International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

A Novel Framework for Detection of DoS/DDoS Attack Using Deep Learning Techniques, and An Approach to Mitigate the Impact of DoS/DDoS attack in Network Environment

Gottapu Sankara Rao [1]*, Dr. P. Krishna Subbarao[2]

Submitted: 25/08/2023 **Revised**: 13/10/2023 **Accepted**: 24/10/2023

Abstract: DoS attacks are a major network security issue. The internet and computer networks are essential to our daily lives and businesses. With our reliance on computers and communication networks, harmful actions have increased. Network hazards plague modern communication. To keep networks running smoothly and users' data safe, network traffic flow must be monitored for malicious activity and assaults. Denial-of-service (DoS) attacks aim to disrupt a network server, website, or web service. Computer networks and services are vulnerable to DoS and DDoS attacks. Flooding may be the simplest DDoS assault. DDoS attacks transmit massive amounts of useless data to a network or server. The study seeks to strengthen network infrastructures against various threats, maintain service continuity, and secure the network. Denial-of-service (DoS) attacks prevent legitimate users from accessing and using information systems and resources. Figure B shows DoS/DDoS attacks using ICMP, UDP, and the more prevalent TCP flood assaults. These strikes must be detected and stopped immediately. Businesses and schools went online during COVID-19. Because so much data is created and stored, traditional Machine Learning-based DoS/DDoS attack detection approaches are ineffective. This study uses SVM, MLP, and LSTM algorithms for Deep Learning. The proposed Deep Learning model learns and builds binary and multiclass classification models that can distinguish network attack activity from normal traffic. We look for outliers and attack signals in traffic patterns and data. Our deep learning model is studied with accuracy and precision. In detection, the system checks for attack or regular network data. MLP Algorithm helps this model discover items 97% of the time. LSVM ML classification compares the suggested system's performance. This paper examines traffic behavior. This study also used traffic filtering to eliminate suspicious or attack-signature traffic. Next, we limited traffic from specified sources and locations using rate-limiting. Python SCAPY and wireshark Sniffer in Linux OS capture network packet data for analysis and repair. Compared wireshark with scapy packet capturing analysis and mitigation. This study examines network DoS/DDoS assaults and their prevention. These approaches detect and mitigate flood-based DoS assaults to keep systems functioning and networks safe. To keep up with DoS assaults and the threat landscape, you must continually studying and developing new tactics.

Keywords: *Classification, Dataset, MLP,NSL-KDD, Scapy, Wireshark,*

1. Introduction

Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks are among the most significant threats to the availability and integrity of computer networks and services[35][3][38].The goal of these attacks is to flood target systems with bad data, making them inaccessible to real users. Understanding what Do's/DDoS attacks are and how they work is important for coming up with effective ways to find them and stop them [39]. Do's attacks use weaknesses in the system or network infrastructure of the target to use up all of its resources, like bandwidth, speed, processing power, or memory [40]. DoS/DDoS attacks can have big effects, like losing money, hurting your image, and stopping important services. These attacks can hurt a wide range of businesses, such as e-commerce sites, banks, government agencies, and internet service providers. Because of this, there is a pressing need for more advanced ways to find and stop Do's/DDoS attacks [1-41].The Global Connection has cut down on journey times, but it has also made our systems more open to attacks. We must protect our data [41]. Concerns about network security can be put into the three groups. These include illegal denial of service, lack of authenticity, and loss of confidentiality. Figure A) below shows it. DoS attacks come in many different forms; so many different names for "embezzlement" have been made. One all-encompassing name, "DDoS," suggests that the attack is coming from many different places that have nothing to do with each other. DoS attacks are the same thing as DDoS attacks. Floods of ICMP (Ping), TCP-SYNCH, and UDP are used in DDoS attacks [1-42], [27].

1 G.Sankara Rao , Research Scholar , JNTUK University,Kakinada,,A.P.
2 Dr P.Krishna Subbarao , Professor , CSE Dept, GVPCE(A),VSP,A.P.

** Corresponding Author Email: shankar.g.p510@gmail.com*

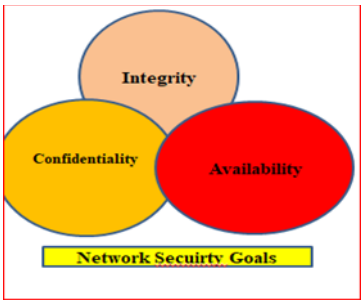


Fig A) Network security Goals/ policies

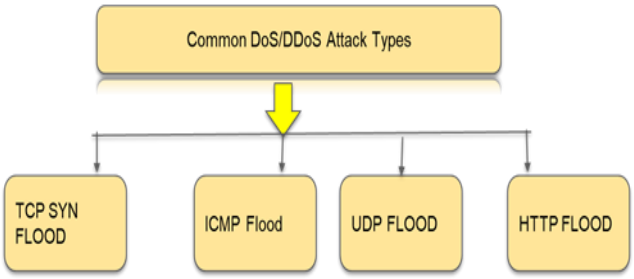


Fig B) Common DoS attacks on Availability

A. ICMP Flood Assault:

ICMP flood is a type of DDoS attack that the attacker sends too many ICMP echo requests or ping requests to a target network server or website.

B. TCP-SYN Flood Assault:

This attack could affect every device attached to the system that can connect to the internet. When a victim host replies with a SYN-ACK, the attacker keeps sending a series of SYN requests while ignoring the answer from the victim host and sending a series of SYN requests from a fake IP address. Every request from a reliable client leads to a "deny of service" message. This method keeps a host in state for fake half connections, leaving no resources for new real connections. It is shown in Figure 1) & Figure 2) below.

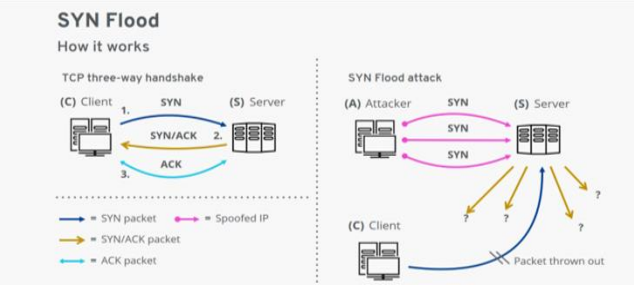


Fig. 1. Picture before SYN Flood attack happens **Fig. 2.** Picture after SYN Flood attack happens

C. UDP Flood Assault

UDP floods are a type of volumetric denial-of-service (DDoS) attack that the attacker sends IP packets, including User Datagram Protocol packets, to any specific or random port on the target. It means that any request from a legit user on the network will be dropped or take long time to get authorized. Hence creating a denial of service on the network. It's shown in the figure (3) below.

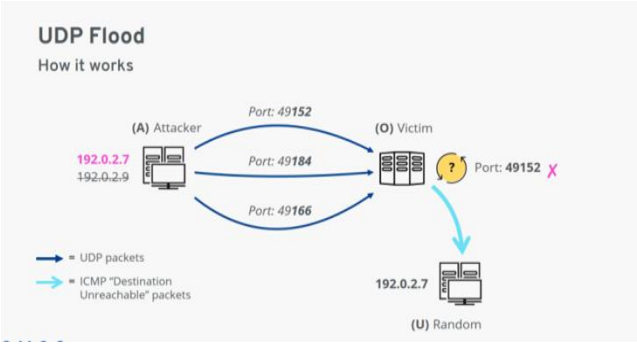


Fig. 3. UDP Flood attack happens Process

D. HTTP Flood attack

This exploit uses HTTP calls. The attacker sends the target user with an HTTP request, which the target user then runs.

Python Scapy introduction:

Python Scapy is a powerful tool and library for manipulating packets that is built on Python. It lets you make, send, receive(capture), and analyse the network packets traffic at different levels of the network stack. With Scapy, you can record and look at network data as it happens. It lets you set up packet filters based on things like source/destination IP, port, protocol, etc. Because of this, it can be used to watch networks, fix bugs, and look at security. Also, Wireshark is a well-known network protocol analyser that lets you record, inspect, and analyse network data.

2. Literature Review

Distributed Denial of Service (DDoS) attacks are a big threat to network security, and machine learning is being used more and more to find and stop them. Several studies have come up with different ways to use machine learning to spot DDoS attacks. Perez-Diaz et al. [1] came up with a flexible SDN-based design that uses machine learning to find and stop low-rate DDoS attacks. Phan and Park came up with a new way to fight against DDoS attacks in an SDN-based cloud environment by combining the hybrid machine learning model and the eHIPF scheme [2]. Dong and Sarem found the DDoS attack by using an updated version of the K-Nearest Neighbours (KNN) method based on Machine Learning (ML) [3]. Sambangi and Gondi used machine learning to find DDoS attacks with the help of multiple linear regressions [4].Basant Agarwal et al. [5] found that a combination of entropy and SVM could be used to find network abnormalities. AHAMED ALJUHANI used ML and DL in this work to put different types of DDoS attacks into groups. [6]. In terms of Precision, Recall, F1-score, and Accuracy (98.34%), YUANYUAN WEI et al.'s [7] model does better than

others. Marwane Zekri and his team [8] made a DDoS detection system based on a decision tree and C4.5. C4.5 was more accurate than other machine learning methods. ML and NN algorithms were used by Shreeekhand Wankhede et al. [9]. MLP & RF recognise datasets as benign or dos attacks. In their deep learning-based DDoS detection method, Xiaoyong et al. [10] found that DeepDefense's error rate went down by 39.69%. Mohammad Tayyab and his colleagues [11] made an IDS that uses machine learning to spot DoS and DDoS attacks. Ensemble learning and design that worked together also did their jobs. Mehdi et al. [12] used a GA and an ANN to spot a DDoS attack reliably and without doubt. Baojun et al. [13] made an Online DDoS tracking system that uses machine learning to find attacks in progress. It is based on spark streaming. We looked at Naive Bayes, Logistic Regression, and Decision Tree. Ahmad Riza'ainYuso and his colleagues [14] made a method for choosing features to improve intrusion detection systems.Obaid et al. [15] showed how J48, RF, SVM, and KNN can be used in an SDN network to find and stop DDoS attacks. Training and testing went well for J48. Pourya et al. [16] used a statistical method to find and describe DDoS attacks, and they found that C2DF is faster and more accurate than previous models. OMer ASLAN tried out many classifiers to tell the difference between DDoS activity and normal traffic and found that the method he suggested was the best [17]. Suman Nandi et al. (18) found that their combination method was better at spotting DDoS attacks than other methods. The visualisations in this study look at multidimensional data patterns, while Chunyuan WU et al. [19] looked at DDoS attacks. It helps track down DDoS attacks.Dos/DDoS attacks can be found by Francisco Sales de Lima Filho et al.'s [20] online smart detection system. A random forest tree method sorts network data into different groups and makes DR, FAR, and PREC better. Mateusz Kozlowski et al. [21] used UDP DDoS attacks to attack machine learning models with very high accuracy in traditional tests. Yuan Tao et al. Flooding DDoS attacks in local area networks were found using a method that doesn't need storage for packet processing or computer power in the router[22]. In this study by Subhashini Peneti et al. [23], he uses feature selection to make an intrusion-based system that works well. Getting rid of features makes IDS faster and uses less memory. Yalda Khosroshahi et al. [24] Use the trial dataset to test the classifier. With 0.98 precision, the model can find problems with Android devices. Bao Cui-Mei et al. came up with the idea of a hierarchical SVM-based attack detection system. [25] Using the suggested method, new attacks that haven't been seen before can be found at the first level. TaeshikShon et al. [26] say that our Enhanced SVM method was made to find and classify new attacks in network traffic. G.SankaraRao et al. [27] came up with a plan for a method to find DoS/DDoS attacks in networks. DoS (Denial of Service) and DDoS (Distributed Denial of Service) attacks are very dangerous to network security because they stop services from working and cost money [32]. With the rise of new technologies like the Internet of Things (IoT) and Software-Defined Networking (SDN), DDoS attacks have more ways to get in. This means that we need better ways to find them and stop them [29]. There have been many ideas in the writings about how to deal with the problems that DoS and DDoS attacks cause. Software-Defined Networking (SDN) is an approach that could help find and stop DoS attacks [28]. SDN provides a scalable and flexible design that can be optimised for the IoT ecosystem [28]. By using SDN, it is possible to find and stop DoS threats as soon as they happen [29]. Galeano-Brajones et al. [28] came up with an experimental way to find DoS and DDoS attacks and stop them in IoT-based stateful SDN. Their plan showed that SDN is a good way to find and stop these kinds of threats. A lot of people have also used machine learning to find DoS and DDoS attacks. The quality of the samples used to train these methods is important [30]. [30] did a thorough study of the literature and found problems with the data sets that can be used to find DDoS attacks on Vehicular Ad hoc NETwork (VANET) systems. They talked about how accurate spotting of DDoS attacks in VANETs needs datasets that are representative and real-world. Aside from machine learning, other methods of data mining have also been used to find DDoS attacks.Abubakar et al. [31] used data mining techniques, like decision trees and K-nearest neighbour algorithms, to make it easier and more accurate to identify DDoS attacks. Their research showed that data mining methods can be used to find DDoS attacks. Block chain technology has also been looked at as a way to find DDoS attacks and stop them. Chaganti et al. [29] talked about how blockchain can help stop DDoS attacks. They talked about the benefits of parties working together to find and stop DDoS attacks using blockchain technology. There have also been ideas for detecting and stopping DDoS attacks that are based on deep learning. Bousalem et al. [34] showed a way to find and stop DDoS attacks in 5G and other mobile networks by using deep learning. Their method was based on machine learning modules and showed how deep learning can be used to find and stop DDoS attacks.DoS and DDoS attacks have also been found using other methods, such as intrusion detection systems (IDSs) and anomaly detection. Using a backpropagation neural network, Khandelwal et al. [33] came up with a way to find DoS attacks. They talked about different queuing methods and how well they protect against DoS attacks.

Neural networks are suggested for classifying DDoS attack s.Selecting and preparing a representative DDoS assault da taset and creating a sequential neural network model for m ulticlass classification are their main goals.Artificial intelli gence is crucial to DDoS detection and classification, accor

ding to the report.[35].Rasheed et al. offer a machine learni ng classification technique for denial of service attack dete ction.The authors create a machine learningbased network traffic classification and DoS attack detection system.They compare machine learning classifier results to assess syste m performance.The research examines machine learning's DoS detection capabilities.[36].M. Alkasassbeh and collea gues found that data mining techniques might be used to de tect distributed denial of service attacks [37].Another work by Bonguet & Bellaiche (2017) analyzes cloud computing DoS and DDoS attacks and defenses.The authors investigat e cloud computing attacks like XML-DoS and HTTP-DoS. They also study attack detection and mitigation methods.DoS and DDoS attacks in cloud s ystems are changing, and this survey paper discusses defen se options[38].DoriguzziCorin et al. (2020) present a viabl e deep learning solution, Bonguet & Bellaiche (2017) exa mine cloud computing assaults and defenses, and Hefeeda & Habib (2011) detect DoS attacks in QoSenabled network s.These publications can help researchers and practitioners understand DoS and DDoS detection and mitigation metho ds.[39].Hefeeda & Habib (2011) also explore QoSenabled network DoS attack and service violation detection.The aut hors examine various DoS attacks and literature-based defenses. They emphasize the relevance of QoS in detecting and mitigating DoS attacks and descri be detection methods and algorithms.[40].B. Hari Krishna et al. detected intrusions using Soft Voting Classifier for N etwork Security Enhancement [41]. In short, the study of the literature shows that different methods have been suggested for finding and stopping DoS and DDoS attacks in networks and security. These methods include SDN, machine learning, data mining, bitcoin, deep learning, and intrusion detection systems. Each method has its own pros and cons, and more study is needed to find ways to find and stop DoS and DDoS attacks that are more reliable and effective. Each way to find and stop an attack has its own strengths and weaknesses. Some of the gaps in study are: 1.Improved detection methods: Current methods may not be able to find DoS/DDoS threats correctly. Explore the latest developments in machine learning, data mining, and artificial intelligence to make detection algorithms that can find new and rising attacks. 2. Strategies for stopping unknown attacks: Traditional methods rely heavily on signature-based detection, which makes them open to zero-day attacks or attacks that don't follow any known patterns. To lessen the effects of DoS/DDoS attacks, it's important to come up with proactive and flexible defences that can react to unknown attacks in real time [1-42].

3. Dataset Description

The NSL-KDD Dataset is used in this proposed method. Table 3.1) shows that the dataset has 42 features, 494021 records, and is 50MB in size. The KDD Cup 1999 dataset is updated to makethe NSL-KDD dataset. It has information about network activity, including DoS attacks and other types of attacks. It is often used to test systems that look for intrusions. Linear SVC and Multilayer perceptron algorithms MLP were used to train the suggested model. The information includes the length of each packet, the type of protocol used, the duration, service, and a label that says "Class" etc.

1	Duration	15	Su Attempted	29	Same Srv Rate
2	Protocol Type	16	Num Root	30	Diff Srv Rate
3	Service	17	Num File Creations	31	Srv Diff Host Rate
4	Flag	18	Num Shells	32	Dst Host Count
5	Src Bytes	19	Num Access Files	33	Dst Host Srv Count
6	Dst Bytes	20	Num Outbound Cmds	34	Dst Host Same Srv Rate
7	Land	21	Is Hot Logins	35	Dst Host Diff Srv Rate
8	Wrong Fragment	22	Is Guest Login	36	Dst Host Same Src Port Rate
9	Urgent	23	Count	37	Dst Host Srv Diff Host Rate
10	Hot	24	Srv Count	38	Dst Host Serror Rate
11	Num Failed Logins	25	Serror Rate	39	Dst Host Srv Serror Rate
12	Logged In	26	Srv Serror Rate	40	Dst Host Rerror Rate
13	Num Compromised	27	Rerror Rate	41	Dst Host Srv Rerror Rate
14	Root Shell	28	Srv Rerror Rate	42	Class

Table 3.1. NSL-KDD datasetits 42 features

4. Methodology

Below Figure 4.1 shows how I plan to do the work for this study. And the steps are as follows: The suggested system can tell if arrived traffic is DOS or harmless. Table 4.1) shows how well the detections were made. Proposed Framework-1 for Detection of DoS/DDoSAttackis shown in below figure 4.1).

DoS / DDoS Attack Detection Framework

4.1. Experiment#1:

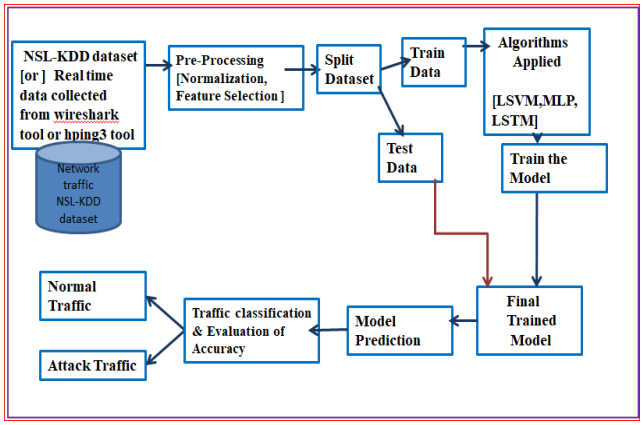


Fig. 4. Proposed Framework for DoS/DDoS attack Detection in network

Proposed Methodology Experiment #1

Experiment #1 Steps:

This novel detection framework for dos attack detection based on Deep Learning Techniques includes four steps:

Step 4.1:Dataset Collection

To get the network data flow, we use the NSL-KDD benchmark dataset, or real-time data from the Wireshark tool or the Hping3 tool. NSL-KDD is a suggested data collection that would fix some of the problems with the KDD'99 data set. Hping3 is a network tool that can send custom ICMP/UDP/TCP packets and show target answers, just like ping does with ICMP replies

Step 4.2: Data Pre-Processing:

In this step, the original data format is changed and the data values are normalised. For the DL model to work better, it needs to turn its category data into binary data. In this way of handling the data, the steps used are cleaning the Data, Normalising the Data, and Choosing the Features. In the step of pre-processing the data, the data must first be cleaned and normalised. After that, the data can be pre-processed. During pre-processing, records from the dump area or the real world that are noisy, unreliable, fragmented, missing values, numeric, or not numeric must be cleaned up. The data normalisation method is used to make a new data vector with numbers that fall within a certain range, such as scaling values between 0 and 1. Normalisation can be done in many ways, such as min-max, z-score, and decimal scale. Figure 5 screens shows the Normalisation process that has been done.

```
[17] # selecting numeric attributes/columns from dataset
     numeric_col = data.select_dtypes(include='number').columns
     numeric_col
```

```
[20] # using standard scaler/z-score for normalizing
     std_scaler = StandardScaler()
     def normalization(df,col):
       for i in col:
         arr = df[i]
         arr = np.array(arr)
         df[i] = std_scaler.fit_transform(arr.reshape(len(arr),1))
       return df
```

```
[21] # calling the normalization() function
     data = normalization(data.copy(),numeric_col) #normalization converts
```

```
[22] # data after normalization
     data.head()
```

[22] # data after normalization
data.head()

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst
0	-0.110249	tcp	ftp_data	SF	-0.007679	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076	...	-0.818890	
1	-0.110249	udp	other	SF	-0.007737	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076	...	-1.035688	
2	-0.110249	tcp	private	S0	-0.007762	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076	...	-0.809857	
3	-0.110249	tcp	http	SF	-0.007723	-0.002891	-0.014089	-0.089486	-0.007736	-0.095076	...	1.258754	
4	-0.110249	tcp	http	SF	-0.007728	-0.004814	-0.014089	-0.089486	-0.007736	-0.095076	...	1.258754	

Fig. 5. Code snippets for Normalization of dataset

Next step is Extracting Features done by using Pearson correlation coefficient (take strongly correlated features thatit has more than 0.5 as shown in below code snippet Figure 6).

```
[53] # finding the attributes which have more than 0.5 correlation
     corr= numeric_bin.corr()  # numeric_bin is Dataset which has
     corr_y = abs(corr['intrusion'])
     highest_corr = corr_y[corr_y >0.5]
     highest_corr.sort_values(ascending=True)
```

```
count                    0.576444
srv_serror_rate          0.648289
serror_rate              0.650652
dst_host_serror_rate     0.651842
dst_host_srv_serror_rate 0.654985
logged_in                0.690171
dst_host_same_srv_rate   0.693803
dst_host_srv_count       0.722535
same_srv_rate            0.751913
intrusion                1.000000
```

Fig. 6. Code snippet for Pearson correlation

Next Pie Chart distribution of dataset with normal, abnormal values of last/target column [label column]in dataset is shown in below code snippet and in Figure7)

```
[40] # pie chart distribution of normal and abnormal labels
     plt.figure(figsize=(8,8))
     plt.pie(bin_data.label.value_counts(),labels=bin_data.label.unique(),autopct='%0.2f%%')
     plt.title("Pie chart distribution of normal and abnormal labels of dataset")
     plt.legend()
     plt.show()
```

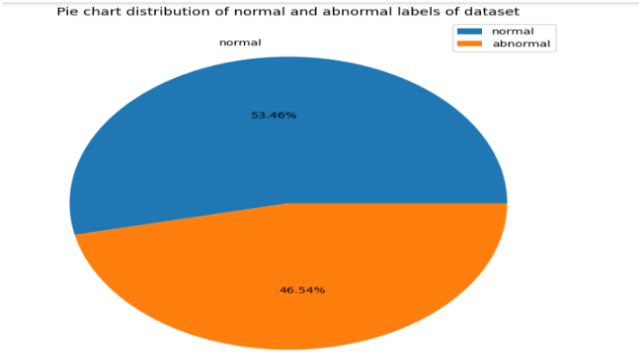


Fig. 7. Pie Chart distribution of dataset with normal, abnormal network traffic

Step 4.3: Training Data:

Then, divide the collected data into train data and test data. LSVM, MLP, and LSTM Deep Learning methods can be used to train the model with the train data.

Step 4.4:Testing:

After the pre-processing and training steps (STEPS 4.2 and 4.3), the test data should be used to check how well the MLP and LSTM models can classify. For example, if the

accuracy rate met the training requirement, the training would stop. If not, the model would restart the training step (STEP 4.3).

Step 4.5:Evaluation of Accuracy:

This step is used to judge how well the model did after it was trained. Metrics for review usually include the rate of accuracy, the rate of detection, and the rate of false alarms.

Step 4.6:Prediction:

Finally this Model classifies the netework traffic as a normal type or attack type traffic.

Metrics used for classification performance evaluation:

In this study, Below Performance evaluation Metrics [27], which are shown in Figure 8), are used to measure how well the suggested model works.

$$Accuracy = \frac{TruPsv + TruNeg}{TruPsv + TruNeg + FlsPsv + FlsNeg}$$

$$Precision = \frac{TruPsv}{TruPsv + FlsPsv}$$

$$Recall = \frac{TruPsv}{TruPsv + FlsNeg}$$

$$F1-Score = \frac{2 * precision * recall}{precision + recall}$$

Fig. 8. Performance Metrics for Model Evaluation

Experiment#1 RESULTS:

A) LSVM (Linear Support Vector Machine) Model:

```
[126] y_pred = lsvm.predict(X_test) # predicting target attribute on testing dataset
      ac = accuracy_score(y_test, y_pred)*100 # calculating accuracy of predicted data
      print("LSVM-Classifier Binary Set-Accuracy is ", ac)

      LSVM-Classifier Binary Set-Accuracy is  96.69460849685655

[127] # classification report
      print(classification_report(y_test, y_pred,target_names=le1_classes_))

                    precision    recall  f1-score   support

          abnormal       0.97      0.96      0.96     14720
            normal       0.96      0.97      0.97     16774

          accuracy                           0.97     31494
         macro avg       0.97      0.97      0.97     31494
      weighted avg       0.97      0.97      0.97     31494
```

Fig. 9. Linear SVM for binary Classification Result

B) MLP(Multi-Layer Perceptron) Model:

```
[88] # splitting the dataset 75% for training and 25% testing
     X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size=0.25, random_state=42)

[90] mlp = Sequential() # creating model layer by layer

mlp.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

[92] mlp.summary()#summary of model layers

     Model: "sequential_1"
     _________________________________________________________________
      Layer (type)                 Output Shape              Param #
     =================================================================
      dense_2 (Dense)              (None, 50)                4700

      dense_3 (Dense)              (None, 1)                 51

     =================================================================
     Total params: 4,751
     Trainable params: 4,751
     Non-trainable params: 0
     _________________________________________________________________
```

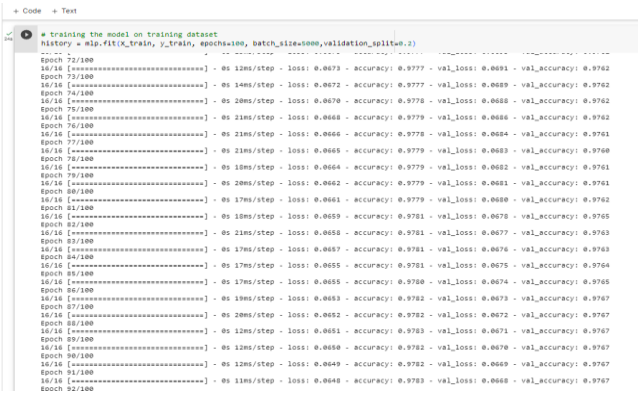


Fig. 10. Multilayer Perceptron Training Result

```
[71] # defining loss function, optimizer, metrics and then compiling model
     mlp.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

     # predicting target attribute on testing dataset
     test_results = mlp.evaluate(X_test, y_test, verbose=1)
     print(f'Test results - Loss: {test_results[0]} - Accuracy: {test_results[1]*100}')

     985/985 [==============================] - 2s 2ms/step - loss: 0.0648 - accuracy: 0.9781
     Test results - Loss: 0.06479588150978088 - Accuracy: 97.80592918395996
```

Fig. 11. Multilayer Perceptron (Model Evaluation) Result

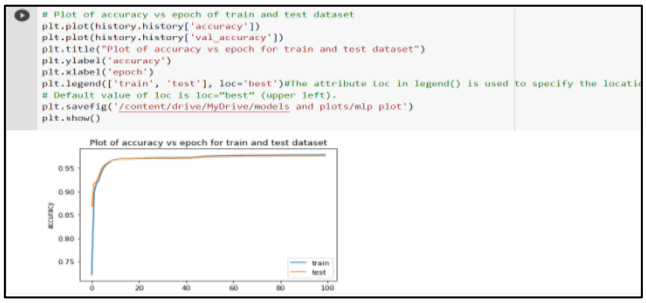


Fig 12. Plot of accuracy vs. epoch on train and test data with MLP model

```
# Plot of loss vs epoch of train and test dataset
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Plot of loss vs epoch for train and test dataset')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='best')
plt.savefig('/content/drive/MyDrive/models and plots/mlp_binary_loss.png')
plt.show()
```

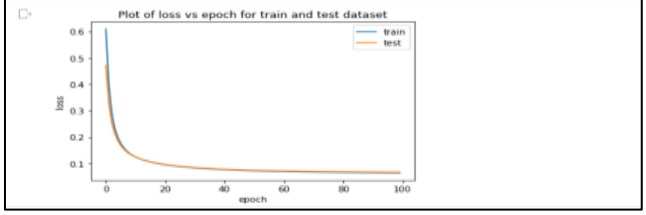


Fig. 13. Plot of loss vs. epoch on train and test data with MLP model

C) LSTM(Long Short Term Memory) Model:

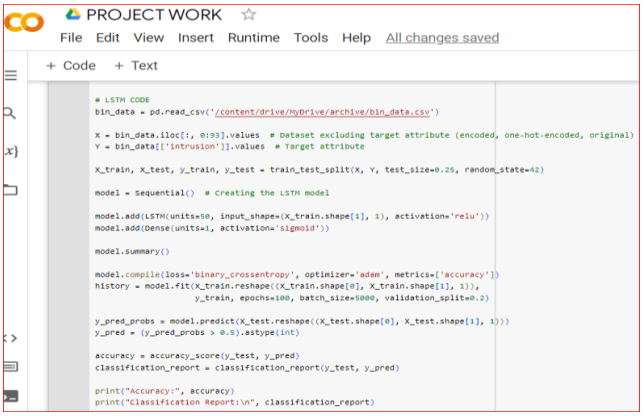
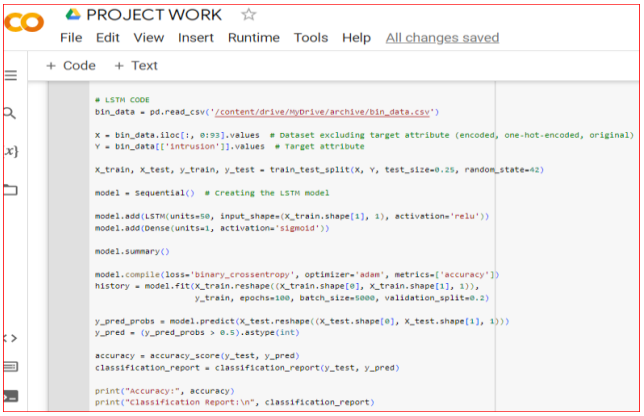
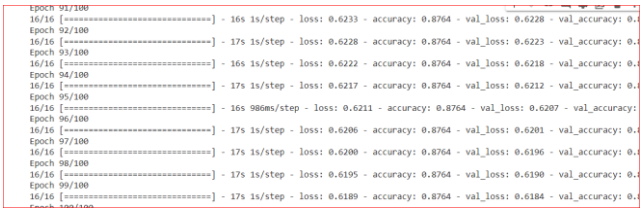


Fig. 14. LSTM Code Snippet






```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 50)                10400

 dense_2 (Dense)             (None, 1)                 51

=================================================================
Total params: 10,451
Trainable params: 10,451
Non-trainable params: 0
_________________________________________________________________
```

Fig. 15. LSTM output screens

ALGORITHM USED	Prediction Accuracy, If NSL-KDD Dataset used
LSVC	96.69%
LSTM	87.64%
Multi Layer Perceptron	97.80%

Table 4.1) Detection Accuracy Results Comparison with different Algorithms

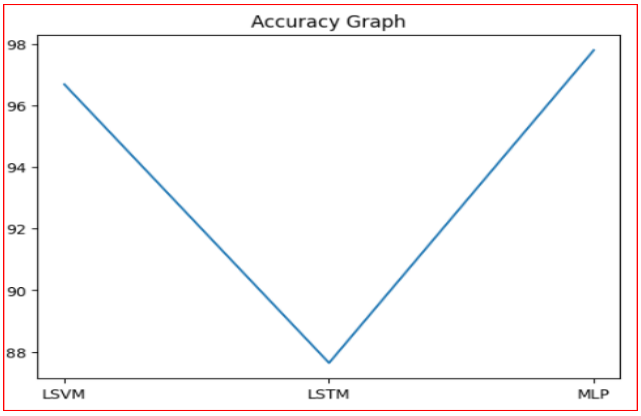


Fig. 16. Accuracy Graph for LSVM, LSTM, MLP Algorithms

5. Experiment#1 Results Discussion:

ML and DL classification methods are used to compare the performance of the proposed system. The results of how well Machine Learning LSVM and Deep Learning MLP, LSTM models can predict a DOS/DDOS attack are shown in Table 4.1). Figure 9) shows the efficiency of the Linear SVM model for classification. Figures 10), 11), 12), 13), 14), and 15) shows the evaluation results for the Deep Learning Models such as Multi-Layer Perceptron (MLP) model and the LSTM (Long Short-Term Memory) Model. Figure 16) is a graph that shows how accurate the LSVM, MLP, and LSTM models are. In the next section we will discuss dos/ddos attack mitigation techniques.

DoS/DDoS Attack Mitigation Approaches

In this section, we elaborated the approaches employed for mitigating DoS/DDoS attacks. Traffic filtering mechanisms were implemented in order to prohibit or restrict the flow of traffic originating from sources that are deemed questionable. Additionally, a rate-limiting strategy was employed to impose restrictions on the influx of packet traffic from particular sources or towards specific destinations in network.

The network has been safeguarded to some extent against DoS/DDoS assaults through the implementation of the following methods:

1)The configuration of firewall rules to restrict access from IP addresses which exhibiting suspicious behaviour in network.

2)The second approach involves implementing a Rate Limiter mechanism to restrict the quantity of IP Requests that can be transmitted within a specific time frame to destination.

One possible solution for TCP/UP/ICMP Flood is blocking IPs that sends too many packet requests to Target server. The plan for the framework is shown in Figure 17). And the steps are as follows: The suggested system reduces the impact of DoS and DDoS attacks on networks.

EXPERIMENT #1:Mitigation Methodology :

Framework for DoS/DDoS Attack Mitigation using Firewall Rules-set configuration Approach:

We created a defense script in Python that use Scapy functionalities on Linux Server, this script count the number of ICMP packets for each IP that sent packets to Server for a specified time, after getting count it is compared with our threshold/limit value for each IP and if number of packets are greater than limit, defense script will ban IP (introduces a DROP rule in iptables). The Proposed Framework is depicted in below Figure 17).For this attack we will use these three steps to implement the methodology

The Wireshark utility is a monitoring tool that allows for the observation of packets flow

being transmitted and received between an Attacker PC and a Target Server PC.

The Scapy tool is utilized on the client side, specifically by the attacker, to generate ICMP packets containing the IP address of the victim. These packets are subsequently flooded to the target server.The proposed solution entails the development of a defensive mechanism script in the

Python programming language, by utilizing the Scapy Tool. This script will be implemented on a server side and its purpose will be to identify and subsequently block or ban suspicious IP address if it detects an excessive number of requests originating from said IP address.

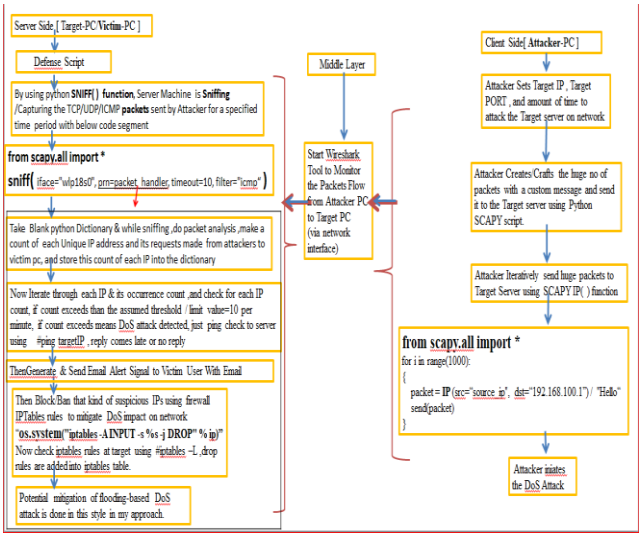


Fig. 17. Proposed Novel Framework for mitigating DoS/DDoS Attack impact

EXPERIMENT#1 STEPS for Attack Mitigation:

The proposed framework-1 for mitigation of flooding based dos/ddos attack impact steps are as follows. First run Linux Terminal on Attacker PC and open Python Scapy tool as Super user.

$ sudo scapy

Attacker(client)Side Steps:

Step 1: Attacker sets the Target IP, Target PORT, and amount of time to hit the Target server on the network.

Step 2: The next step is for the attacker to make a huge number of IP/ICMP packets with a unique message and hit the target server using a Python SCAPY script.

Step 3: Using the below code, an attacker sends a flood of large packets to hit the target server using the SCAPY IP () method.

```
from scapy.all import *
for i in range(1000):
{
packet = IP (src="source_ip", dst="192.168.100.1") / "Hello"
send(packet)
}
```

It means that the attack is started by the attacker(client). Figure 18) shows the script / code snippet that the attacker used to launch the DOS/DDOS attack.

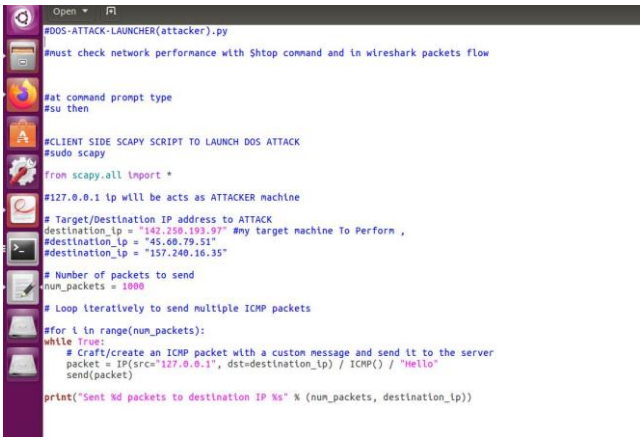


Fig. 18. Attacker's Python Script attack.py to iniate DoS attack

In Next step, on Attacker's computer, open another Linux Terminal and open the wireshark tool as Super user wth cmd

$ sudo wireshark

This tool was used to keep an eye on the packets flow analysis moving from the attacker's computer to the target's computer via the selected network interface. It's shown in the figure below 19)

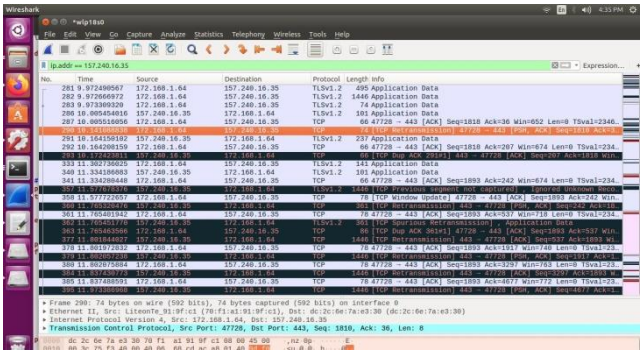


Fig. 19. packets flow between attacker and Target PC, its observation by using wireshark Tool.

Server Sider Steps : i.e (Target Server (Victim Server)):

Open Another Linux Terminal, run cmd

$ sudo python Dekstop/defense.py

Defense Script at Server Side steps are described below:

Step 1:Using the python SNIFF () function, the target server is sniffing and capturing any attacker's(client's) TCP/UDP/ICMP messages so they can be analyzed. The below code shows how to do this.

from scapy.all import *

sniff(iface="wlp18s0", prn=packet_handler, timeout=10, filter="icmp or udp or tcp")

Step 2: while listening ,packet_handler() function activated and look at the each packet, Here take a blank Python dictionary and keep track of how many times each unique IP address made an IP request from each attacker(source) to hit the Target computer. Then, store this count values for each IP into the dictionary **ip_counts[ ].**

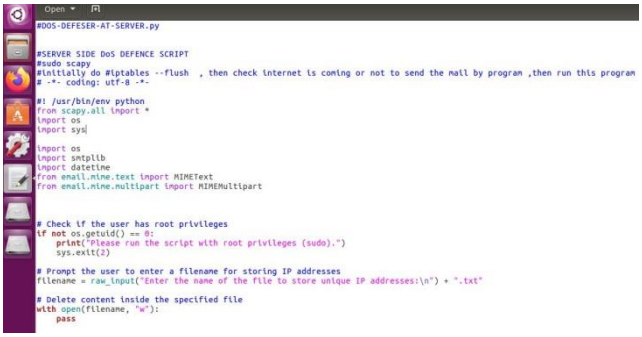
Step 3: Now, Iterate through each attacker(source) IP address and the number of times it has been requested to hit the target. For each IP address, check to see if its requests count is exceeds than the supposed threshold or limit value of 10reqeusts allowed per minute. If it is crossed that means a DoS attack has been found. Store these attackers' IP addresses into a text file attackersIPsfile1.txt and later open this text file to see suspicious/attackers ip addresses, which are blocked to prevent further Damage.

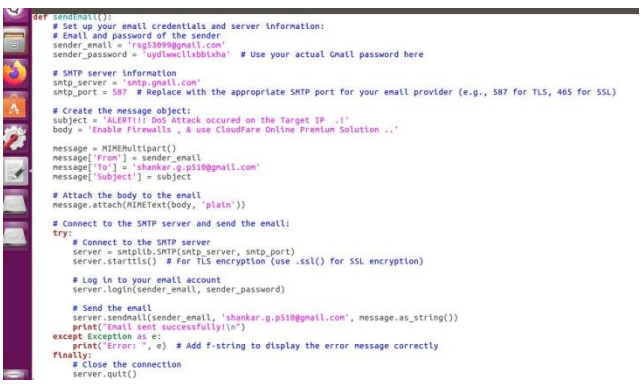
Step 4: Then Generate &trigger Dos Attack Email Alert Signal to Victim User.

Step 5: Then, it uses firewall IP-Tables rules to block or ban those suspicious IPs if it thinks it is getting too many requests from that suspected IPs. This is done to reduce the effect of DoS on the network using "os.system("iptables -A INPUT -s %s -j DROP" % ip)"command. Now check iptables rules at target using #iptables –L cmd ,drop rules are added&visible in iptables table.

Step 6: Potential mitigation of flooding-based DoS attack is done in this style using my approach.

The Code snippet(Defense Script) used at the server is depicted in below figure 20)





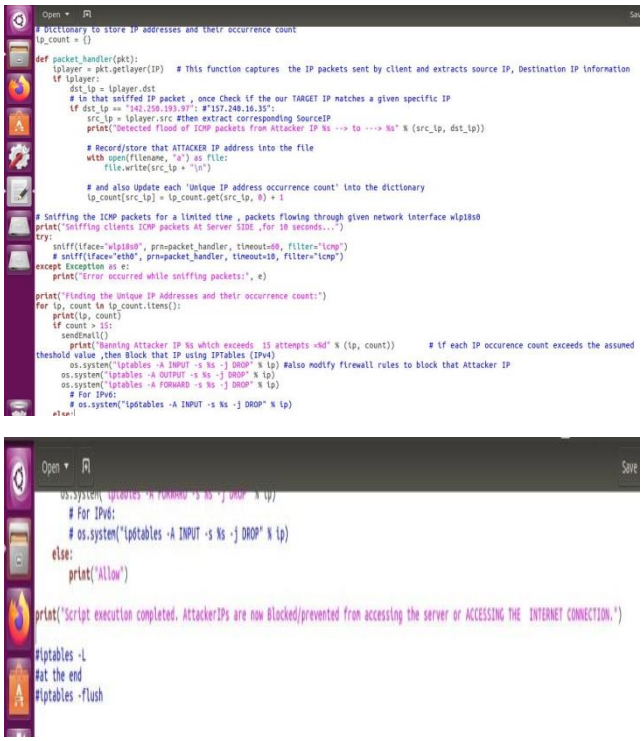


Fig. 20. Server side (Defense script) Code snippet defence.py

We can open iptables on Server PC and we can see that it have DROP rule added for Victim IP packets , just check with command

$ sudo iptables –L

Now we sent ping request to Victim Server PC from command line on Victim PC for verification:

$ ping 142.250.197.93

If no responses from Target Server ,it means that the DoS Attack succeeds!

There **are some configurations to make here** in the above Defense script: defense.py

-dst_ip should be set to our target server's IP address in the packet_handler variable dst_ip.

-iface stands for "currently using network interface" in the sniff function.

- In the sniff ( ) method, alsothe variable timeout sets a limit on how long it takes to count how many packets each attacker IP sends to hit the target.

- In the sniff function, the variable filter tells the script what kind of packets to count (in our case, TCP, UDP, or ICMP).

-In this script we set a limit of 15IP/ICMP packets in 10 seconds from the same IP.

-Also configure the Gmail to Trigger Email alert signal to victim user.

EXPERIMENT #1Results:

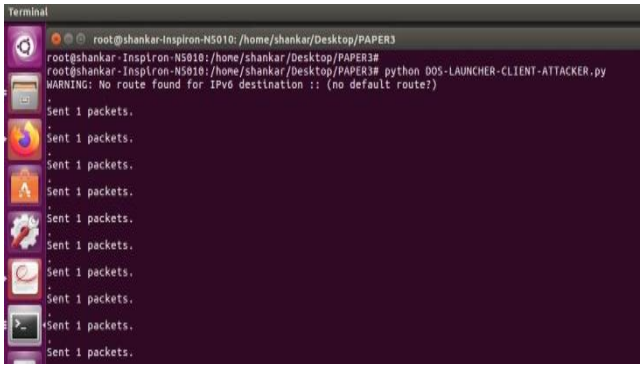
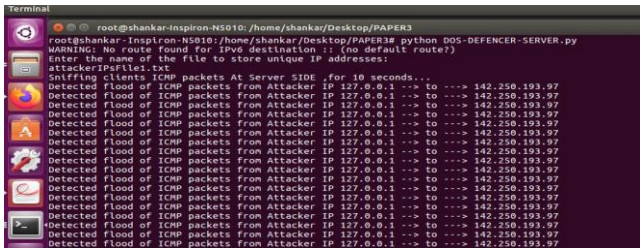


Fig. 21. Client (Attacker) Side Script to Launch the DoS Attack



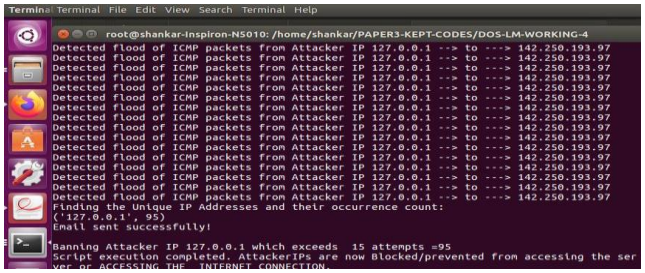


Fig. 22. Target(victim) Server Side Defense Script to detect and Ban the Suspicious IPs

Fig .23. Generated attack email alert signal to victim user

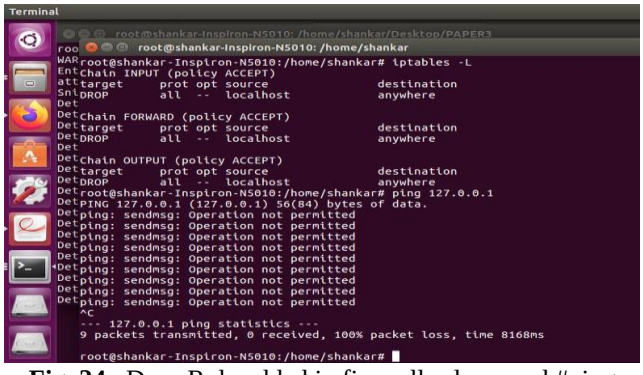


Fig. 24. Drop Rule added in firewall rules , and #ping TargetIP command showing rejection of request

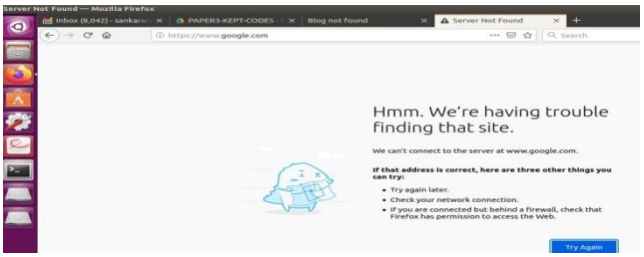


Fig.25. victim user unable to access internet connection after blocking its IPAddress

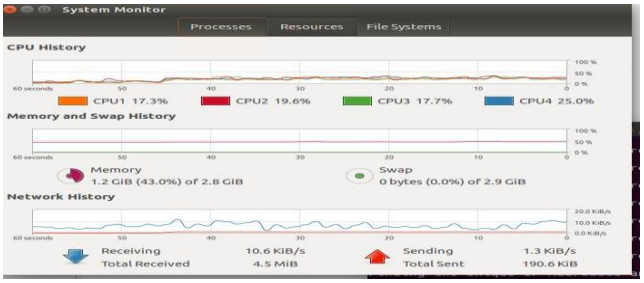


Fig 25.1) Network performance during DoSattack perform

Fig. 26. Attackers-IPAddreses listed in this text file

Experiment#1Results Discussion:

Client (Attacker) Side Script is used to Launch the DoS Attack is depicted in above Figure 21)

packets flow between attacker and Target PC, its monitoring by using wireshark Tool is depicted in above Figure 19) ,Target(victim)Server Side Defense Script is used to detect and Ban the Suspicious IPs is depicted in above Figure 22),Then Generated attack email alert signal to victim user is depicted in Figure 23),Drop Rule added in firewall rules and #ping Target command showing rejection of requests is shown in Figure 24), once attack success victim user unable to access internet connection after blocking its IP, it is shown in Figure 25).Figure 25.1) shows the cpu utilization and memory consumption ,network performance during attack. The spikes are visible during the attack but before attack these spikes were linear. Attackers-IP Addreses(Blocked IPaddresses)are listed in this text file and open it to see all the suspicious ip addresses it is shown in above figure 26).The effects of this Experiment#1 script is to stop flooding-based DOS/DDOS attacks are shown in the above figures 21–27 above. So, the suggested method of setting up firewall rules with the assumed threshold can successfully reduce the impact of DoS/DDoS attacks on networks. This reduction helps protect networks from attacks to reduce the further damage and makes the networks safer and more secure.

EXPERIMENT#2 :

Proposed DoS/DDoS Attack Mitigation Framework-2 with firewall Rules Configuration Approach [Experiment done With Wireshark Tool]:

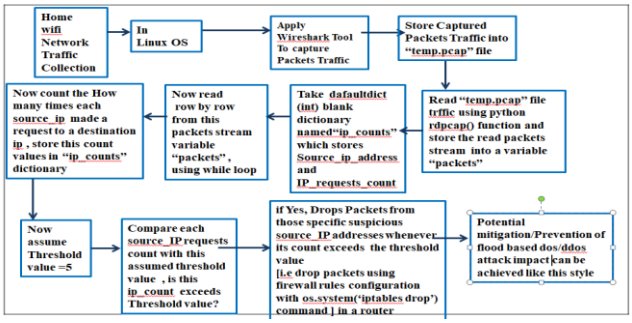


Fig. 27. Framework - 2 for Proposed Methodology [with Firewall rules configuration approach] to mitigate DoS attack

EXPERIMENT #2 STEPS For Attack Mitigation:

The framework-2 for mitigation of flooding based dos/ddos attack steps are as follows.

Step 4.1: Data Collection:

In this step, "Taken Wifi Network of Home," I used the wireshark sniffer tool to capture the live streaming of packets flow on the specific network interfacein Linux operating system platform. Now, save that stream of packets into a "temp.pcap" file, as shown in figure 20).

Step 4.2: Reading pcap file and processing packets

In this step, use the python rdpcap () function to read the temp.pcap file and iteratively read the captured packets, packet by packet.

Step 4.3: Counting How many many times each source_ip made a request to destination

In this step, take the defaultdict (int) of Python dictionary and name it as "ip_counts" dictionary. This is a blank dictionary that saves the each unique source_ip addresses and the number of times eachipmade requests to hit target from the source address.

Step 4.5:

In this step, assumed that the threshold / limit value is = 5. Compare the number of IP requests counted for each IP address with the threshold / limit value, which is supposed to be 5. Does this number exceed the threshold? If Yes, Drop Packets /Block or ban traffic fromthose suspicious IP addresses if their count exceeds the threshold value [i.e. using firewall rules setup with os.system ('iptables drop') rule] in a router. Blocking IPs that look suspicious.

Step 4.6: Potential mitigation of flood based dos/ddos attack can be achieved like this style.

EXPERIMENT#2 RESULTS :

This Experiment-2 is done to reduce the effects of a DoS attack. The results of putting it into action are shown in below Figures 21), 22), 23), 24), 25), 26), 27), 28), 29), 30), and 31).

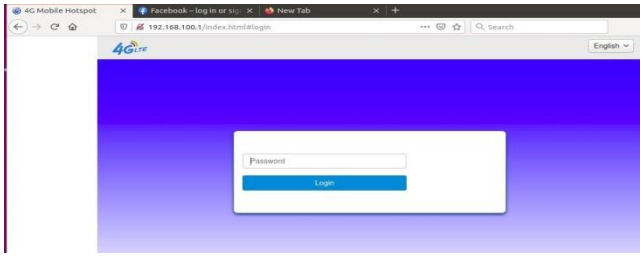


Fig. 28. Before Login, The user interface of My Home Wifi Network Router [192.168.100.1

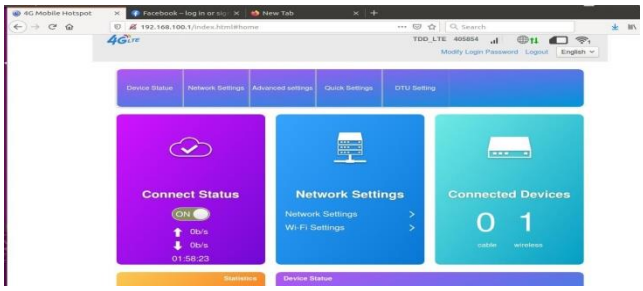


Fig. 29. After Login, the user interface of My Home Wifi Network Router [192.168.100.1]

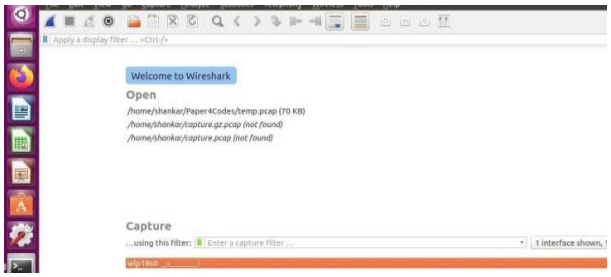


Fig. 30. Before capturing packets stream, The WiresharkGUI interface

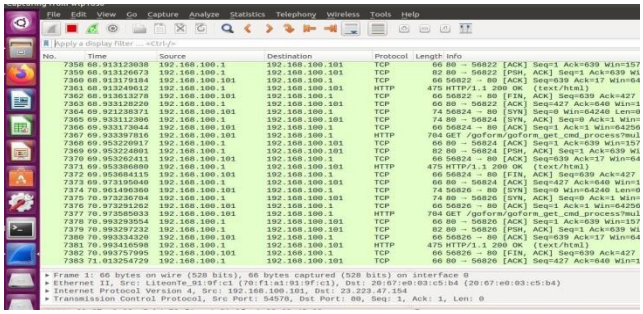
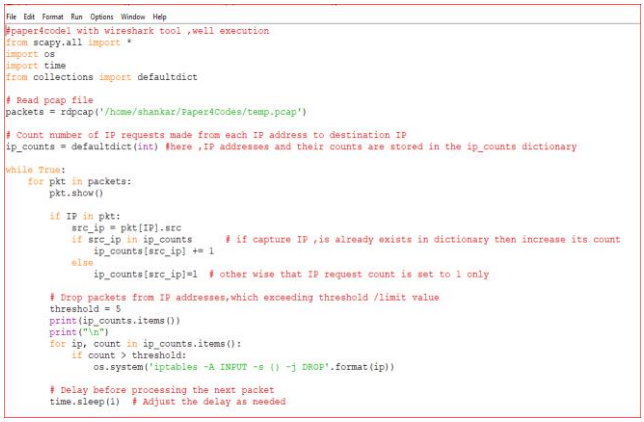
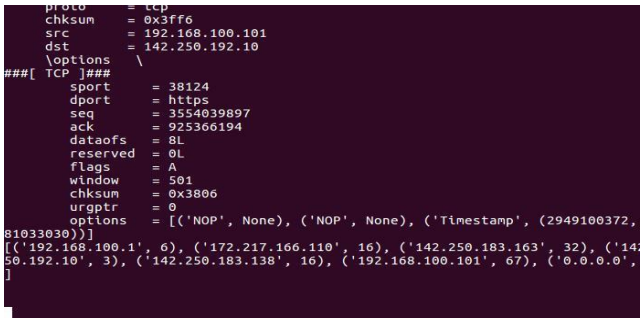


Fig. 31. The packets flow Results of Wireshark sniffer tool , after capturing network packets stream from interface



```
#paper4codel with wireshark tool ,well execution
from scapy.all import *
import os
import time
from collections import defaultdict

# Read pcap file
packets = rdpcap('/home/shankar/Paper4Codes/temp.pcap')

# Count number of IP requests made from each IP address to destination IP
ip_counts = defaultdict(int) #here ,IP addresses and their counts are stored in the ip_counts dictionary

while True:
    for pkt in packets:
        pkt.show()

        if IP in pkt:
            src_ip = pkt[IP].src
            if src_ip in ip_counts     # if capture IP ,is already exists in dictionary then increase its count
                ip_counts[src_ip] += 1
            else
                ip_counts[src_ip]=1  # other wise that IP request count is set to 1 only

        # Drop packets from IP addresses,which exceeding threshold /limit value
        threshold = 5
        print(ip_counts.items())
        print("\n")
        for ip, count in ip_counts.items():
            if count > threshold:
                os.system('iptables -A INPUT -s {} -j DROP'.format(ip))

        # Delay before processing the next packet
        time.sleep(1)  # Adjust the delay as needed
```

Fig. 32. code snippet for filtering/dropping the packets traffic from the specific ip addresses which exceeds assumed threshold value=5, filtering is done with os.system (‘iptables –A INPUT –s { } –j DROP’) command



```
    chksum    = 0x3ff6
    src       = 192.168.100.101
    dst       = 142.250.192.10
    \options   \
###[ TCP ]###
       sport     = 38124
       dport     = https
       seq       = 3554039897
       ack       = 925366194
       dataofs   = 8L
       reserved  = 0L
       flags     = A
       window    = 501
       chksum    = 0x3806
       urgptr    = 0
       options   = [('NOP', None), ('NOP', None), ('Timestamp', (2949100372,
81033030))]
[('192.168.100.1', 6), ('172.217.166.110', 16), ('142.250.183.163', 32), ('14
50.192.10', 3), ('142.250.183.138', 16), ('192.168.100.101', 67), ('0.0.0.0',
]
```

Fig. 33. The results of IP Layer Packet Structureand Sourceip addresses and its requests count Stored into python dictionary .ip_counts dictionary values[each ip_address& its requests count]

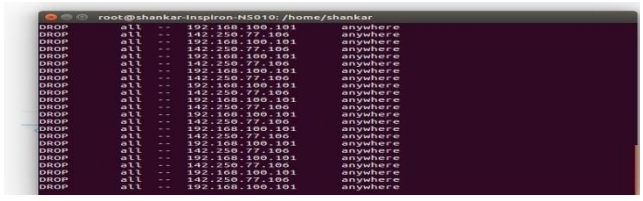


Fig. 34. the results of Firewall rules added by running above Script, see with #iptables –L command

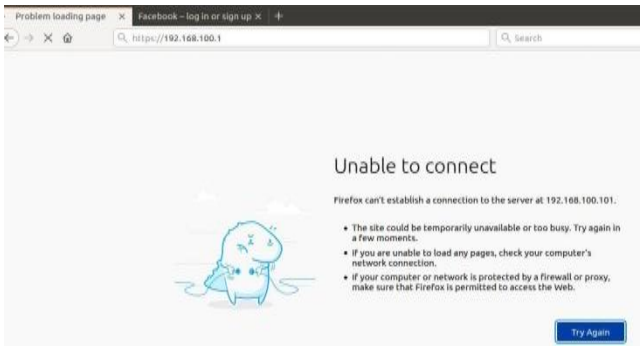


Fig. 35. the results of network disconnection in router login page, if firewalls rules changed

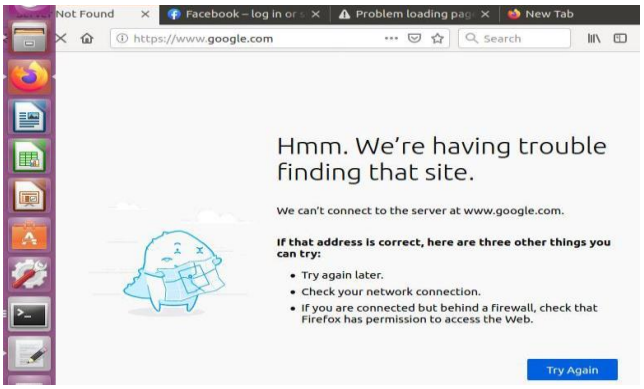


Fig. 36. the results of network disconnection google.com page, if firewalls rules changed

Experiment#2Results Discussion:

The effects of this Experiment#2 script is to stop flooding-based DOS/DDOS attacks are shown in the above figures 28–37 above. So, the suggested method of configuring firewall rules with the assumed threshold can successfully reduce the impact of DoS/DDoS attacks on networks. This reduction helps protect networks from attacks and makes the networks safer and more secure.

EXPERIMENT #3 :

DoS/DDoS Attack Mitigation Framework-3with Flask Rate Limiter Technique:

A rate limiter performs the function of a traffic police by imposing predefined limitations on the number of times a request can be made from a single source. The proposed framework-3 for mitigating dos attack impact is depicted in below Figure 38) and the are steps sexplained as follows

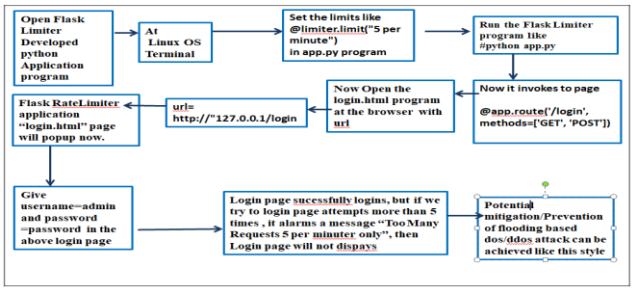


Fig. 37. Framework – 3 for Proposed Methodology [with flask Rate limiter technique]

EXPERIMENT #3Steps:

Step 7.1:Open the rate limiter python application script developed using Python Flask in Linux OS as shown in above figure 40)

Step 7.2: set the rate limits (ip packet requests) like @limiter. Limit ("5 requests per minute")

Step 7.3: Run the flask limiter Script using #python app.py

Step 7.4: now it triggers/navigates to page @app.route ('/login')

Step 7.5: now open the login.html page with the url= http://127.0.0.1/login

Step 7.6: now login.html interface page will popup here enter the username =admin and password=password

Step 7.7: login page is successful, now try to do more login attempts, which sends excessive traffic to hit the Target Server. It leads to page disconnection Due to rate limiter setting enabled.

Step 8.8: in thus way potential mitigation/prevention of dos attack impact can be achieved using my approach.

EXPERIMENT #3 Results:

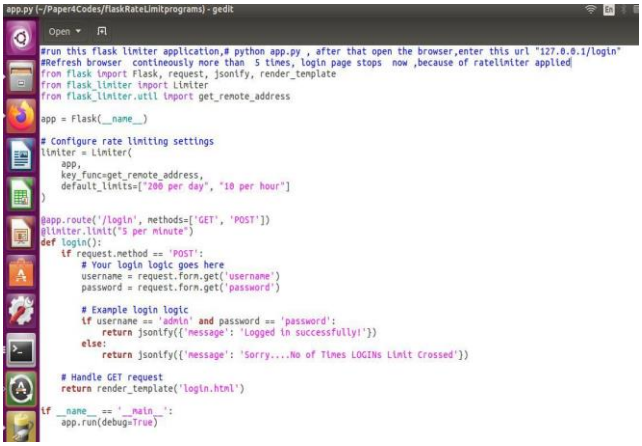


Fig. 39. Rate Limiter Flask Program code snippet [app.py] /script

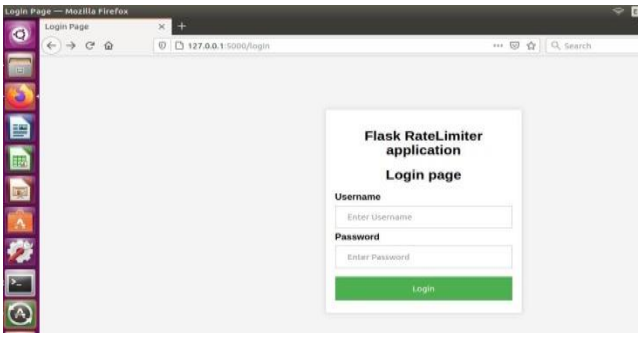


Fig. 42. Login.html page user interface

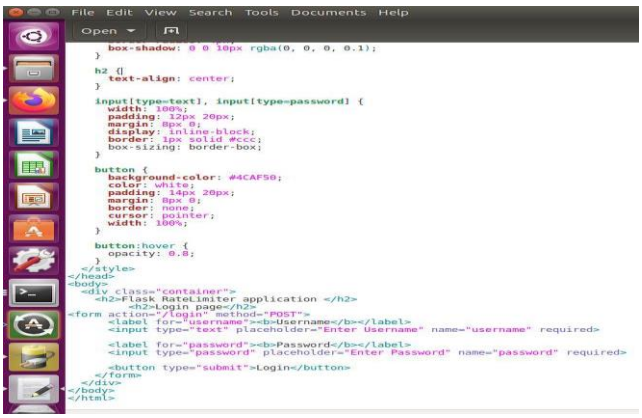


Fig. 40. login.html page code snippet

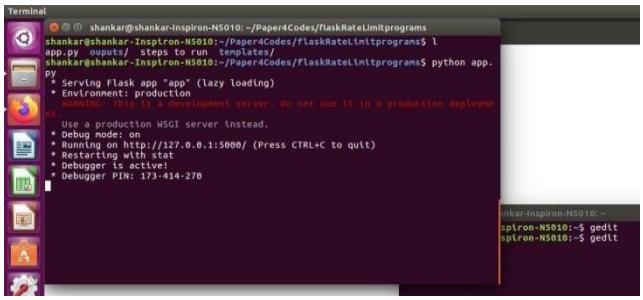


Fig. 41. Flask application 'app.py' is running screen

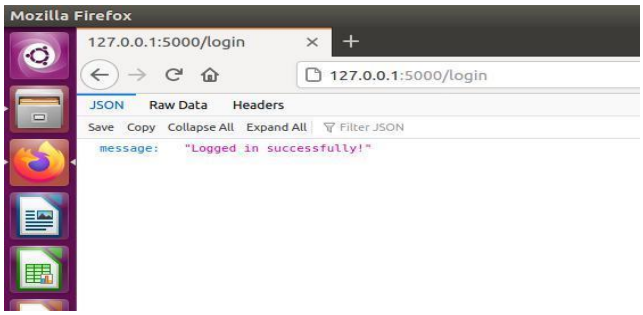


Fig. 42. user interface After Login page success

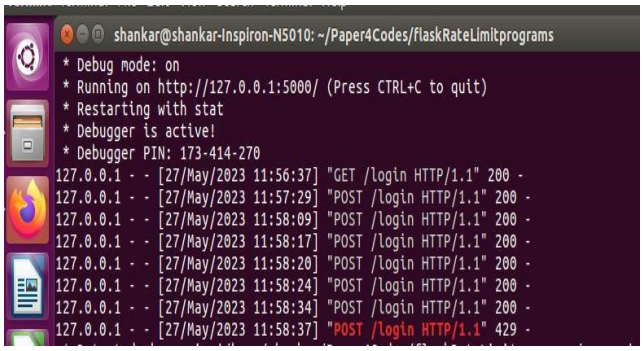


Fig. 43. Login page attempts exceeds more than 5 times

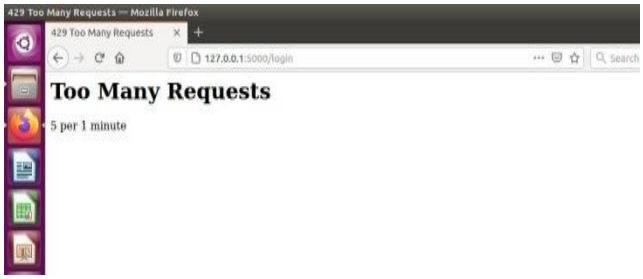


Fig. 44. Result is,if too many IP Requests from attacker[more than 5(limit)requests],it is not allowed to this target server

Experiment #3 Results Discussion:

The effects of this Experiment#3 script is to stop flooding-based DOS/DDOS attacks are shown in the above figures 39–44 above. So, the suggested method of setting up rate limiting approach can successfully reduce the impact of DoS/DDoS attacks on networks. This reduction helps protect networks from attacks and makes them safer and more secure.

EXPERIMENT #4

DoS/DDoS Attack Mitigation Framework-4 using Honeypot Approach:

Honeypots are used for a number of reasons, the key one is being to acquire useful information on the strategies, procedures, and methods that the attackers employ, as well as to divert their attention away from the production system itself .Logging is performed in order to record every interaction that takes place within the honeypot. It is important to gather information such as IP addresses, commands, payloads, and timestamps and close the attacker connection. Its process is shown in below Code Snippet Figure 45).

```
#HONEYPOT.py [ At serverside ]

##      Honeypot finds the attackers,LOGs the attacker's suspicious activity and closes the attacker connections to target
import socket
import threading

def honeypot_listener():

    # Honeypot configuration
    honeypot_ip = "0.0.0.0"  # Listen on all available interfaces
    honeypot_port = 80       # Port to listen on


    # Create a socket for the honeypot

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket:
        server_socket.bind((honeypot_ip, honeypot_port))
        server_socket.listen()

        print(f"Honeypot listening on {honeypot_ip}:{honeypot_port}")

        while True:
            client_socket, client_address = server_socket.accept()  # attackers(client) connections accepted
            print(f"Connection from: {client_address}")

            # Log and handle incoming requests (you can customize this part)
            data = client_socket.recv(1024)     #Honeypot Logs the data ,if any data received from attackers(clients),
            if data:
                print(f"Received data is := {data}")

            # Closes the connection from attacker IP Address
            client_socket.close()

# Start the honeypot listener in a separate thread

honeypot_thread = threading.Thread(target=honeypot_listener)
honeypot_thread.start()

# Wait for the honeypot thread to finish
honeypot_thread.join()
```

Fig. 45. Honeypot(Honeytrap) approach to closes the attacker connections to targetIP

Experiment#4 Result Discussion: Here this Honeypot algorithm which logs all the data received from attackers and closes the attackers socket connections if any suspicious activity found.

6. Conclusion

DoS/DDOS type attacks have become more common in spread systems like the internet and social media networks. It is important to know what kinds of attacks cause DoS and DDoS attacks, which stop computers from working. Deep Learning models can be used to find DoS attacks detection. These DL models can be used to train and test the network data and find these kinds of attacks. MLP model gives a prediction accuracy of 97.80% when our recommended model is used. The suggested method identifies DoS and DDoSattacks well enough to protect computer networks better. To make this study even better, the effects of DoS attacks on networks should be lessened. And it is important to find ways to stop DoS and DDoS attacks, which stop computers from working. To protect against DoS/DDoS attacks, you can set up packet filtering technique and use rate-limiting methods or honeypot approach. DoS and DDoS attacks have less of an effect on computer networks when the suggested method is used.

The DoS/DDoS mitigation methods we've talked about will definitely help any network protect itself from a DoS/DDoS attack which can reduce the further damage of networks. The plan for the future of this study is to focus on big networks that use different security protocols and ways to stop these attacks.

References

[1] Perez-Diaz, Valdovinos, Choo, Zhu (2020). A Flexible SDN-Based Architecture for Identifying and Mitigating Low-Rate DDoS Attacks Using Machine Learning. IEEE Access, (8), 155859-155872. https://doi.org/10.1109/access.2020.3019330

[2] Phan, Park (2019). Efficient Distributed Denial-of-Service Attack Defense in SDN-Based Cloud. IEEE Access, (7), 18701-18714. https://doi.org/10.1109/access.2019.2896783

[3] [Dong, Sarem (2020). DDoS Attack Detection Method Based on Improved KNN With the Degree of DDoS Attack in Software-Defined Networks. IEEE Access, (8), 5039-5048. https://doi.org/10.1109/access.2019.2963077

[4] Sambangi, Gondi (2020). A Machine Learning Approach for DDoS (Distributed Denial of Service) Attack Detection Using Multiple Linear Regression.. https://doi.org/10.3390/proceedings2020063051

[5] B. Agarwal and N. Mittal, "Hybrid Approach for Detection of Anomaly Network Traffic using Data Mining Techniques," Procedia Technol., vol. 6, pp. 996–1003, 2012, doi: 10.1016/j.protcy.2012.10.121.

[6] A. Aljuhani, "Machine Learning Approaches for Combating Distributed Denial of Service Attacks in Modern Networking Environments," IEEE Access, vol. 9, pp. 42236–42264, 2021, doi: 10.1109/ACCESS.2021.3062909.

[7] Y. Wei, J. Jang-Jaccard, F. Sabrina, A. Singh, W. Xu, and S. Camtepe, "AE-MLP: A Hybrid Deep Learning Approach for DDoS Detection and Classification," IEEE Access, vol. 9, pp. 146810–146821, 2021, doi: 10.1109/ACCESS.2021.3123791.

[8] M. Zekri, S. El Kafhali, N. Aboutabit, and Y. Saudi, "DDoS attack detection using machine learning techniques in cloud computing environments," Proc. 2017 Int. Conf. Cloud Comput. Technol. Appl. CloudTech 2017, vol. 2018-Janua, pp. 1–7, 2018, doi: 10.1109/CloudTech.2017.8284731.

[9] Wankhede and D. Kshirsagar, "DoS Attack Detection Using Machine Learning and Neural Network," Proc. - 2018 4th Int. Conf. Comput.Commun. Control Autom. ICCUBEA 2018, 2018, doi: 10.1109/ICCUBEA.2018.8697702.

[10] X. Yuan, C. Li, and X. Li, "DeepDefense: Identifying DDoS Attack via Deep Learning," 2017 IEEE Int. Conf. Smart Comput. SMARTCOMP 2017, pp. 1–8, 2017, doi: 10.1109/SMARTCOMP.2017.7946998.

[11] M. Tayyab, B. Belaton, and M. Anbar, "ICMPV6-based DOS and DDoS attacks detection using machine learning techniques, open challenges, and blockchain applicability: A review," IEEE Access, vol. 8, pp. 170529–170547, 2020, doi: 10.1109/ACCESS.2020.3022963.

[12] M. Barati, A. Abdullah, N. I. Udzir, and ..., "Distributed Denial of Service detection using hybrid machine learning technique," Biometrics …, pp. 268–273, 2014, [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7013133/

[13] B. Zhou, J. Li, J. Wu, S. Guo, Y. Gu, and Z. Li, "Machine-learning-based online distributed denial-of-service attack detection using spark streaming," IEEE Int. Conf. Commun., vol. 2018-May, 2018, doi: 10.1109/ICC.2018.8422327.

[14] A. R. A. Yusof, N. I. Udzir, A. Selamat, H. Hamdan, and M. T. Abdullah, "Adaptive feature selection for denial of services (DoS) attack," 2017 IEEE Conf. Appl. Inf. Netw. Secur. AINS 2017, vol. 2018-Janua, pp. 81–84, 2017, doi: 10.1109/AINS.2017.8270429.

[15] O. Rahman, M. A. G. Quraishi, and C. H. Lung, "DDoS attacks detection and mitigation in SDN using machine learning," Proc. - 2019 IEEE World Congr. Serv. Serv. 2019, vol. 2642–939X, pp. 184–189, 2019, doi: 10.1109/SERVICES.2019.00051.

[16] P. Shamsolmoali and M. Zareapoor, "Statistical-based filtering system against DDOS attacks in cloud computing," Proc. 2014 Int. Conf. Adv. Comput.Commun. Informatics, ICACCI 2014, pp. 1234–1239, 2014, doi: 10.1109/ICACCI.2014.6968282.

[17] Ö. ASLAN, "A Methodology to Detect Distributed Denial of Service Attacks," BilişimTeknol.Derg., vol. 15, no. 2, pp. 149–158, 2022, doi: 10.17671/gazibtd.1002178.

[18] S. Nandi, S. Phadikar, and K. Majumder, "Detection of DDoS Attack and Classification Using a Hybrid Approach," ISEA-ISAP 2020 - Proc. 3rd ISEA Int. Conf. Secur. Priv. 2020, pp. 41–47, 2020, doi: 10.1109/ISEA-ISAP49340.2020.234999.

[19] S. Sheng, C. Wu, and X. Dong, "Research on Visualization Systems for DDoS Attack Detection," Proc. - 2018 IEEE Int. Conf. Syst. Man, Cybern. SMC 2018, pp. 2986–2991, 2019, doi:

10.1109/SMC.2018.00507.

[20] F. S. De Lima Filho, F. A. F. Silveira, A. De Medeiros Brito Junior, G. Vargas-Solar, and L. F. Silveira, “Smart Detection: An Online Approach for DoS/DDoS Attack Detection Using Machine Learning,” Secur. Commun. Networks, vol. 2019, 2019, doi: 10.1155/2019/1574749.

[21] M. Kozlowski and B. Ksiezopolski, “A new method of testing machine learning models of detection for targeted DDoS attacks,” Proc. 18th Int. Conf. Secur.Cryptogr.SECRYPT 2021, no.Secrypt, pp. 728–733, 2021, doi: 10.5220/0010574507280733.

[22] Y. Tao and S. Yu, “DDoS attack detection at local area networks using information theoretical metrics,” Proc. - 12th IEEE Int. Conf. Trust.Secur. Priv. Comput. Commun.Trust. 2013, pp. 233–240, 2013, doi: 10.1109/TrustCom.2013.32.

[23] S. Peneti and Hemalatha, “DDOS Attack Identification using Machine Learning Techniques,” 2021 Int. Conf. Comput.Commun. Informatics, ICCCI 2021, 2021, doi: 10.1109/ICCCI50826.2021.9402441.

[24] Y. Khosroshahi and E. Ozdemir, “Detection of Sources Being Used in DDoS Attacks,” Proc. - 6th IEEE Int. Conf. Cyber Secur. Cloud Comput. CSCloud 2019 5th IEEE Int. Conf. Edge Comput. Scalable Cloud, EdgeCom 2019, pp. 163–168, 2019, doi: 10.1109/CSCloud/EdgeCom.2019.000-1.

[25] C. M. Bao, “Intrusion detection based on one-class SVM and SNMP MIB data,” 5th Int. Conf. Inf. Assur. Secur. IAS 2009, vol. 2, pp. 346–349, 2009, doi: 10.1109/IAS.2009.124.

[26] T. Shon, Y. Kim, C. Lee, and J. Moon, “A machine learning framework for network anomaly detection using SVM and GA,” Proc. from 6th Annu.IEEE Syst. Man Cybern. Inf. Assur. Work. SMC 2005, vol. 2005, pp. 176–183, 2005, doi: 10.1109/IAW.2005.1495950.

[27] G. SankaraRao et al., "Security Attacks DoS/DDoS attack Detection in Networks," NeuroQuantology, vol. 20, no. 11, pp. 8452-8463, Sep. 2022, doi: 10.48047/nq.2022.20.11.NQ66839.

[28] J. Galeano-Brajones, J. Carmona-Murillo, J. Valenzuela-Valdes, F. Luna, "Detection and Mitigation Of Dos And Ddos Attacks In Iot-based StatefulSdn: An Experimental Approach", Sensors, vol. 20, no. 3, p. 816, 2020. https://doi.org/10.3390/s20030816

[29] R. Chaganti, B. Bhushan, R. Vinayakumar, "The Role Of Blockchain In Ddos Attacks Mitigation: Techniques, Open Challenges and Future Directions",, 2022. https://doi.org/10.48550/arxiv.2202.03617

[30] F. Alhaidari, A. Alrehan, "A Simulation Work For Generating a Novel Dataset To Detect Distributed Denial Of Service Attacks On Vehicular Ad Hoc Network Systems", International Journal of Distributed Sensor Networks, vol. 17, no. 3, p. 155014772110002, 2021. https://doi.org/10.1177/15501477211000287

[31] R. Abubakar, X. Huang, M. Javed, "An Intelligent Agent-based Detection System ForDdos Attacks Using Automatic Feature Extraction and Selection", Sensors, vol. 23, no. 6, p. 3333, 2023. https://doi.org/10.3390/s23063333

[32] J. Zhang, L. Qidi, R. Jiang, X. Li, "A Feature Analysis Based Identifying Scheme Using Gbdt For Ddos With Multiple Attack Vectors", Applied Sciences, vol. 9, no. 21, p. 4633, 2019. https://doi.org/10.3390/app9214633

[33] M. Khandelwal, D. Gupta, P. Bhale, "Dos Attack Detection Technique Using Back Propagation Neural Network",, 2016. https://doi.org/10.1109/icacci.2016.7732185

[34] B. Bousalem, V. Silva, R. Langar, S. Cherrier, "Deep Learning-based Approach For Ddos Attacks Detection and Mitigation In 5g And Beyond Mobile Networks",, 2022. https://doi.org/10.1109/netsoft54395.2022.9844053

[35] A. Chartuni, J. Márquez, "Multi-classifier Of Ddos Attacks In Computer Networks Built On Neural Networks", Applied Sciences, vol. 11, no. 22, p. 10609, 2021. https://doi.org/10.3390/app112210609

[36] M. M. Rasheed, A. K. Faieq, and A. A. Hashim, "Development of a new system to detect denial of service attack using machine learning classification," Indonesian Journal of Electrical Engineering and Computer Science, vol. 23, no. 2, pp. 1068-1072, Aug. 2021.

DOI: 10.11591/ijeecs.v23.i2.pp1068-1072.

[37] M. Alkasassbeh, et al., "Detecting distributed denial of service attacks using data mining techniques," International Journal of Advanced Computer Science and Applications, vol. 7, no. 1, pp. 128-134, 2016

[38] A. Bonguet and M. Bellaiche, "A Survey Of Denial-of-service and Distributed Denial Of Service Attacks And Defenses In Cloud Computing," Future Internet, vol. 3, no. 9, p. 43, 2017. [Online]. Available: https://doi.org/10.3390/fi9030043

[39] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Rincon, D. Siracusa, "Lucid: a Practical, Lightweight Deep Learning Solution For Ddos Attack Detection", IEEE Trans. Netw. Serv. Manage., vol. 17, no. 2, p.

876-889, 2020. https://doi.org/10.1109/tnsm.2020.2971776

[40] M. Hefeeda, A. Habib, "Detecting Dos Attacks and Service Violations In Qos-enabled Networks", Handbook of Security and Networks, p. 191-220, 2011. https://doi.org/10.1142/9789814273046_0007

[41] G. S. Rao et al., "A Novel Approach for Detection of DoS / DDoS Attack in Network Environment using Ensemble Machine Learning Model," Int. J. Recent Innov. Trends Comput. Commun., vol. 11, no. 9, pp. Page Numbers, 2023. doi: 10.17762/ijritcc.v11i9.8340.

[42] B. Hari Krishna, "A Soft Voting Classifier based Intrusion Detection for Network Security Enhancement," Industrial Engineering Journal, vol. 52, no. 6, pp. 1-6, June 2023.