Software Defined Networks using Machine Learning Techniques

SANJEETHA R[1], ANITA KANAVALLI[2], ANSHUL GUPTA[3], ASHUTOSH PATTANAIK[4], SASHANK AGARWAL[5]

[1]Dept. of Computer Science & Engineering, M S Ramaiah Institute of Technology, India (affiliated to VTU), (e-mail sanjeetha.r@msrit.edu)
[2]Dept. of Computer Science & Engineering, M S Ramaiah Institute of Technology, India (affiliated to VTU), (e-mail: anithak@msrit.edu)
[3]Verizon Media, Bengaluru, Karnataka, India, (e-mail: anshulg3237@gmail.com)
[4] GE Healthcare, Bengaluru, Karnataka, India, (e-mail: ashutosh.pattanaik28@gmail.com)
[5]Red Hat India Pvt. Ltd., Bengaluru, Karnataka, India, (e-mail: sashank058@gmail.com)

Corresponding author: Sanjeetha R (e-mail: sanjeetha.r@msrit.edu).

ABSTRACT Software Defined Network (SDN) is the new era of networking technology based on a centralized controller that separates the switch hardware from its operating software. The most important challenge is the security of SDN and the most prominent attack is the Distributed Denial of Service (DDoS) attack. Some of the research work done so far detects DDoS attacks using a threshold, which is usually assumed without proper scientific reason and hence may not be always accurate. The mitigation techniques used by some researchers block the host from sending the network traffic beyond a threshold, by installing drop rules in the flow table of the switch connected to that host. Doing so will not only block the attack traffic but also the genuine ones from other applications of that host. In this paper, we propose a model that calculates the threshold limit for the type of applications sending data to a particular switch, in real-time using a machine learning (ML) model, and determines whether that application traffic is DDoS traffic. After the detection, only application type sending DDoS traffic is blocked while other genuine applications are allowed to send the network traffic without any interruption. The use of a dynamic threshold, based on the current network traffic, will help in detecting DDoS efficiently.

KEYWORDS SDN; Threshold; DDoS; Controller; Machine learning.

I. INTRODUCTION

SOFTWARE Defined Network provides new frontiers by offering robust and centralized network architectures which may be deemed ideal to the large networks ubiquitous in today's large-scale data centers. Software Defined Network (SDN) is designed to perform well-defined traffic forwarding decisions. Traditional Networks offer limited flexibility as they are governed by the administrators to adjust the underlying network to different conditions. The emerging data centers differ from old networks in terms of scalability of topology, traffic patterns, and the sheer scale and require a high amount of re-configuration in real-time.

SDN architecture on the other hand de-couples the Data Plane from the Control Plane. The data plane is implemented using switches and the control plane by an SDN controller. The control decision is taken out from the switch and is shifted to the control plane of the network. The controller gets the global view of the entire topology, which helps it to make decisions efficiently. Whenever a switch receives a new packet from a host, it forwards it to the controller which decides the path for it and installs a rule in the flow table of the requesting switch, as well flow tables of switches along the path until the destination. These flow entries are installed dynamically in real-time and are updated regularly. This feature is not available in traditional networks as the forwarding rules are defined once at the time of network configuration.

The management plane is implemented using REST APIs. The communication between the data plane and control plane is done using OpenFlow protocol. OpenFlow protocol is used by switches to send messages to the controller, and by the controller to install flow table rules on the switches. This new architecture has its advantages, but it is prone to attacks of different kinds. Methods should be used to identify such new ways of attack and mitigation techniques should be proposed.

Existing preventive techniques for DDoS attacks in SDN rely on threshold values to detect a DDoS attack. The thresholds are generally static and assumed values and no

longer work when the network traffic changes behavior. Hence there is a need to calculate thresholds dynamically, at runtime, such that DDoS detection works even with changing traffic scenarios. Also, to mitigate the DDoS attack, existing methods completely block the zombie client or sometimes even the entire switch. This impacts legitimate applications running on the client machine or other hosts connected to the network. It is more efficient to block the traffic from attacking application type instead of blocking the client or the switch.

II. RELATED WORK

Murtuza et al. proposed a technique to detect and mitigate a DDoS attack targeted to exhaust the resources at switch and controller by sending repetitive packets to the switch which does not have any flow entry. Since every new flow switch establishes a connection with the controller to determine the forwarding decision, it leads to resource exhaustion when there is a huge amount of new connections. To detect this attack, the authors have used dynamic and adaptive threshold which determines the class boundary for every packet. Packets are classified as safe or risky. The threshold keeps on changing based on traffic behavior. In their proposed solution they are using complete packet parameters such as IP addresses, port, payload, past behavior. By default, the system labels the packets as risky. Over time host gains the trust. For every new connection within a limited time, the connection timeout keeps on reducing as the number of connections increases [1].

Lawal et al. proposed a solution to detect and mitigate DDoS attacks on a targeted system. They make use of the sFlow technique to do so which is a time-based real-time flow monitoring, sampling, and analyzing technique. Out of every N packet encountered at switches, one is selected and sent to the collector. N is determined by Sampling Rate and the interval between two collections is determined by Pooling Interval. From the collector, the packets are sent to the analyzer in case of any suspicion. A threshold is also set which is determined by monitoring the traffic and it is made sure that the number of packets from every switch remains below the threshold. Once this number increases from the threshold, the sFlow triggers an alarm and upon analyzing the traffic, it indicates the controller to install drop rule or allow the flow [2].

Ahalawat et al. made use of entropy to detect the DDoS attack on the network. They used the entropy defined by Shannon which is a matric of unpredictability linked with a random variable, in this case, it is calculated on the inflow, and the random variable consists of the parameters of the packet. A threshold is used and set to the standard deviation of the normalized statistics of various parameters. If for a periodic time interval, the value of standard deviation becomes greater than the threshold then the flow is considered to be an attack. Mitigation of an attack is done by making use of meters. Meters are used to monitor and control the inflow packets at the switch. They trigger one of the defined bands and therefore help in limiting the inflow rate of packets at the controller [3].

Dayal et al. suggested ML technique to detect the DDoS attack. They used a Radial Bias Function (RBF) network along with Particle Swarm Optimization (PSO). They divided the proposed model into three modules: Statistics collector, Attack Detection and Mitigation. Statistics collector collects incoming flow stats from every switch after a certain time interval and analyzes it, if the entropy of the destined IP addresses is found less than a predefined threshold then an alarm is triggered, and the Attack detection module is called. The attack detection module which is the RBF-PSO network extracts the relevant features from the stats collected from switches and tests them against the pre-trained neural network. It then marks a flow as an attack or normal. If marked as an attack, the mitigation module is activated which generates and installs a flow rule on switches to drop the packets destined to the victim's IP [4].

Queiroz et al. used Open Flow protocol to implement SDN architecture in which network devices use multiple counters. Fine-grained monitoring is challenging but helps us in load balancing, capacity planning, network provisioning, and anomaly detection. In this scenario, the counters are updated for every packet crossing the switch and hence must be retrieved in streaming fashion which tells us to use Big Data techniques for the processing of counter values. In this paper, the resources monitored are switch ports, flow tables, and flow entries. It consists of mainly 3 activities –

Data Acquisition – It populates network traffic and information obtained from both the data plane and control plane using the SDN controller.

Data Aggregation – It processes the gathered information and calculates various parameters in real-time.

Data Persistence – It is used to store and provide input to the traffic analysis systems [5].

Mehr et al. implemented a DDoS attack on the Ryu controller and applied the SVM algorithm for detecting them. Firstly, data is collected from PACKET_IN messages and useful information like source IP, destination IP, port numbers are extracted and sent to the SVM model for detection of the attack. A script is written in python to generate traffic from 2 different hosts and another script spoofs the random IP address to generate attack traffic. Parameters such as Standard deviation of flow packets, Speed of flow entries, Speed of source IP, Ratio of pair flow entries, Standard deviation of flow bytes are considered for the SVM model. To mitigate this attack, the flow collector tells the Ryu controller to add new rules which impose the switch to send all malicious packets to the flow collector. The time pattern of DDoS attacks was also considered in this approach [6].

Elsayed et al. discussed several ML techniques. There are two main approaches based on ML that are currently used to detect exploitable attacks on SDN networks: (a) approaches based on simulation, and (b) approaches based on public datasets [7].

The experiment results show that the model has high detection accuracy. The applying of the deep learning model of DDoS attack detection to OpenFlow-based SDN was also described. According to the detection result obtained from the model, the SDN controller will lead to a drop policy and a problem to the switch. The paper described the deep learning DDoS defender's implementation.

The consequences of re-examination. The results of a real-time DDoS attack experiment check that DDOS attacks can be detected and defended effectively by the defender. Real-time DDoS attacks test the security architecture for defensive impact. As can be seen from the final experimental outcomes, if the source address is an attack packet IP address, it is practically observable in the DDoS attack detection scheme

based on deep learning. According to other DDoS attack data traffic function areas, such as we may also obtain the right detection results for the destination IP address, the source MAC address, the source or destination TCP/UDP port number, and other fields [8].

Dennis et al. compared statistical and ML approaches for DDoS detection in Software Defined Network. In this paper an adaptive threshold was generated using a statistical approach and compared with the Random Forest Classifier ML approach on the UCLA dataset. They came up with the conclusion that the ML model is better in the prediction of DDoS as compared to the statistical model in terms of performance and accuracy [9].

Verma et al. compared various ML and deep learning methods to find the best out of them on various datasets namely NSL-KDD, KDDCUP99, CAIDA, DARPA. And they came up with the conclusion that the Random forest classifier gives the best result [10].

III. PROPOSED MODEL

A. DATA COLLECTION AND PREPROCESSING

It is the first and foremost task to be carried out. In this step, the real-time data of the network is sniffed with the help of CICFlowMeter, which gives various network attributes in output. The data consists of 84 attributes like IP, port, packet length, timestamp, protocol, etc.

Collected data is preprocessed by choosing required attributes namely, Src IP, Src Port, Dst IP, Dst Port, Protocol, Timestamp, Total Fwd Pkts, Total Bwd Pkts.

For training we are using a Kaggle Dataset [22]. One additional feature, i.e., threshold is generated based on the collected data and is appended to the data, calculation of which is discussed in a later section.

B. BUILDING A MACHINE LEARNING MODEL

As ML has been used for threshold prediction, building the model is always a crucial part. This paper uses random forest regression as the ML model. Random forests or random decision forests is an ensemble learning method for classification, regression, and other tasks that function by constructing a variety of decision trees at training time and generating the class that is the mode of the individual trees' classes (classification) or means prediction (regression). It consists of 100 decision trees and works as a regressor to predict the value of the threshold. It is also important that a good dataset is being used to train the model.

The ML model is trained on the Kaggle dataset and this model is used for predicting the threshold value for incoming data packets.

C. MITIGATION MODULE

In this step, the calculated threshold values for incoming data packets are compared with the statistical packet count kept by the OpenFlow switches. OpenFlow switches maintain a count of the number of times a flow entry is referenced and send these stats to the controller on a regular basis. These statistics can be made specific to a use case by employing a feature of SDN flow rules, Packet Matching. Packet matching provides many filters known as Match Fields. We are using Port, IP and MAC address of source and IP and port of destination as match fields to track the stats pertaining to a particular type of application.

Comparison of application specific packet count and threshold, which is predicted by ML model, is done at the controller on switch by switch basis. If the packet count is found to be exceeding the threshold along with a decrease in entropy, the data flow is marked as an attack and it is blocked at the switch level by installing a flow rule entry at the switch, thereby dropping all the packets coming from the attacking application.

IV. METHODOLOGY

A. NETWORK TOPOLOGY

Fig. 1 demonstrates the topology used. It is a scenario with simple network structure, consisting of various SDN components, i.e., a controller, switches that are open-flow enabled, and various hosts, connected to at least one open-flow switch, running multiple applications that are making use of different protocols. The controller runs a program with an ML algorithm on the real-time data provided by the switches and identifies the traffic as either attack or normal traffic.

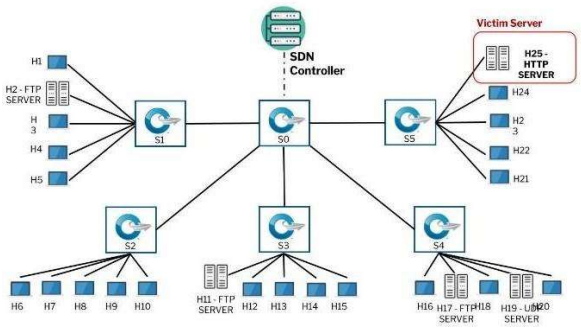


Figure 1. Network Topology

Three types of traffic are being used for the network simulation and testing. These are: HTTP traffic, UDP traffic, and FTP traffic. Traffic is generated in a way as shown in Table 1.

Table 1. Topology Description

Function	Hosts
HTTP server	H25
FTP server	H2, H11, H17, H25
UDP server	H19
HTTP client	H1, H3, H5, H8, H14, H18, H22
FTP client	H3, H6, H8, H13, H23
UDP client	H9, H10
Attacking hosts	H3, H20

Hosts sending HTTP traffic are requesting for a simple index.html page from the HTTP server. UDP traffic is generated by sending random messages in a packet to the UDP server, with the help of packet manipulation tool Scapy [23]. FTP traffic is generated by using python client-server socket programming, and periodically uploading and downloading a file to and from the server.

Two hosts are set to mimic the attack traffic by continuously sending HTTP requests to the HTTP server with spoofed IP addresses and overwhelming it with requests.

B. DATA COLLECTION, PRE-PROCESSING AND MODEL BUILDING

Initially, for training ML models, the data is gathered from a network. In this project, a Kaggle dataset is used. Once the data is gathered, preprocessing must be done. It is known that for a regression model labeled data cannot be considered. Only features with numerical value can be used. The labels must be encoded, if not already in numerical form. IP addresses and protocol are encoded with numerical value. Protocol encoding is a weighted encoding which takes into account the number of packets in a flow, type of packets and payload, its size, etc. The proposed model is using Random Forest as the base ML model. Few important reasons for choosing random forest are:

- Higher accuracy as random forest acquires results from large number of decision trees.
- The random forest has a nature of preventing overfitting when applied to a large dataset.

ML model is trained on the dataset, which along with selected features, has an additional threshold feature, calculation of which is shown in next section. We train and save the model by the method called Pickling, which not only provides us with the benefit of saving and keeping models to be used later, but it also allows us to train the pickled model on new data as it comes. Thus, this makes our model to adapt to network changes real-time and predict correct traffic behavior.

C. CALCULATING THE THRESHOLD

This section highlights the statistical threshold calculation part. Statistically calculated threshold is only used to train ML model. For every packet that is introduced in the network, it is stored in memory with the key as a tuple pair of the source IP and source port. This tuple is then used to dynamically calculate the threshold. The threshold is calculated statistically and stored for further use.

The formula used to calculate the threshold, statistically, is mentioned below. It calculates a running mean and corresponding standard deviation of a window (of length 10 sec) and they are used to calculate the dynamic threshold for the n-th window.

$$\bar{x}_n = \alpha * x_n + (1-\alpha) * \bar{x}_{n-1}$$

$$\sigma_n = \sqrt{\alpha * (x_n - \bar{x}_{n-1})^2 + (1-\alpha) * (\sigma_{n-1})^2}$$

$$th_n = \bar{x}_n + k\sigma_n$$

Here $0 < \alpha < 1$, $k = 1$ and x_n is the packet count of the data traffic taken into consideration for the n-th window. Once the x_n value is known the running mean is calculated. σ_n is the standard deviation for the n-th window. After calculating σ_n, the threshold is calculated using the third equation shown above. The calculated threshold is the statistical value and is appended to the dataset. The ML model is trained on this dataset and is pickled to be used later in the DDoS attack detection and mitigation part.

D. DDOS DETECTION AND MITIGATION

In this section, the detection and mitigation of the attack is explained. The network is monitored for the different types of traffic like HTTP, FTP, UDP. For every incoming new packet, the threshold is predicted by the prediction module and threshold is stored in cache.

In the attack detection module, the packet count from each flow entry of the switch is fetched and is compared with the threshold from the cache. Once the server load has surpassed its limit, the alert is raised and the entropy of the traffic is calculated. Entropy, in this case, is the randomness in the packet header structure and is calculated using Shannon's formula [24]. If the entropy is found to be lower than that of normal traffic, and the packet count is more than the threshold then the traffic is marked as an attack, and a drop rule to drop the packets referring to the flow rule is installed at the switch. By doing this the attack traffic is no longer able to enter the network and thereby mitigating the attack.

Although other types of traffic (normal) are still allowed into the network, only traffic from attacking application is blocked.

E. ALGORITHMS

E.1. PREDICTION MODULE

```
Begin
  Get sniffed data from CICFlowMeter
  Predict the threshold
  Put to Redis
End
```

The prediction algorithm gets the data from CICFlowmeter and predicts the threshold using the ML model and stores the threshold in the cache.

E.2. DETECTION AND MITIGATION MODULE

```
def get_dpid(ip):
    return switch_id
while (True):
Begin:
    Get packet count of each application
    Store in dictionary
    if load_at_server >
server_capacity:
    Do
        Check entropy
            if  (applications  crossed
 threshold):
        Do
            Install the drop rule
        End
     End
End
```

The detection and mitigation module does the traffic analysis for the incoming traffic, retrieves the threshold from the cache, and classifies the traffic based on the threshold.

V. RESULTS

Fig. 2 shows the results of threshold predicted by multi-linear regression ML model versus threshold obtained from statistical methods during preprocessing. It can be seen that in Fig. 1, the values of predictions are almost constant and it cannot cope up with the calculated threshold. Thus, a multi-linear regression model is not a good choice for this problem definition.

Fig. 3, on the other hand, gives much more promising results which are generated by using Random Forest with 100 decision trees. As compared to the multi-linear model, the predictions of this model are much better and thus can be used for the prediction of the threshold. The data preprocessed with a window size of 3 seconds is fed into this model.

Fig. 4 shows the same Random Forest model as in the last case but with the data preprocessed with a window size of 10 sec. It comparatively gives better results as the value of the threshold is increased. Due to the lack of hardware components, we were not able to use this model.

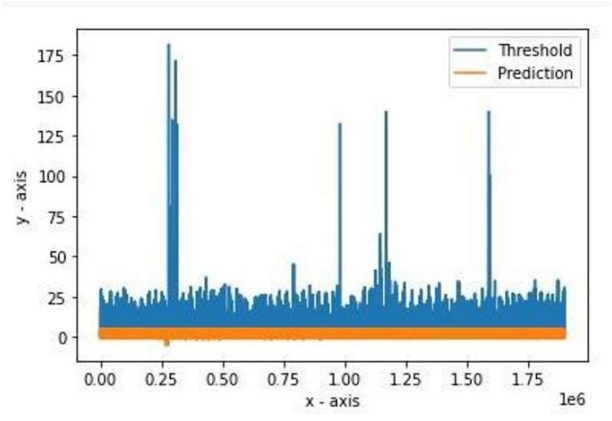


Figure 2. Multi-Linear Regression Model

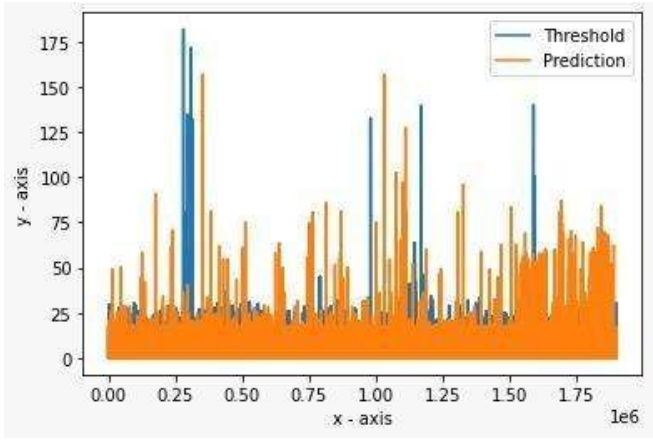


Figure 3. Random Forest Model with window 3 sec

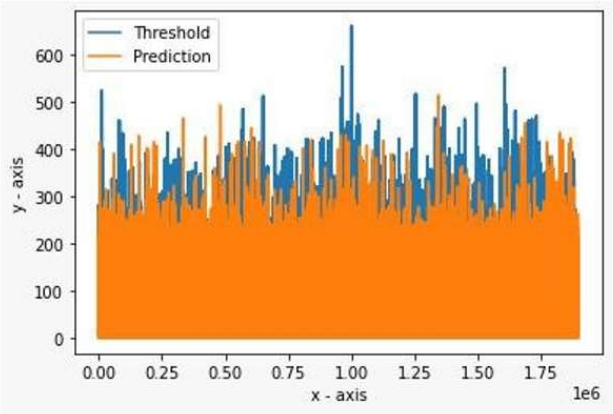


Figure 4. Random forest Regression with window 10 sec

Fig. 5 shows the server load at a particular time. In this scenario, all the hosts are sending benign traffic until 10:03:00. At that moment, one host has started sending Attack traffic, so, the load on the server increases. When this traffic crosses the load handling capacity of the server, the mitigation module will check if any application has crossed the corresponding threshold. When the module finds the attacking application, it installs a drop flow entry in the switch from which the attack traffic is identified to enter the network, thereby, dropping the packets from the malicious application. It can be seen that the load on the server gets back to normal as soon as the packets from the malicious application are dropped and everything else works normally.

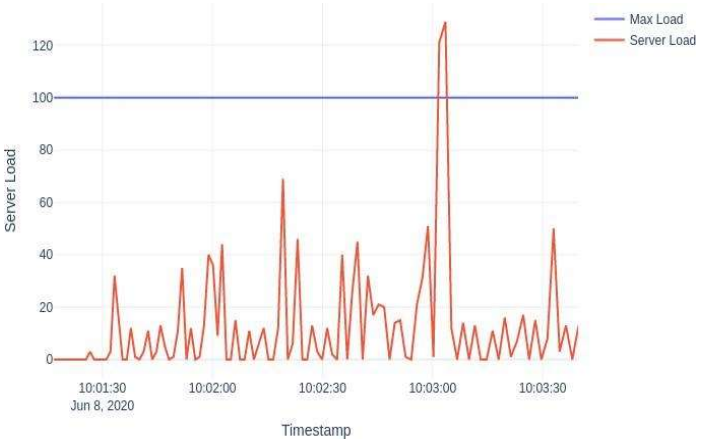


Figure 5. Attack traffic crossing the threshold

Fig. 6 shows traffic on a switch. It consists of two types of traffic, HTTP traffic, and FTP traffic. During the normal traffic, all the packets from the host to the server are allowed but as soon as a host's application instigates the attack, packets from that application type get blocked at the nearest switch and all normal traffic from other applications is allowed to pass.

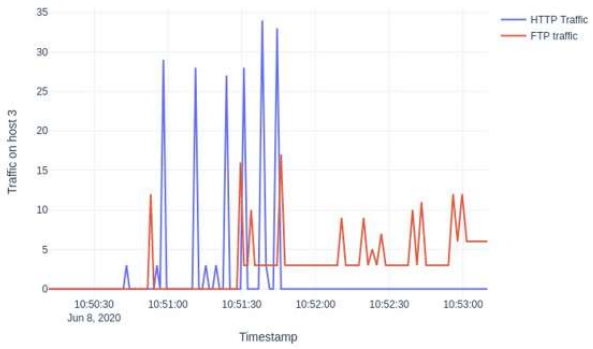


Figure 6. Shows HTTP and FTP flow for host H3

VI. CONCLUSION AND FUTURE SCOPE

In this paper, the data is collected, and efficient ML is built using the Random Forest prediction technique. The model predicts the thresholds dynamically for each application. The system detects DDoS in real-time by analyzing the traffic flow. Mitigation is done successfully by blocking the application used by the zombie client to execute the attack.

In the future, it may be extended to use neural networks instead of using regular ML models. It may provide higher accuracy. The calculations were done for the destination port. Also, it can be experimented with to find a source application port so that attack users may be identified. Other than this we can try to improve the methods of calculating the threshold value. The proposed work uses statistical methods such as standard deviation and weighted mean. Other methods can be used for the calculation of the threshold.

References

[1] S. Murtuza, K. Asawa, "Mitigation and detection of DDoS attacks in software defined networks," *Proceedings of the 2018 Eleventh IEEE International Conference on Contemporary Computing (IC3)*, 2018, pp. 1-3. https://doi.org/10.1109/IC3.2018.8530514.

[2] B. H. Lawal, A. T. Nuray, "Real-time detection and mitigation of distributed denial of service (DDoS) attacks in software defined networking (SDN)," *Proceedings of the 2018 26th IEEE Signal Processing and Communications Applications Conference (SIU)*, 2018, pp. 1-4. https://doi.org/10.1109/SIU.2018.8404674.

[3] A. Ahalawat, S. D. Shashank, A. Panda, K. S. Babu, "Entropy based DDoS detection and mitigation in OpenFlow enabled SDN," *Proceedings of the 2019 IEEE International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, 2019, pp. 1-5. https://doi.org/10.1109/ViTECoN.2019.8899721.

[4] N. Dayal, S. Srivastava, "An RBF-PSO based approach for early detection of DDoS attacks in SDN," *Proceedings of the 2018 10th IEEE International Conference on Communication Systems & Networks (COMSNETS)*, 2018, pp. 17-24. https://doi.org/10.1109/COMSNETS.2018.8328175.

[5] W. Queiroz, M. A. M. Capretz, and M. Dantas, "An approach for SDN traffic monitoring based on big data techniques," *Journal of Network and Computer Applications*, vol. 131, pp. 28-39, 2019. https://doi.org/10.1016/j.jnca.2019.01.016.

[6] S. Y. Mehr, B. Ramamurthy, "An SVM based DDoS attack detection method for RYU SDN controller," *Proceedings of the 15th International Conference on Emerging Networking Experiments and Technologies*, 2019, pp. 72-73. https://doi.org/10.1145/3360468.3368183.

[7] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Machine-learning techniques for detecting attacks in SDN," ArXiv preprint arXiv:1910.00817, 2019. https://doi.org/10.1109/ICCSNT47585.2019.8962519.

[8] C. Li, Y. Wu, X. Yuan, Z. Sun, W. Wang, X. Li, and L. Gong, "Detection and defense of DDoS attack–based on deep learning in OpenFlow-based SDN," *International Journal of Communication Systems*, vol. 31, no. 5, article e3497, 2018. https://doi.org/10.1002/dac.3497.

[9] M. J. R. Dennis, *Machine-learning and Statistical Methods for DDoS Attack Detection and Defense System in Software Defined Networks*, Master Thesis, Toronto, Ontario, Canada, 2018.

[10] P. Verma, S. Tapaswi, and W. W. Godfrey, "An adaptive threshold-based attribute selection to classify requests under DDoS attack in cloud-based systems," *Arabian Journal for Science and Engineering*, vol. 45, no. 4, pp. 2813-2834, 2020. https://doi.org/10.1007/s13369-019-04178-x.

[11] A. M. Sukhov, E. S. Sagatov, and A. V. Baskakov, "Rank distribution for determining the threshold values of network variables and the analysis of DDoS attacks," *Procedia Engineering*, vol. 201, pp. 417-427, 2017. https://doi.org/10.1016/j.proeng.2017.09.666.

[12] P. Verma, S. Tapaswi, and W. W. Godfrey, "An adaptive threshold-based attribute selection to classify requests under DDoS attack in cloud-based systems," *Arabian Journal for Science and Engineering*, vol. 45, no. 4, pp. 2813-2834, 2020. https://doi.org/10.1007/s13369-019-04178-x.

[13] Y. Chen, K. Hwang, and W.-S. Ku, "Collaborative detection of DDoS attacks over multiple network domains," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 12, pp. 1649-1662, 2007. https://doi.org/10.1109/TPDS.2007.1111.

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

[15] M. Sachdeva, K. Kumar, and G. Singh, "A comprehensive approach to discriminate DDoS attacks from flash events," *Journal of Information Security and Applications*, vol. 26, pp. 8-22, 2016. https://doi.org/10.1016/j.jisa.2015.11.001.

[16] D. Kshirsagar, and S. Kumar, "A feature reduction based reflected and exploited DDoS attacks detection system," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1-13, 2021. https://doi.org/10.1007/s12652-021-02907-5.

[17] D.-T. Truong, K.-D. Tran, Q.-B. Nguyen, and D.-T. Tran, "Detection of DoS, DDoS attacks in software-defined networking," In: *Research in Intelligent and Computing in Engineering*, Springer, Singapore, 2021, pp. 25-35. https://doi.org/10.1007/978-981-15-7527-3_3.

[18] R. M. A. Ujjan, Z. Pervez, K. Dahal, W. A. Khan, A. M. Khattak, and B. Hayat, "Entropy based features distribution for Anti-DDoS model in SDN," *Sustainability*, vol. 13, no. 3, pp. 15-22, 2021. https://doi.org/10.3390/su13031522.

[19] S. Saharan, and V. Gupta, "DDoS prevention: Review and issues," *Advances in Machine Learning and Computational Intelligence*, pp. 579-586, 2021. https://doi.org/10.1007/978-981-15-5243-4_53.

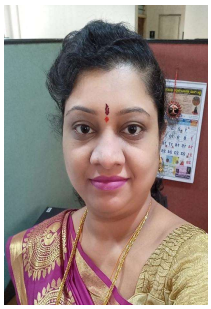
[20] K. F. Xylogiannopoulos, P. Karampelas, and R. Alhajj, "Advanced network data analytics for large-scale DDoS attack detection," In: *Research Anthology on Combating Denial-of-Service Attacks*, IGI Global, pp. 358-370, 2021. https://doi.org/10.4018/978-1-7998-5348-0.ch019.

[21] G. Megala, S. Prabu, and B. C. Liyanapathirana, "Detecting DDoS attack: A machine-learning-based approach," In: *Applications of Artificial Intelligence for Smart Technology*, IGI Global, pp. 55-66, 2021. https://doi.org/10.4018/978-1-7998-3335-2.ch004

[22] Kaggle DDoS Dataset by Devendra. [Online]. Available at: https://www.kaggle.com/devendra416/ddos-datasets/data#

[23] P. Biondi, "Scapy documentation(!)," 2010. [Online]. Available at: https://scapy.net/

[24] M. Idhammad, K. Afdel, M. Belouch, "Detection system of HTTP DDoS attacks in a cloud environment based on information theoretic entropy and random forest," *Security and Communication Networks*, vol. 2018, Article ID 1263123, 13 pages, 2018. https://doi.org/10.1155/2018/1263123.



SANJEETHA R, *a Research Scholar, an Assistant Professor, is working in the Department of Computer Science and Engineering at M S Ramaiah Institute of Technology. Area of interests includes Software Defined Networks, Computer Networks, and Data Communications.*

ANITA KANAVALLI *is currently working as Head of Dept. of CSE, M S Ramaiah Institute of Technology. She has received her Ph.D. in Computer Science and Engineering from Bangalore University in 2013. Her areas of interest include Networks, SDN, Cyber-Physical Systems.*

ANSHUL GUPTA, *an Associate Production Engineer at Verizon Media. He studied Computer Science Engineering at Ramaiah Institute of Technology, Bangalore. Areas of interest include Computer Networks, Software Defined networking.*

ASHUTOSH PATTANAIK, *Build & Release Engineering Specialist at GE Healthcare. He successfully pursued his degree in Computer Science & Engineering from Ramaiah Institute of Technology, Bengaluru in 2020. An open-source code contributor, his areas of innovation include Software Defined Networks, Cybersecurity and DevOps.*

SASHANK AGARWAL, *an Associate Software Engineer at Red Hat. He completed his B.E. in Computer Science and Engineering from Ramaiah Institute of Technology in 2020. His area of interest includes Computer Networks, Software Defined Networks, DevOps.*