A Machine Learning Based DDoS Attack Detection Method In SDN Networks

Anonymous author, *Student, XJTU*

***Abstract*—The concept of Software Defined Networking (SDN) represents a modern way to organize computer network as it decouples the control plane from the data plane through network abstraction. However, countering Distributed Denial-of-Service (DDoS) attacks aimed at controllers has become a major issue in SDNs, as the controller responsible for managing network traffic is a sensitive failure point in the entire network architecture. This article mainly introduces a method for extracting traffic packet features in SDN Networks and utilizing machine learning algorithm for their classification. This technology can be used to identify packets in SDNs that are utilized for conducting DDoS attacks to the network and protect the network from failing. In our testing on a simple SDN Network using KDD-CPU99 dataset, this method demonstrated acceptable performance.**

***Index Terms*—Software Defined Networking, DDoS attack, Machine learning, Network attack protection, SYN Flood.**

I. Introduction

SOSTWARE Defined Networking (SDN) offers a higher level of network automation compared to traditional network architectures. This is primarily due to the decoupling of network functions into control plane and data plane, as well as the centralized management and control of network logical views through dedicated SDN controllers. However, this renders the controller a sensitive failure point within SDN networks. The controller is responsible for managing the operational logic of the entire network, meaning that attacks targeting any part of the network could potentially impact its functioning. Moreover, its malfunction could inflict catastrophic damage to the network.

In this circumstances, malicious cyber attacks, particularly Distributed Denial-of-Service (DDoS) attacks, mainly exert affection on SDN controllers through the following two ways:

(i)Directly. Sending a flood of nonlegitimate packets to the controller, causing congestion, impairing its operational capacity, or even causing it to crash. However, the control plane of SDN is usually not exposed to public networks. The controller is situated behind firewalls or even unreachable outside of the network it belongs to, reducing the efficiency of such attacks.

(ii)Indirectly. Sending a flood of random packets to the data plane, causing the network to redirect numerous packets that don't match existing forwarding rules to the controller, thereby affecting its performance. This method is particularly effective in stateless SDN networks with complex structures or forwarding rules, which is inpossible to set default forwarding settings making it a popular way for attacking SDN networks.

Author is anonymized according to submission requirements

Normally, the packets of DDoS attacks share some certain features, which are often embedded within multiple packet features and possess a certain temporal pattern. Traditional networks struggle to differentiate and effectively block them as they usually can only distinguish packet by one or some certain feature, disregarding their interconnections and temporal changes. Fortunately, the flexibility of SDN facilitates the extraction of these features and targeted mitigation of attack packets. Additionally, machine learning (ML) algorithm offers a relatively effective means to discern such packets from normal traffic.

In this paper we introduce a way to extract features from packs in SDN network with Ryu controller [1] and classify them using ML algorithms. along with related Experiment results. Fig. 1 shows the framework of our job.

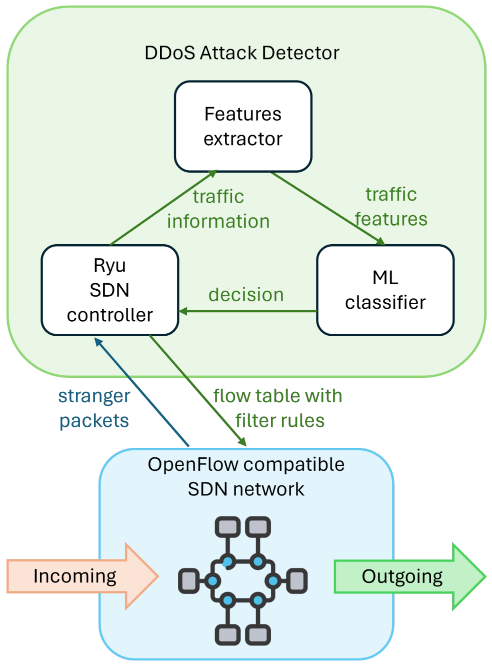

Fig. 1. The framework of our job.

The rest of the paper is organized as follows. Section 2 provides a brief overview of the background of SDN and DDoS attack, and related work. In Section 3 we show how we extract and summarize packet features and manage different connections in the network using Ryu controller. In Section 4 we describe ML algorithms we use to classify packets. Section 5 shows our experiment on a simple simulated network and evaluation datas. Section 6 gives the limitations of our work in this paper and our future work plan. Finally, Section 7 concludes the paper.

II. Backgrounds

A. DDoS Attack

Denial-of-Service (DoS) attacks are malicious attempts to disrupt the normal traffic of a targeted server, service or network by overwhelming the target and its surrounding infrastructure with a flood of Internet traffic. It achieves such effectiveness by utilizing multiple compromised computer systems as sources of attack traffic. Exploited machines can include computers and other networked resources such as IoT devices. The Distributed version of DoS (or DDoS), are often launched by a network of remotely controlled, well organized, known as “Botnet” or “Zombies”, sending a flood of traffic or a large amount of service requets to victim through direct connections or/and third-party hosts, causing target hosts network congestion or computational malfunctioning (e.g., CPU overload). Blocking these types of attacks based on static policies such as IP-address blacklist is diffcult since distributed attackers often have different and dynamic IP addresses. The most utilized DDoS attacks are typically grouped in the following categories: TCP/TCP-SYN flood, UDP flood, ICMP flood and HTTP flood. These attacks can be further categorized into Transport Layer attacks and Application Layer attacks based on the network layer at which they are executed.

Note that Transport Layer attacks, includeing TCP(-SYN)/ UDP/ICMP flood, besides overloading transmission, computing and memory resources in the attack targets, also afect other network elements (i.e., routers and network gateways) in the transmission path. Some of them can even be reflectd and amplified by innocent public service provider. Particularly, DNS amplification attack can generates a huge UDP flood by sending relatively few false DNS query requets to public DNS providers. Causing not only malfunction of target network but also damage to public service.

According to previous studies [2], classifed based on the location of where the detection engine is implemented, there are three defense strategies are typically employed to mitigate DDoS attacks, fig.2 shows how to perform and detect a DDoS attack:

- **Source-based detection**, detect illegal packets sending from local network, implemented at the attacking hosts and their Internet Service Providers (ISPs). Hard to deploy to all hosts in botnet as they are all around the world.
- **Destination-based detection**, filter all malicious packets from incoming traffic, implemented at the victim hosts. A high performance filter is needed to handle massive traffic.
- **Network-based detection**, detect and filter all suspicious packets passing the network, implemented at the intermediate network nodes. Acceptable load for every nodes and cover a wide range of hosts.

Due to the aforementioned reasons, in this paper we focus on how to detect Transport Layer attacks. Which is convenient for intermediate network to detect and causing the most of

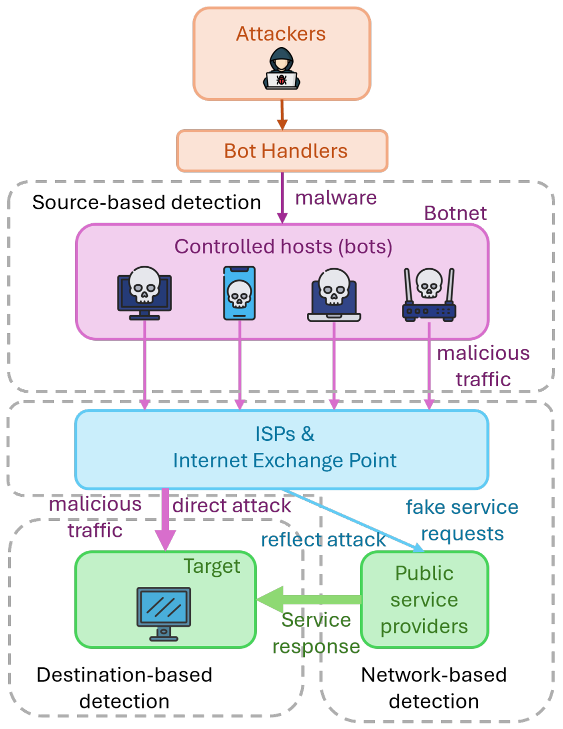

Fig. 2. Perform and detect a DDoS attack.

all damage in various kinds of DDoS attacks. The objective of this paper is to detect attacks in network by deploying defense mechanisms directly at the SDN controllers, which is hopefully to be used by ISPs and Internet exchange points.

B. Related work

A number of ML-based methods for DDoS detection have been introduced in a series of papers. Several studies proposed to use Support Vector Machine (SVM) classifiers to detect DDoS packets. Such as [3], which gives a prototype, and [4]–[6]. Some [7], [8] use other methods such as kNN and Random Forest algorithms. Some papers narrow their work down into some specifc context or application of network to increase performance, such as [9], detect TCP-SYN particularly in P4 SDN. Meanwhile, [10] focus on bandwidth control mechanism. And more attempts [11]–[13] work fine in the context they concern about. Further on, reinforcement learning has been also adopted [14] to this task. As deep learning and artificial intelligence develops, some attempts [15], [16] deploy neural network to their classifier to mitigate DDoS attacks.

C. Dataset

KDD-CUP99 [17] is the data set used for The Third International Knowledge Discovery and Data Mining Tools Competition. [18] This database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment. The KDD training dataset comprises approximately 4.9 million individual connection records, each consisting of 41 features and labeled as either normal or attack, with only one specific attack type per record. Simulated attacks are categorized into the following four types:(i) User to Root Attack, (ii) Remote to Local Attack, (iii) Probing Attack and (iv) Denial of Service Attack [19] which is what we need.

III. Feature extractor

The aim of this paper is to perform DDoS attack detection(DAD), The first step is to extract features from the network. If we define window:

$$w = \{c_i | c_i[addr] = (< IP\_addr_1 >, < IP\_addr_2 >)\}$$

as a set of all connections c between two exactly address, the Internet along with all private networks can be denoted as the universe $\mathbb{W}$ which contains all possible connections in the cyber space. Considering that we are deploying detectors in intermediate network, we treat the data plane as a integrated unit, which can be abstracted as a set of windows $\mathcal{W}$, $\mathcal{W} = \{w_i | w_i \in \mathbb{W}\}_{i=1}^{N_w}$. As each window contains only packets with the same address, the SDN controller can easily shut down a window by modify flow table of data plane. This provide the possibility to treat each window individually and form a temporal sequences of packets features within each window. For a single window, each connection in this window is denoted as a 1x5 vector $fp_i(p_i \in w)$, which is consisted as follows.

$$(Bytes, TCP\_flag, SYN\_flag, \\ UDP\_flag, ICMP\_flag)$$

For each window w_T^t of duration T start at given time offset t, count all packets transfered in T, we get the vector of features for as $fw^t = (\{f_i\}_{i=1}^7)$, each f_i stands for a feature defined and calculated by as follow:

- Size: Bytes transfered in window, normalsized by total traffic size in network.
$$f_1 = \frac{Sum(\{fp_i[Bytes] | fp_i \in w_T^t\})}{Sum(\{fp_j[Bytes] | fp_j \in \mathcal{W}_T^t\})}$$
- Average length: Average length of packets in the window, not normalsized.
$$f_2 = \frac{Sum(\{fp_i[Bytes] | fp_i \in w_T^t\})}{Len(\{fp_i | fp_i \in w_T^t\})}$$
- TCP rate: The percentage of TCP packets out of the total.
$$f_3 = \frac{Sum(\{fp_i[TCP\_flag] | fp_i \in w_T^t\})}{Len(\{fp_i | fp_i \in w_T^t\})}$$
- SYN rate: The percentage of TCP-SYN packets out of the total TCP packets.
$$f_4 = \frac{Sum(\{fp_i[SYN\_flag] | fp_i \in w_T^t\})}{Sum(\{fp_i[TCP\_flag] | fp_i \in w_T^t\})}$$
- UDP rate: The percentage of UDP packets out of the total.
$$f_5 = \frac{Sum(\{fp_i[UDP\_flag] | fp_i \in w_T^t\})}{Len(\{fp_i | fp_i \in w_T^t\})}$$
- ICMP rate: The percentage of ICMP packets out of the total.
$$f_6 = \frac{Sum(\{fp_i[ICMP\_flag] | fp_i \in w_T^t\})}{Len(\{fp_i | fp_i \in w_T^t\})}$$

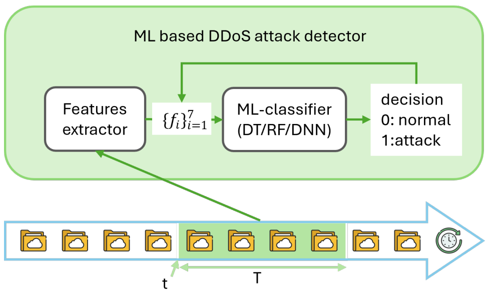

Fig. 3. Features extraction and window classification.

- Network flag: A flag stands if the network under attack. To predict such flag we define a set of special windows w_{ui} for whole network as each of them stand for a remote host connected to any local address.
$$w_{ui} = \{c_j | c_j \in \mathcal{W}, cj[addr] \subset \\ (< IP\_addr_1 > \cup Local\_addr)\}$$
Extract features from w_{ui} using above methods and set this network flag from last special window prediction, and send such vectors to ML-classifier, set the prediction answers to all related network flag.

$$f_7 = classifier(w_u i), (w_u i[IP\_addr_1] \in c_i[addr])$$

IV. ML-based classifier

In our work we consider three different machine learning algorithms as follows, Decision Tree (DT), Random Forest (RF) and Deep Neural Network (DNN), to implement our binary classifiers. The classifier accept the features of each window mentioned above and output a bool value which stand if the window is used for attack, Fig.3 shows how this work.

A. Decision Tree

As a famous algorithm for classifiers, we briefly introduce DT as a directed acyclic graph with three kinds of nodes: (i) Decision nodes, (ii) Chance nodes,(iii) End nodes. In which each chance node represents a "test" on an input features , each branch and decision nodes represents the outcome of the test, and each end node represents a class label. The paths from root to end represent classification rules. The train of a DT is to find the best attribute to split the data and form the structure of the tree. We use the following entropy provided in Toolkit as the criterion, and also other parameters of the DT classifier implemented by scikit-learn [20] are listed in Table. I

$$I_E(i) = -\sum_{j=1}^{m} f(i,j) \log_2 f(i,j)$$

TABLE I
PARAMETERS FOR DT

criterion	entropy
split	best
max_depth	20
min_samples_split	2
min_samples_leaf	1

TABLE II
PARAMETERS FOR RF

criterion	entropy
n_estimators	300
verbose	True
max_depth	20

TABLE III
PARAMETERS FOR DNN

hidden_layer_sizes	(100,200,50)
max_iter	300
verbose	True
max_depth	20

B. *Random Forest*

Random Forest, first introduced in 1995 by Tin Kam Ho [21], operates by constructing a multitude of DTs at training time. Each tree in the forest is trained by a randomly and repeatedly selected subsets of the train dataset. For binary classification tasks, the output of the random forest is the flag selected by most trees. It can help correct for DTs' habit of overfitting to their training set [22]. we expect it will perform better in low-rate attack. The parameters for RF classifier is listed in Table. II

C. *Deep Neural Network*

Neural Network is a expansion of perceptron, while Deep Neural Networks (DNN) can be understood as neural networks with multiple hidden layers between the input and output layer [23]. The value of each node in a layer is determined by the values of the nodes in the previous layer and the weights of the related edges. For nodes in layer $k+1$, we use all n nodes in layer k and weights between them to compute their values.

$$v_i^{k+1} = \sum_{j=1}^{n} v_j^k * w_{ij}^k$$

Table.III lists the parameters we use for DNN classifier implemented by scikit-learn.

V. EXPERIMENT & TEST RESULT

A. *Methodology for Test*

Considering that our method for extracting traffic features considers the data plane as a whole, we use a simple network architecture contains only 2 switchs for Simulating input and output gateways. Meanwhile, as a simulated test network, a single host can handle the task of sending packets from all incoming addresses without requiring a separate host for each address. so we use only two virtual hosts in our network, one for sending incoming traffic, another for outgoing. Fig. 4 show our test network.

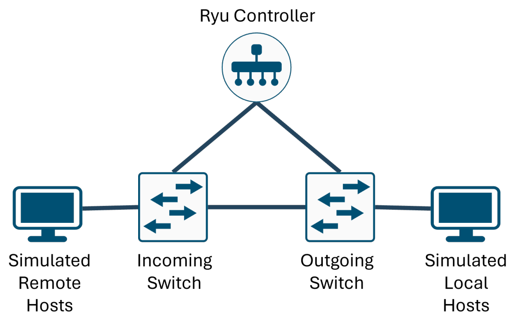

Fig. 4. The architecture of our test network.

We implement binary classifiers using each of the aforementioned three methods and test it using KDD-CUP99 dataset separately. The classifiers was trained on a 40% split of the dataset. And the rest part of dataset is used for generate simulated packets for test.

Note that the KDD-CUP99 dataset does not contain the address information, it is impossible to generate the packets with addresses directly. Fortunately, it contains the flags indicate the the relationship between packets and windows, so we preprocessed the dataset, split the packets in same windows and faked the addresses for each windows, it works well in our test.

B. *Evaluation Metrics*

As we expect a bool output from our binary classifier, it is easy to define indicators of evaluation as follows:

- *true positives (TP)*: classifier prediction give "True" in where we expect it.
- *true negatives (TN)*: classifier prediction give "False" in where we expect it.
- *false positives (FP)*: classifier prediction give "True" which should be "False".
- *false negatives (FN)*: classifier prediction give "False" which should be "True".

Based on these defnitions, it is easy to give the formula for following popular evaluation metrics in machine learning:

- *Accuracy*: The proportion of correctly classified instances.

$$A_{ccuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

- *Precision*: The fraction of correctly-classifed positive windows out of the total number of windows classifed as positive. A higher value of this metric indicates fewer normal packets being misclassified as attacks.

$$P_{recision} = \frac{TP}{TP+FP}$$

- *Recall*: The fraction of correctly-classifed positive windows out of the total number of windows which are

TABLE IV
TEST RESULTS

algorithm	Accuracy	Precision	Recall	f-1 score
Decision Tree	0.97520	0.97531	0.96217	0.96864
Random Forest	0.97791	0.97791	0.95877	0.96824
DNN	0.99721	0.99715	0.99662	0.99689

actually positive. This is very important beacause any leak of attack traffic is fatal to the network.

$$R_{call} = \frac{TP}{TP + FN}$$

- *f-1 score*: The fraction considers both precision and recall. Can evaluate the result in a balanced way.

$$f_{1Score} = \frac{2 * P_{recision} * R_{ecall}}{P_{recision} + R_{ecall}}$$

C. Test Results

We test the workbench using above methods the Table.IV shows the result.

Our testing results on KDD-CUP99 dataset are acceptable, all of three ML algorithm tested get the evaluation metrics over 0.95 which shows that we have over 95% of confidence we can detect a DDoS attack. In detail, DT and RF algorithms exhibit relatively same performance as they are similar algorithms. And DNN performs better and get all metrics over 0.99.

VI. LIMITATIONS & PLANS

Although our testing has shown promising results, there is still a considerable gap between this result and practical application. The main limitations of current work will be discussed in following.

A. Limitations of Dataset

The KDD-CUP99 a well-known dataset in DAD tasks. But it was created in year 1999 and can now be considered outdated. It only contains the attack of simple DoS but without distributed data. Although we revised this by faking different addresses, it still different from real DDoS attack.

The critical weakness of this dataset is its severe inadequacy in comparison to the real traffic of modern Internet and DDoS attack. According to the survey [19] on KDD-CUP99, it contains $4,176,086$ records of packets and only $291,556$ distinct records. But according to technical report [24], the typical DDoS attack in year 2023 often transfers more than $1.9Tbit$ in a second, which is approximately 1.3 trillion packets as the default MTU for Ethernet is 1500. It is a gaint difference. Also the high discrepancy in quantity between total records and distinct records means a high reduction rate in data which may leads a overfitting of classifiers.

B. Limitations of Methodology

Besides from dataset, our method also have limitations. The most obvious one is the way how packets were groupd. We simply distinguish packets by the concept of “window”, which means once a DDoS attack is detected, all of the connections between the two address this window related will be shutdown. It is not a big deal in the past, but may will cause a big accessibility issue nowadays beacause a massive hosts are connecting behind many NAT networks and share the same IP address of NAT gateways [25].

Moreover, our method is take the data plane as a unified entity and unable to use the information of route forwarding within network, which is a good way to detect malicious packets as normal traffic between two address usually follow a explicit path and will never reach some nodes in the network.

Last, the test bench is only implemented by a software simulated network simply formed by 4 node and only capable for a very light traffic, so it is impossible to test in a more realistic scenarios. As a result, the real-time detection capability can not be tested as both network and controller is lack of performance.

C. Future Works

The most crucial future task is to conduct testing using newer datasets. In the short term, we have decided to replace the existing dataset with the CICDoS2019 [26] dataset for our testing purposes. In further, we plan to generate our own dataset using Spirent N4U traffic generator [27] as what other paper [9] did. Additionally, if possible, deploying improved and more realistic testing equipment and platforms will be beneficial. In such situation, every switchs in the network will be treated separately and hopefully a better performance will show in further tests.

VII. CONCLUSION

In this paper, we introduced a workable detector prototype for DDoS attack. In our testing, all three machine learning algorithms exhibited acceptable performance, with all metrics over 0.95. Due to implementation limitations, this prototype cannot yet be used in real-world networks. However, it demonstrates good potential for improvement and has broad applicability.

REFERENCES

[1] Ryu SDN Framework Community, “Ryu controller,” https://ryu-sdn.org/, 2014, archived in: Jun 10, 2022. I

[2] S. T. Zargar, J. Joshi, and D. Tipper, “A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013. II-A

[3] M. S. Hoyos Ll, G. A. Isaza E, J. I. Vélez, and L. Castillo O, “Distributed denial of service (ddos) attacks detection using machine learning prototype,” in *Distributed Computing and Artificial Intelligence, 13th International Conference*, S. Omatu, A. Semalat, G. Bocewicz, P. Sitek, I. E. Nielsen, J. A. García García, and J. Bajo, Eds. Cham: Springer International Publishing, 2016, pp. 33–41. II-B

[4] A. Ramamoorthi, T. Subbulakshmi, and D. S. M. Shalinie, “Real time detection and classification of ddos attacks using enhanced svm with string kernels,” *2011 International Conference on Recent Trends in Information Technology (ICRTIT)*, pp. 91–96, 2011. [Online]. Available: https://api.semanticscholar.org/CorpusID:7146786 II-B

[5] K. S. Sahoo, B. K. Tripathy, K. Naik, S. Ramasubbareddy, B. Balusamy, M. Khari, and D. Burgos, “An evolutionary svm model for ddos attack detection in software defined networks,” *IEEE Access*, vol. 8, pp. 132 502–132 513, 2020. II-B

[6] J. Ye, X. Cheng, J. Zhu, L. Feng, and L. Song, "A ddos attack detection method based on svm in software defined network," *Security and Communication Networks*, vol. 2018, p. 9804061, Apr 2018. [Online]. Available: https://doi.org/10.1155/2018/9804061 II-B

[7] A. Aljuhani, "Machine learning approaches for combating distributed denial of service attacks in modern networking environments," *IEEE Access*, vol. 9, pp. 42 236–42 264, 2021. II-B

[8] J. a. H. Corrêa, P. M. Ciarelli, M. R. N. Ribeiro, and R. S. Villaça, "Ml-based ddos detection and identification using native cloud telemetry macroscopic monitoring," *J. Netw. Syst. Manage.*, vol. 29, no. 2, apr 2021. [Online]. Available: https://doi.org/10.1007/s10922-020-09578-1 II-B

[9] F. Musumeci, A. C. Fidanci, F. Paolucci, F. Cugini, and M. Tornatore, "Machine-learning-enabled ddos attacks detection in p4 programmable networks," *Journal of Network and Systems Management*, vol. 30, no. 1, p. 21, Nov 2021. [Online]. Available: https://doi.org/10.1007/s10922-021-09633-5 II-B, VI-C

[10] H. A. Alamri and V. Thayananthan, "Bandwidth control mechanism and extreme gradient boosting algorithm for protecting software-defined networks against ddos attacks," *IEEE Access*, vol. 8, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3033942 II-B

[11] L. Zhu, X. Tang, M. Shen, X. Du, and M. Guizani, "Privacy-preserving ddos attack detection using cross-domain traffic in software defined networks," *IEEE J. Sel. Areas Commun.*, vol. 36, 2018. [Online]. Available: https://doi.org/10.1109/JSAC.2018.2815442 II-B

[12] W. Zhijun, X. Qing, W. Jingjie, Y. Meng, and L. Liang, "Low-rate ddos attack detection based on factorization machine in software defined network," *IEEE Access*, vol. 8, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.2967478 II-B

[13] N. Ravi and S. M. Shalinie, "Learning-driven detection and mitigation of ddos attack in iot via sdn-cloud architecture," *IEEE Internet Things J.*, vol. 7, 2020. [Online]. Available: https://doi.org/10.1109/JIOT.2020.2973176 II-B

[14] K. A. Simpson, S. Rogers, and D. P. Pezaros, "Per-host ddos mitigation by direct-control reinforcement learning," *IEEE Trans. Netw. Serv. Manage.*, vol. 17, 2020. [Online]. Available: https://doi.org/10.1109/TNSM.2019.2960202 II-B

[15] A. Bhardwaj, V. Mangat, and R. Vig, "Hyperband tuned deep neural network with well posed stacked sparse autoencoder for detection of ddos attacks in cloud," *IEEE Access*, vol. 8, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3028690 II-B

[16] X. Yuan, C. Li, and X. Li, "Deepdefense: Identifying ddos attack via deep learning," in *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2017, pp. 1–8. II-B

[17] S. J. Stolfo and etc, "KDD cup 1999 data," 1999. [Online]. Available: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html II-C

[18] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml II-C

[19] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6. II-C, VI-A

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. IV-A

[21] T. K. Ho, "Random decision forests," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 1, 1995, pp. 278–282 vol.1. IV-B

[22] T. Hastie, R. Tibshirani, and J. Friedman, *Random Forests*. New York, NY: Springer New York, 2009, pp. 587–604. [Online]. Available: https://doi.org/10.1007/978-0-387-84858-7_15 IV-B

[23] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608014002135 IV-C

[24] O. Yoachimik and J. Pacheco, "Ddos threat report for 2023 q4," Jan 2024. [Online]. Available: https://blog.cloudflare.com/ddos-threat-report-2023-q4 VI-A

[25] O. Kanaris and J. Pouwelse, "Mass adoption of nats: Survey and experiments on carrier-grade nats," 2023. VI-B

[26] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *2019 International Carnahan Conference on Security Technology (ICCST)*, 2019, pp. 1–8. VI-C

[27] Spirent Test Center, "Spirent n4u trafc generator," 2020, accessed Mar 2024. [Online]. Available: https://www.spirent.com/products/testcenter VI-C