International Journal of Mathematical, Engineering and Management Sciences
Vol. 7, No. 1, 81-91, 2022
https://doi.org/10.33889/IJMEMS.2022.7.1.006

A Step Towards Generation of DoS/DDoS Attacks Dataset for Docker-Centric Computing

Aparna Tomar
Department of Computer Science and Engineering,
Graphic Era University, Dehradun, India.
E-mail: aparnatomar29@gmail.com

Preeti Mishra
Department of Computer Science,
Doon University, Dehradun, India.
Corresponding author: scholar.preeti@gmail.com

Rahul Bisht
Department of Computer Science and Engineering,
Graphic Era University, Dehradun, India.
E-mail: me.rahul.bisht@gmail.com

Peddoju Sateesh Kumar
Department of Computer Science and Engineering,
Indian Institute of Technology Roorkee, Roorkee, India.
E-mail: sateesh@ieee.org

(Received on August 19, 2021; Accepted on December 23, 2021)

Abstract
Docker provides an effective containerized environment for modern computing. However, the security issues present in Docker provide an edge to the attackers thus resulting in various attacks. Denial of Service (DoS) and Distributed Denial of Service (DDoS) are the common ones. In this paper, DoS and DDoS attack datasets have been generated using realistic testbed environments as older datasets have their own set of limitations, making them insufficient for today's computing. An architectural framework is provided to depict the process of packet capturing and feature extraction. A total of 45 features are extracted using Flowtbag among which 17 best features are selected using the average correlation coefficient. Six machine learning algorithms namely Logistic Regression (LR), Naïve Bayes (NB), K-Nearest Neighbors (KNN), Decision Tree (DT), Random Forest (RF), and Support Vector Machine (SVM) are applied on datasets with full features and selected features to obtain accuracy, precision, recall, and F1 score. NB gave the lowest accuracy of 0.94917 on full features and DT provided the most accurate results with a performance matrix of 0.99254 accuracy, 0.997 precision, 0.998 recall, and 0.997 F1 Score. Whereas on selected features, accuracies of both the algorithms increased to 0.962434 and 0.992703 respectively.

Keywords- Docker, Docker security, Docker swarm, Dataset generation, DoS/DDoS.

1. Introduction

Cloud computing is one of the distributed computing paradigms which employs a decentralized computing framework by making use of the compute, network, and storage which are close to the users. The container technologies such as Docker provide a portable, high-performance, and lightweight alternative solution for hosting applications at edge servers. It makes Docker-based applications much faster than VM-based applications at the edges. However, security is a major loophole in Docker containers (White et al., 2021). A number of attacks can take place in the

Docker environment. The Denial of Service (DoS) and Distributed Denial of Service (DDoS) are the common attacks that can be easily launched due to the availability of multiple attacking tools. They can cause huge financial losses to organizations (Tomar et al., 2020). In the entire world, more than 20% of the organizations have reported at least one incident of DDoS attack (Somani et al., 2017). Both of these attacks focus on targeting the system or services thus making them unavailable for authorized customers.

On the basis of several studies, it was found out that the older datasets have their own set of restrictions which limit their usability (Al-Hadhrami and Hussain, 2020; Koroniotis et al., 2019; Moustafa and Slay, 2015). Some of the limitations are as follows: (i) KDDCUP'99 contains a large number of unnecessary and missing records present in the training set which may affect the final results. (ii) NSLKDD dataset is the upgraded version of the KDD'99 dataset that solves many issues that were present in KDD'99. However, it is also observed that it does not provide expected and appropriate results in the modern attacking environment. (iii) Most of the older datasets (except KDD'99 and DARPA) are not labeled, which makes it difficult to filter out the malicious traffic from the normal traffic. As container-based technologies are the demand of current times hence it raises a strong need to generate a dataset in the containerized environment. To the best of our knowledge, there is no such dataset available that is generated in the Docker-based environment using Docker swarm.

In this paper, we present realistic containerized testbed environments for DoS and DDoS attacks. A detailed description of architectural frameworks along with the tools and methodologies have also been presented in both scenarios. The attack logs are initially extracted in PCAP format. Furthermore, 45 features are extracted using Flowtbag tool during the process of feature extraction. Lastly, feature selection has been carried out on the basis of the average correlation coefficient of features. The processed dataset with optimal features has been evaluated using various machine learning algorithms. The major contributions of this paper can be summarized as below:

- To design and describe realistic testbed environments for DoS and DDoS attacks in the containerized platforms.
- To design and describe an architectural framework for the dataset generation process.
- To extract features for each network trace and provide their description.
- To evaluate the dataset with full features and selected features for various machine learning algorithms.

The rest of the paper is organized as follows: Section 2 gives the details of the literature review. The testbed environment for generating DoS attack dataset is presented in section 3. Section 4 describes Docker swarm in brief. Section 5 presents the testbed environment for DDoS attack dataset generation. Section 6 describes traffic capturing and feature extraction. Section 7 describes the experimental results which are being produced by using the classification algorithms. Finally, section 8 concludes the paper.

2. Related Work

Gupta and Badve (2016) described how cloud computing have always been a primary target for DoS and DDoS attacks. Even after an immense amount of advancement in the field of security, these attacks can easily be launched and can cause damage to organizations. They presented various security challenges in Cloud computing along with a detailed description of DoS attacks, their types, and various tools that can be used to launch this attack.

Chelladhurai et al. (2016) presented various threats to Docker under which attacks like ARP spoofing, MAC flooding, and DoS are discussed in brief. They also proposed and discussed the solution to address the security of Docker servers. A solution was proposed to prevent Docker from DoS attacks. The main focus of their solution is to control the memory limit.

Moustafa and Slay (2015) discussed the drawbacks of KDDCUP99 and NSLKDD which are the most widely adopted data sets for NIDS. The main focus was the generation of the UNSW-NB15 dataset. The details of the testbed environment were presented and dataset statistics are provided along with an architectural framework that describes the entire process of the dataset generation. Lastly, the final shape of the generated dataset was given by providing details like how many CSV files are there and how many records are present in each file.

Bhatia et al. (2014) described how the research in the field of anomaly detection particularly DDoS attack detection suffers because of the absence of recently generated datasets. They stated all the publicly available datasets are either too old for the current scenario of network security or they are full of unnecessary and missing records which stops them from providing reliable and accurate results. They also presented a testbed architecture that generates traffic for various types of DDoS attacks, Flash Events (FEs), and other benign traffic traces.

Bhatia et al. (2018) provided a brief description of DoS and DDoS attacks and the three main types of DDoS attacks. A description of how-to setup Docker swarm has also been given. The main focus was to depict how Docker swarm is vulnerable to DoS and DDoS attacks. The conclusion of the analysis indicated that Docker swarm is extremely vulnerable to DoS/DDoS attacks and it can be brought down within minutes if any DoS/DDoS takes place.

Tien et al. (2019) proposed a container anomaly analysis tool known as KubAnomaly that can be used to detect anomalies in Kubernetes which is a cloud container orchestration tool. The potency of the proposed model was illustrated by comparing its accuracy with the accuracy of other machine learning algorithms and it was found to be 96%. The proposed model successfully identified four real attacks that were carried out by hackers in September 2018.

3. Testbed Environment for DoS Attack

In order to generate the dataset for the DoS attack, the testbed environment had been set up which is shown in Figure 1. As per the testbed environment, the attack was launched from the host machine to the Docker running inside the host. Along with Docker, some other tools have also been installed in the host such as Bit-Twist (normal traffic generator), tcpdump, and wireshark. Inside Docker, two httpd and one nginx containers were running having IP addresses 172.17.0.2, 172.17.0.3, and 172.17.0.4 respectively. Nginx container was the victim container that was being flooded by using hping3 and GoldenEye tools respectively. Hping3 was used to perform the TCP SYN flood whereas GoldenEye was used to perform HTTP SYN flood.

The following command shows how hping3 is used to flood the victim: *hping3 -V -c 100 -d 100 -S -p 80 –flood 172.17.0.4*. This command indicates that 100 packets (-c 100) are being sent each of size 100 bytes (-d 100). The SYN Flag (-S) should be enabled. To perform the attack on the victim, port 80 (-p 80) has been specified and the IP address provided at the last is the IP address of the victim. The -flood flag has been used to send packets as fast as possible. On the other hand, the following command depicts the usage of the GoldenEye in flooding:

sudo./goldeneye.py http://172.17.0.4:80/ -s 2 -m random. It means that we are hitting port number 80 of IP address 172.17.0.4 in 'random' mode with 10 workers running 2 connections each.

The attack traffic from the nginx container and normal traffic from Bit-Twist will be captured in the form of pcap files using tcpdump and these pcap files can later be viewed in wireshark. If we want to flood any other running container then we can simply change the IP address of the victim in the command of hping3 and GoldenEye.

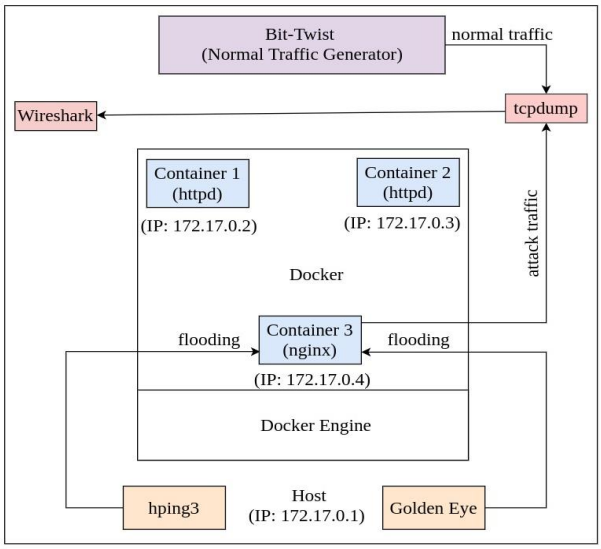

Figure 1. The testbed environment for dos attack.

4. Docker Swarm

A Docker swarm is a group of machines that are running Docker and joined into a cluster. Docker swarm is a tool for container orchestration. The process of managing and controlling multiple Docker containers as a single service is known as orchestration. Various tools are available for orchestration, such as Docker swarm, Kubernetes, Apache Mesos (Bhatia et al., 2018; Wenhao and Zheng, 2020). In Figure 2, it is clearly shown that the httpd service is running in swarm manager and 4 different replicas are created of the same service that will run in 4 different worker nodes. We can create as many replicas as possible, depending on our needs.

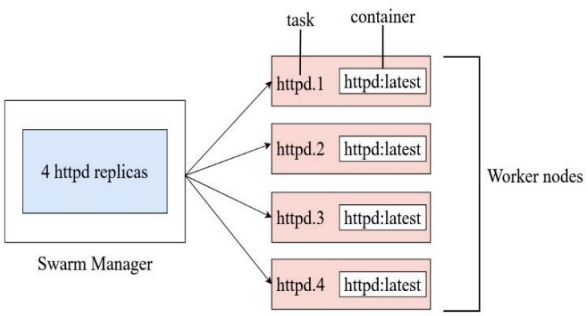

Figure 2. Docker swarm.

5. Testbed Environment for DDoS Attack

In order to generate the dataset for DDoS using Docker swarm, the testbed environment had been set up, as shown in Figure 3. As per the testbed environment, the attack was launched from Docker swarm to the victim container present inside Docker. Both Docker swarm and Docker were running inside the host having IP address 172.7.0.1. Apart from these two, Bit-Twist, tcpdump, and Wireshark were also running inside the same host.

In Docker swarm, swarm manager was having IP address 172.16.22.39 and a service called *ddos_attack_service* was running inside the swarm manager which was scaled to 30 (i.e., 30 containers running the same service). All these replicas had distinct IP addresses. Now, the same service i.e., *ddos_attack_service*, running in the swarm manager, was also running in the replicas. These replicas were under the control of the swarm manager. They behaved as bots and were used to flood the victim machine. In the Docker host, three containers were running just like the testbed environment of the DoS attack. Here, the first container (victim container) having IP address 172.17.0.32 which was running the httpd service was flooded using the Docker swarm cluster.

When the attack took place, the *ddos_attack_service* was running in the manager as well as in the created replicas. These replicas were used to flood the victim container using hping3 (used for TCP-IP DDoS) and GoldenEye (used for HTTP DDoS). The description of the usage of hping3 and GoldenEye for flooding the victim container has already been described in Section 3.

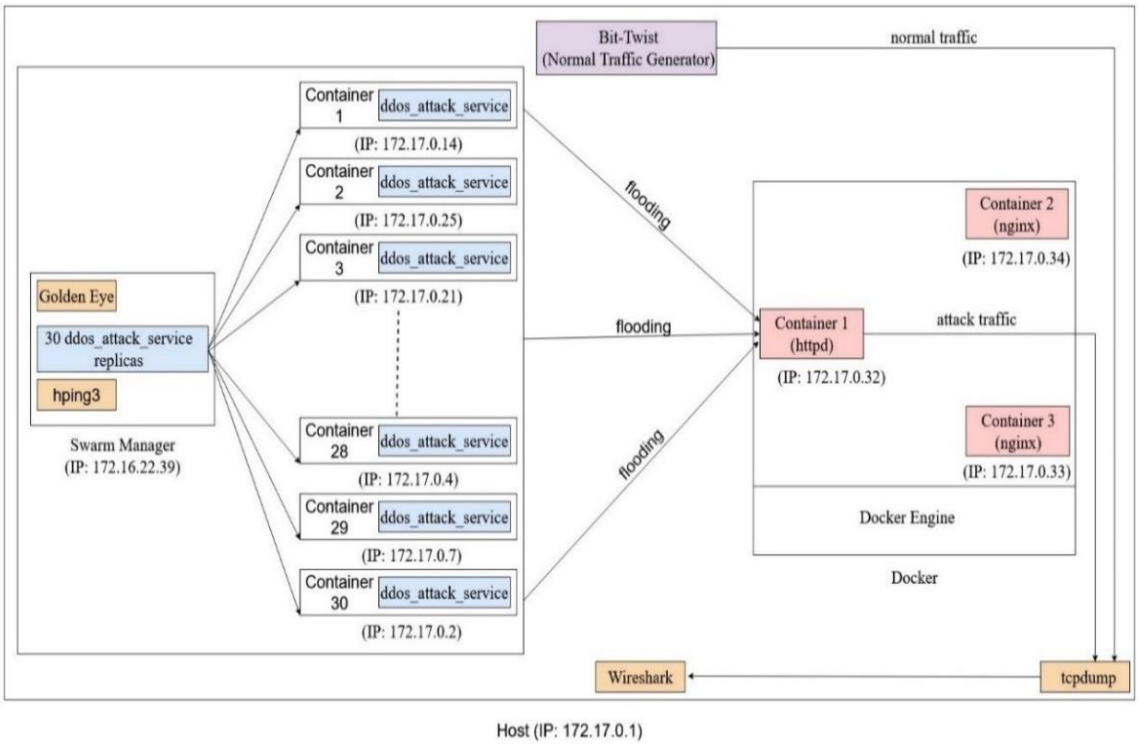

Figure 3. The testbed environment for DDoS attack.

6. Traffic Capturing and Feature Extraction

The process of traffic capturing and feature extraction which is the same for both the datasets is being shown in Figure 4. When the attack on the victim Docker container was taking place, the attack traffic and normal traffic were captured by tcpdump in the form of PCAP files. These captured files can be viewed in wireshark. The captured PCAP files were given to Flowtbag as input in order to convert them into CSV files. The extracted CSV files contain a total of 45 features. The names of the extracted features are given in Table 1. The selection of best features on the basis of the average correlation coefficient has been discussed in the next section.

The Flowtbag tool has been used for feature extraction of bidirectional traffic flows logs. A total of 45 features were generated among which 30 statistical features were calculated separately in the forward and backward directions. The remaining 15 features include the basics of the packets without having any resemblance to the bidirectional flows. The CSV file through the Flowtbag tool can be obtained just by running one simple command: *sudo .//flowtbag filename.pcap > test.out* .

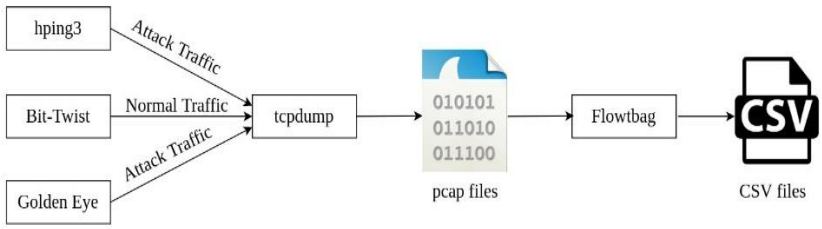

Figure 4. Architectural framework for dataset generation.

Table 1. Extracted features.

1. srcip	10. min_fwdpktl	19. mean_fwdiat	28. mean_act	37. sflow_bwdpkt
2. srcport	11. mean_fwdpktl	20. max_fwdiat	29. max_act	38. sflow_bwdbts
3. dstip	12. max_fwdpktl	21. std_fwdiat	30. std_act	39. fwdpsh_cnt
4. dstport	13. std_fwdpktl	22. min_bwdiat	31. min_idl	40. bwdpsh_cnt
5. prt	14. min_bwdpktl	23. mean_bwdiat	32. mean_idl	41. fwdurg_cnt
6. total_fwdpkt	15. mean_bwdpktl	24. max_bwdiat	33. max_idl	42. bwdurg_cnt
7. total_fwdvol	16. max_bwdpktl	25. std_bwdiat	34. std_idl	43. total_fwdhlen
8. total_bwdpkt	17. std_bwdpktl	26. duration	35. sflow_fwdpkt	44. total_bwdhlen
9. total_bwdvol	18. min_fwdiat	27. min_act	36. sflow_fwdbts	45. dscp

The CSV file generated by Flowtbag has various features. In order to provide feature description clearly, features are divided into two categories:

6.1 Feature Description of Forward and Backward Direction Packets

The features included in this category are the features of forward and backward direction packets i.e. the packets going from source to destination and the packets going from destination to source. Feature names and their description of forward and backward direction packets are given in Table 2.

Table 2. Feature description of forward and backward direction packets.

#	Feature Name	Feature Description
1	total_fwdpkt / total_bwdpkt	Total packets in forward/backward direction
2	total_fwdvol / total_bwdvol	Total volume of packets in forward/backward direction
3	min_fwdpktl / min_bwdpktl	Minimum length of a packet in forward/backward direction
4	mean_fwdpktl / mean_bwdpktl	Mean size of packet in forward/backward direction
5	max_fwdpktl / max_bwdpktl	Maximum length of a packet in forward/backward direction
6	std_fwdpktl / std_bwdpktl	Standard deviation size of packet in forward/backward direction
7	min_fwdiat / min_bwdiat	Minimum time between two packets sent in forward/backward direction
8	mean_fwdiat / mean_bwdiat	Mean time between two packets sent in forward/backward direction
9	max_fwdiat / max_bwdiat	Maximum time between two packets sent in forward/backward direction
10	std_fwdiat / std_bwdiat	Standard deviation time between two packets sent in forward/backward direction
11	sflow_fwdpkt / sflow_bwdpkt	The average number of packets in a sub flow in forward/backward direction
12	sflow_fwdbts / sflow_bwdbts	The average number of bytes in a sub flow in forward/backward direction
13	fwdpsh_cnt / bwdpsh_cnt	Number of times the PSH flag was set in packets traveling in forward/backward direction
14	fwdurg_cnt / bwdurg_cnt	Number of times the URG flag was set in packets traveling in forward/backward direction
15	total_fwdhlen / total_bwdhlen	Total bytes used for headers

6.2 Feature Description of Packets having no Direction

It involves the description of general features that has no connection with bidirectional flows. The names and description of features for all such packets are provided in Table 3.

Table 3. Feature description of packets having no direction.

#	Feature Name	Feature Description
1	srcip	source ip address
2	srcport	source port
3	dstip	destination ip address
4	dstport	destination port
5	prt	protocol
6	duration	duration of the flow
7	min_act	Minimum time a flow was active before becoming idle
8	mean_act	Mean time a flow was active before becoming idle
9	max_act	Maximum time a flow was active before becoming idle
10	std_act	Standard deviation time a flow was active before becoming idle
11	min_idl	Minimum time a flow was idle before becoming active
12	mean_idl	Mean time a flow was idle before becoming active
13	max_idl	Maximum time a flow was idle before becoming active
14	std_idl	Standard deviation time a flow was idle before becoming active
15	dscp	Differentiated Services Code Point

7. Experimental Results and Discussion

After the preprocessing of the dataset, six classification algorithms namely Logistic Regression (LR), Naive Bayes (NB), K-Nearest Neighbors (KNN), Decision Tree (DT), Random Forest (RF), and Support Vector Machine (SVM) were individually implemented in order to find the accuracy, precision, recall and F1 score of the dataset with and without feature selection. Without feature selection, NB gave an accuracy of 0.94917 with precision, recall, and F1 score of 0.994, 0.956, and 0.975 respectively. These results are achieved by manipulating the hyperparameters.

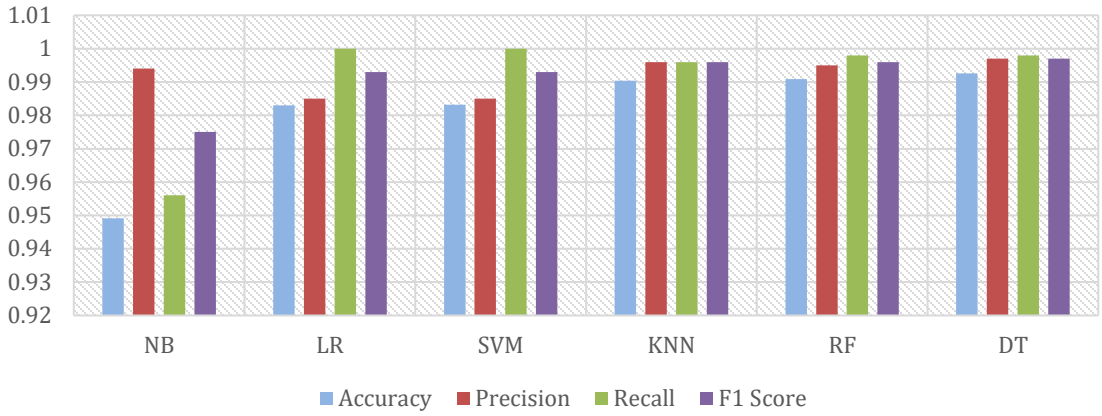

Figure 5. Accuracy, precision, recall and F1 score for full features.

A similar analysis is performed for LR, SVM, KNN, RF, and DT. LR and SVM provided an accuracy of 0.983026 and 0.983211 respectively and also performed better than NB on every matrix. KNN and RF achieved approximately similar accuracies of 0.990384 and 0.990879. DT provided the most accurate results with a performance matrix of 0.992549 accuracy, 0.997 precision, 0.998 recall, and 0.997 F1 Score. The results are shown in Figure 5. Further, we deployed machine learning algorithms on the best feature set using Correlation Coefficient to improve the results and to remove redundant features in the dataset.

7.1 Correlation Coefficient

Correlation is a technique that is used to find the association between two variables. Pearson's correlation coefficient (r) is a measure of the strength of the association between the two variables. The value of the correlation coefficient lies between -1 to 1. A positive value of correlation depicts that both the variables are moving in the same direction which means that if the value of one variable will increase, the value of another variable will also increase. A correlation of 1.0 shows a perfect positive correlation. A negative value of correlation depicts that the variables are moving in opposite directions which means that an increment in the value of one variable will cause a decrement in the value of another variable. A correlation of -1.0 shows a perfect negative correlation. When the value of correlation is 0 then it depicts that there is no linear relationship between the variables.

In order to calculate the Pearson's correlation coefficient between the features, a correlation matrix was generated using Python and an average correlation coefficient was then calculated for all the features with the help of the correlation matrix. The idea is that the features having lower values of correlation coefficient would be more suitable in our dataset (Koroniotis et al., 2019). Table 4 presents the average correlation coefficient of the features in ascending order.

On the basis of the average scores of correlation coefficients, 20 best features were extracted. However, 4 features namely duration, min act, mean act, and max act were found to have similar values of the average correlation coefficient. Hence, 17 best features were extracted. On these selected features, the six classification algorithms were again implemented to find the accuracies in order to know how the accuracy can vary for full features and selected features. The sequence of

the algorithm having the highest accuracy to the algorithm having the lowest accuracy is the same for both full features and selected features. In the case of four algorithms namely NB, LR, RF, and DT, accuracies on selected features increased to 0.962434, 0.983057, 0.99216, and 0.992703 respectively whereas, in the case of SVM and KNN the accuracies decreased to 0.98318 and 0.990353 respectively. Accuracy, precision, recall, and F1 score for selected features have been shown in Figure 6.

Table 4. Average correlation coefficient scores.

Feature	Avg CC	Feature	Avg CC	Feature	Avg CC
1. min_fwdpktl	-0.36807	13. std_fwdiat	0.26609	25. max_bwdpktl	0.36624
2. min_bwdpktl	-0.36733	14. max_fwdiat	0.2813	26. total_fwdhlen	0.38768
3. srcport	-0.01696	15. max_bwdiat	0.31309	27. total_fwdpkt	0.40197
4. mean_fwdpktl	0.10538	16. std_bwdpktl	0.31603	28. sflow_fwdpkt	0.40197
5. std_fwdpktl	0.11396	17. duration	0.33631	29. bwdpsh_cnt	0.40944
6.dstport	0.13831	18. min_act	0.33631	30. total_bwdhlen	0.40951
7. min_fwdiat	0.14654	19. mean_act	0.33631	31. total_bwdpkt	0.41306
8. mean_bwdiat	0.14925	20. max_act	0.33631	32. sflow_bwdpkt	0.41306
9. min_bwdiat	0.14976	21. mean_bwdpktl	0.35578	33. total_fwdvol	0.41622
10. mean_fwdiat	0.15206	22. fwdpsh_cnt	0.36507	34. sflow_fwdbts	0.41622
11. max_fwdpktl	0.18547	23. total_bwdvol	0.3652		
12. std_bwdiat	0.26003	24. sflow_bwdbts	0.3652		

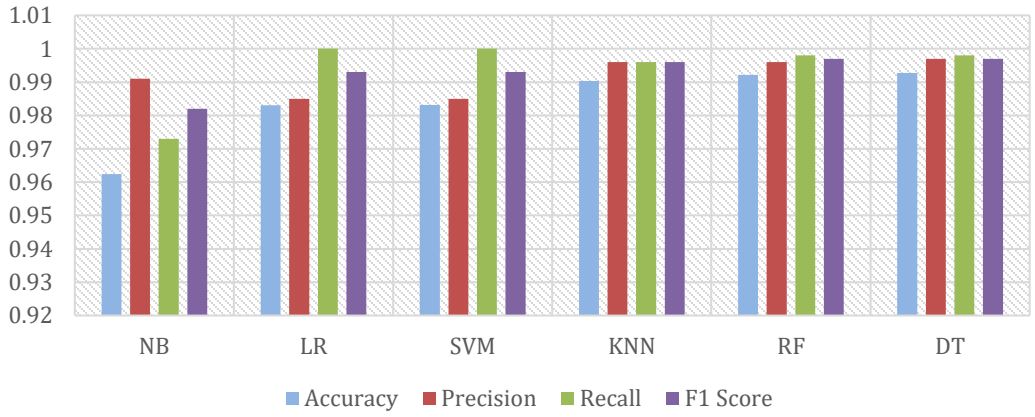

Figure 6. Accuracy, precision, recall and F1 score for selected features.

8. Conclusion

Docker is a crucial part of modern technologies such as cloud computing which are adopting Docker as-a-platform. There is a very high possibility of attacks taking place in the Docker environment. Among all the possible attacks, DoS and DDoS can be easily launched as the tools used in the launching of these attacks are very easily available. Incompetency of the older datasets and the absence of datasets being generated in the Docker environment became the reasons for the generation of the new dataset.

In this paper, we presented the realistic testbed environments for DoS and DDoS attacks in a containerized environment. These testbed environments are used to generate the datasets for both

of these attacks. An architectural framework has also been provided which describes the entire process of dataset generation for both the attacks. We further listed down all the 45 features that were extracted by using the Flowtbag tool and provided their description Lastly, 17 best features were extracted by calculating the average correlation coefficient of features. The generated datasets have been evaluated using six machine learning algorithms namely logistic regression (LR), Naive Bayes (NB), K-nearest neighbors (KNN), decision tree (DT), random forest (RF), and support vector machine (SVM) to find the accuracy, precision, recall and F1 score of the dataset with and without feature selection. On full features, DT provided the most accurate results with a performance matrix of 0.992549 accuracy, 0.997 precision, 0.998 recall, and 0.997 F1 Score. On selected features, the best result was again provided by DT. Values of precision, recall, and F1 score were the same as that of full features whereas the accuracy increased to 0.992703. This dataset is available for research purposes and interested readers can obtain it by e-mailing the authors. In the future, we will propose an efficient DoS/DDoS attack detection algorithm by exploring advanced machine learning algorithms in the containerized platform.

Conflict of Interest

As confirmed by the authors, there is no conflict of interest for this publication.

Acknowledgments

The authors would like to thank the editor and anonymous reviewers for their valuable time.

References

Al-Hadhrami, Y., & Hussain, F.K. (2020). Real time dataset generation framework for intrusion detection systems in IoT. *Future Generation Computer Systems*, *108*, 414-423. https://doi.org/10.1016/j.future.2020.02.051.

Bhatia, S., Schmidt, D., Mohay, G., & Tickle, A. (2014). A framework for generating realistic traffic for distributed denial-of-service attacks and flash events. *Computers & Security*, *40*, 95-107. https://doi.org/10.1016/j.cose.2013.11.005.

Chelladhurai, J., Chelliah, P.R., & Kumar, S.A. (2016). Securing docker containers from denial of service (dos) attacks. In *2016 IEEE International Conference on Services Computing (SCC)* (pp. 856-859). IEEE. San Francisco, USA. https://doi.org/10.1109/scc.2016.123.

Gupta, B.B., & Badve, O.P. (2017). Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a cloud computing environment. *Neural Computing and Applications*, *28*(12), 3655-3682. https://doi.org/10.1007/s00521-016-2317-5.

Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2019). Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-Iot dataset. *Future Generation Computer Systems*, *100*, 779-796. https://doi.org/10.1016/j.future.2019.05.041.

Moustafa, N., & Slay, J. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)* (pp. 1-6). IEEE. Canberra, Australia. https://doi.org/10.1109/milcis.2015.7348942.

Somani, G., Gaur, M.S., Sanghi, D., Conti, M., & Buyya, R. (2017). DDoS attacks in cloud computing: Issues, taxonomy, and future directions. *Computer Communications*, *107*, 30-48. https://doi.org/10.1016/j.comcom.2017.03.010.

Tien, C.W., Huang, T.Y., Tien, C.W., Huang, T.C., & Kuo, S.Y. (2019). KubAnomaly: anomaly detection for the docker orchestration platform with neural network approaches. *Engineering Reports*, *1*(5), e12080. https://doi.org/10.1002/eng2.12080.

Tomar, A., Jeena, D., Mishra, P., & Bisht, R. (2020). Docker security: A threat model, attack taxonomy and real-time attack scenario of dos. In *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)* (pp. 150-155). IEEE. Noida, India. https://doi.org/10.1109/confluence47617.2020.9058115.

Wenhao, J., & Zheng, L. (2020). Vulnerability analysis and security research of docker container. In *2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE)* (pp. 354-357). IEEE. Dalian, China. https://doi.org/10.1109/iciscae51034.2020.9236837.

Bhatia, G., Choudhary, A., & Dadheech, K. (2018, April). Behavioral analysis of docker Swarm under DoS/DDoS attack. In *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)* (pp. 985-991). IEEE. Coimbatore, India. https://doi.org/10.1109/icicct.2018.8472953.

White, A., Galloway, M., O'Boyle, P., & Wyllie, S. (2021, March). The Impact of DDoS attacks on the power usage of virtual execution environments. In *SoutheastCon 2021* (pp. 1-6). IEEE. Atlanta, USA. https://doi.org/10.1109/southeastcon45413.2021.9401839.