DDOS Attack Detection using Machine Learning

Akshay Kulkarni 17BCE1115

Eshwar Reddy 17BCE1078

Mohika Thampi 17BCE1079

A Project Report Submitted To

Prof. Gayathri R

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

for the course of

CSE4001 PARALLEL AND DISTRIBUTED COMPUTING

in

B.Tech. COMPUTER SCIENCE AND ENGINEERING

VIT CHENNAI

Vandalur – Kelambakkam Road

Chennai – 600127

NOVEMBER 2019

BONAFIDE CERTIFICATE

Certified that this project report entitled **“DDOS Attack Detection using Machine Learning”** is a bonafide work of **AKSHAY KULKARNI (17BCE1115), ESHWAR REDDY (17BCE1078) & MOHIKA THAMPI (17BCE1079)** who carried out the project work under my supervision and guidance.

Prof. Gayathri R

School of Computing Science and Engineering,

VIT University, Chennai

Chennai – 600 127.

ABSTRACT:

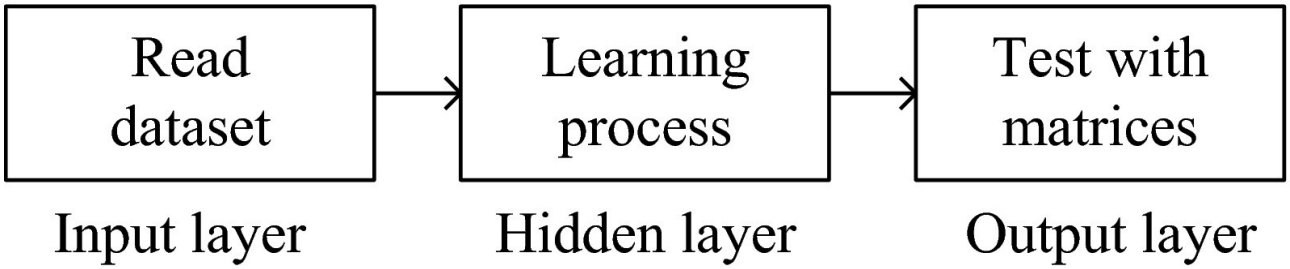
Users and Internet service providers (ISPs) are constantly affected by denial-of-service (DoS) attacks. This cyber threat continues to grow even with the development of new protection technologies. Developing mechanisms to detect this threat is a current challenge in network security. This article presents a machine learning- (ML-) based DoS detection system. The proposed approach makes inferences based on signatures previously extracted from samples of network traffic. The experiments were performed using four modern benchmark datasets. The results show an online detection rate (DR) of attacks above 96%, with high precision (PREC) and low false alarm rate (FAR) using a sampling rate (SR) of 20% of network traffic.

INTRODUCTION:

In recent years, distributed denial-of-service (DDoS) attacks have caused significant financial losses to industry and governments worldwide, as shown in information security reports. These records are in line with the growing number of devices connected to the Internet, especially driven by the popularization of ubiquitous computing, materialized through the Internet of Things (IoT) paradigm and characterized by the concept of connecting anything, anywhere, anytime. In most Internet scenarios, devices interact with applications that run remotely on the network, which enables malicious agents to take control of devices. In this way, it is possible to have the interruption of services or the use of devices as a launching point of attacks for diverse domains, as is the case of the DDoS attack, which has been consolidated for several reasons, such as (i) simplicity and facility of execution, not requiring vast technological knowledge on the attacker side, and (ii) variety of platforms and applications for facilitated attack orchestration. Many of these attacks succeeded in disrupting essential Internet services such as DNS, affecting millions of users around the world, and commercial platforms such as the GitHub, prompting severe financial losses to the organizations that depend on those services.

METHODOLOGY:

This project makes use of Machine Learning Algorithms to detect the presence of DdoS Attacks in a system.



The basic working principle of a ML Algorithm is as illustrated above.

Here:

There are two main programs:

train.py test.py

*train.py is used to train the models(sklearn ones) on different protocols which will also take some parameters as input
The file will exclusively ask for pretrained model to be stored on the system

*test.py is used to extract the pretrained model and test the model on the parameters given

The code is as follows:

train.py

```
import numpy as np
import pandas as pd
import sys
import pickle
from sklearn.neighbors import KNeighborsClassifier

df =
pd.read_csv("./revised_kddcup_dataset.csv",index_col=
0)
```

```
def train_icmp(df, classifier=0):
    """
    Only two best classifiers have been employed on
these datasets
    """
    icmp_df = df[df.loc[:,"protocol_type"] == "icmp"]
    icmp_features =
["duration","src_bytes","wrong_fragment","count","urg
ent","num_compromised","srv_count"]
    icmp_target = "result"
    X = icmp_df.loc[:,icmp_features]
    y = icmp_df.loc[:,icmp_target]
    classes = np.unique(y)
    for i in range(len(classes)):
        if i == 2:
            icmp_df = icmp_df.replace(classes[i], 0)
        else:
            icmp_df = icmp_df.replace(classes[i], 1)

    #turning the service attribute to categorical
values
    #icmp_df=icmp_df.replace("eco_i",-0.1)
    #icmp_df=icmp_df.replace("ecr_i",0.0)
    #icmp_df=icmp_df.replace("tim_i",0.1)
    #icmp_df=icmp_df.replace("urp_i",0.2)

    y = icmp_df.loc[:,icmp_target] #updating the y
variables
    print("Data preprocessing done.")

    #choose KNN if classifier == 0 else choose ID3
    if str(classifier) == "0":
        k = 3
        model = KNeighborsClassifier(n_neighbors=k)
    elif str(classifier) == "1":
        from sklearn.tree import
DecisionTreeClassifier
        model = DecisionTreeClassifier()
    else:
        print("Wrong model chosen! Placing default
model 0 to model training!")
        k = 3
        model = KNeighborsClassifier(n_neighbors=k)
```

```
    #fitting our model
    model.fit(X,y)
    print("The model has been fit.")

    print("Save the fitted model?(y/n):")
    choice = input()
    if choice == "y":
        pickle.dump(model,
open("./saved_model/icmp_data.sav", 'wb'))

def train_tcp_syn(df, classifier=0):
    """
    Only two best classifiers have been employed on
these datasets
    """
    tcp_syn_df = df[df.loc[:,"protocol_type"] ==
"tcp"]
    tcp_syn_df =
tcp_syn_df[tcp_syn_df.loc[:,"srv_serror_rate"] > 0.7]

    service_values =
np.unique(tcp_syn_df.loc[:,"service"])
    mid = (len(service_values)+1)/2
    for i in range(len(service_values)):
        tcp_syn_df =
tcp_syn_df.replace(service_values[i], (i-mid)/10)

    features =
["service","count","srv_count","src_bytes","serror_ra
te"]
    target = "result"

    X = tcp_syn_df.loc[:,features]
    y = tcp_syn_df.loc[:,target]
    classes = np.unique(y)
    for i in range(len(classes)):
        if i == 2:
            tcp_syn_df =
tcp_syn_df.replace(classes[i], 0)
        else:
            tcp_syn_df =
tcp_syn_df.replace(classes[i], 1)
```

```
    #turning the service attribute to categorical
values
    #icmp_df=icmp_df.replace("eco_i",-0.1)
    #icmp_df=icmp_df.replace("ecr_i",0.0)
    #icmp_df=icmp_df.replace("tim_i",0.1)
    #icmp_df=icmp_df.replace("urp_i",0.2)

    y = tcp_syn_df.loc[:,target] #updating the y
variables

    print("Data preprocessing done.")

    #choose KNN if classifier == 0 else choose ID3
    if str(classifier) == "0":
        k = 3
        model = KNeighborsClassifier(n_neighbors=k)
    elif str(classifier) == "1":
        from sklearn.tree import
DecisionTreeClassifier
        model = DecisionTreeClassifier()
    else:
        print("Wrong model chosen! Placing default
model 0 to model training!")
        k = 3
        model = KNeighborsClassifier(n_neighbors=k)


    #fitting our model
    model.fit(X,y)
    print("The model has been fit.")

    print("Save the fitted model?(y/n):")
    choice = input()
    if choice == 'y':
        pickle.dump(model,
open("./saved_model/tcp_syn_data.sav", 'wb'))

def train_udp(df, classifier=0):
    """
    Only two best classifiers have been employed on
these datasets
    """
    udp_df = df[df.loc[:,"protocol_type"] == "udp"]
```

```
    service_values =
np.unique(udp_df.loc[:,"service"])
    mid = (len(service_values)+1)/2
    for i in range(len(service_values)):
        udp_df = udp_df.replace(service_values[i],
(i-mid)/10)

    udp_features =
["dst_bytes","service","src_bytes","dst_host_srv_coun
t","count"]
    udp_target = "result"

    X = udp_df.loc[:,udp_features]
    y = udp_df.loc[:,udp_target]
    classes = np.unique(y)
    for i in range(len(classes)):
        if i == 2:
            udp_df = udp_df.replace(classes[i], 0)
        else:
            udp_df = udp_df.replace(classes[i], 1)

    #turning the service attribute to categorical
values
    #icmp_df=icmp_df.replace("eco_i",-0.1)
    #icmp_df=icmp_df.replace("ecr_i",0.0)
    #icmp_df=icmp_df.replace("tim_i",0.1)
    #icmp_df=icmp_df.replace("urp_i",0.2)

    y = udp_df.loc[:,udp_target] #updating the y
variables
    print("Data preprocessing done.")

    #choose KNN if classifier == 0 else choose ID3
    if str(classifier) == "0":
        k = 3
        model = KNeighborsClassifier(n_neighbors=k)
    elif str(classifier) == "1":
        from sklearn.tree import
DecisionTreeClassifier
        model = DecisionTreeClassifier()
    else:
        print("Wrong model chosen! Placing default
model 0 to model training!")
        k = 3
```

```
        model = KNeighborsClassifier(n_neighbors=k)

    #fitting our model
    model.fit(X,y)
    print("The model has been fit.")

    print("Save the fitted model?(y/n):")
    choice = input().lower()
    if choice == "y":
        pickle.dump(model,
open("./saved_model/udp_data.sav", 'wb'))


if __name__ == "__main__":
    if str(sys.argv[1]) == "icmp":
        train_icmp(df,sys.argv[2])
    elif str(sys.argv[1]) ==  "tcp_syn":
        train_tcp_syn(df,sys.argv[2])
    elif str(sys.argv[1]) == "udp":
        train_udp(df,sys.argv[2])
    else:
        print("Did not select correct protocol. Try
again.")
```

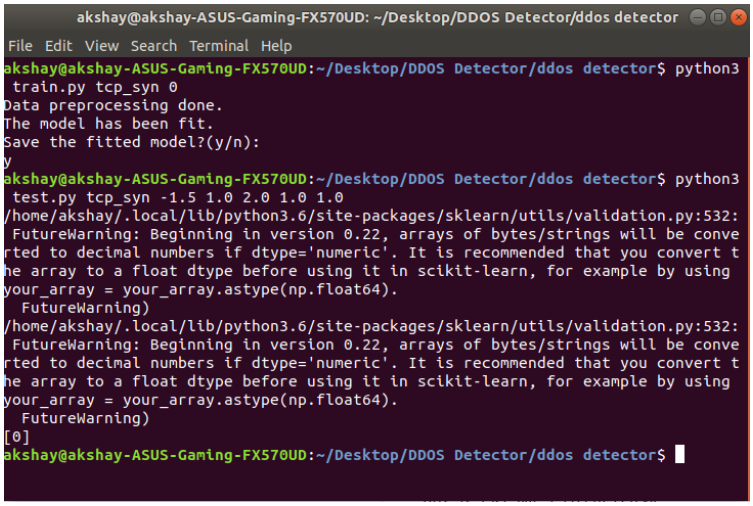
test.py

```
import numpy as np
import sys
import pickle
def icmp_test(attributes):
    model =
pickle.load(open("./saved_model/icmp_data.sav",
'rb'))
    result = model.predict([attributes])
    print(result)
```

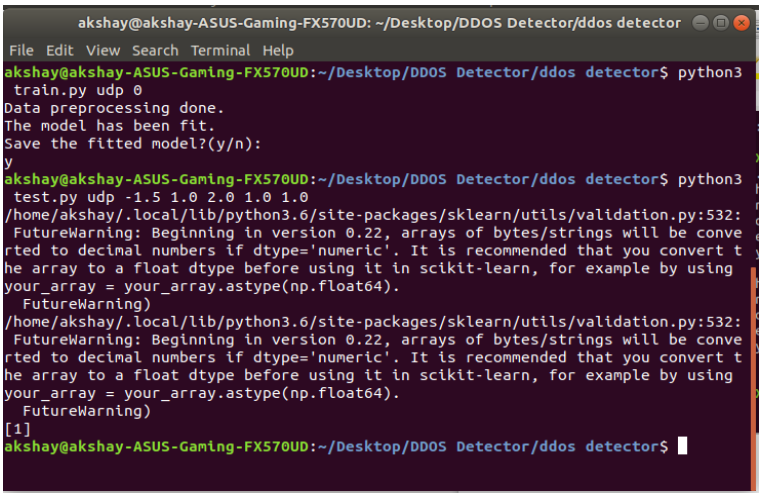
```
def udp_test(attributes):
    model =
pickle.load(open("./saved_model/udp_data.sav", 'rb'))
    result = model.predict([attributes])
    print(result)

def tcp_syn_test(attributes):
    model =
pickle.load(open("./saved_model/tcp_syn_data.sav",
'rb'))
    result = model.predict([attributes])
    print(result)

if __name__ == "__main__":
    if sys.argv[1] == "icmp":
        icmp_test(sys.argv[2:])
    elif sys.argv[1] == "tcp_syn":
        tcp_syn_test(sys.argv[2:])
    elif sys.argv[1] == "udp":
        udp_test(sys.argv[2:])
    else:
        sys.exit("Incorrect protocol has been chosen
for testing. Try again.")
```

OUTPUT:

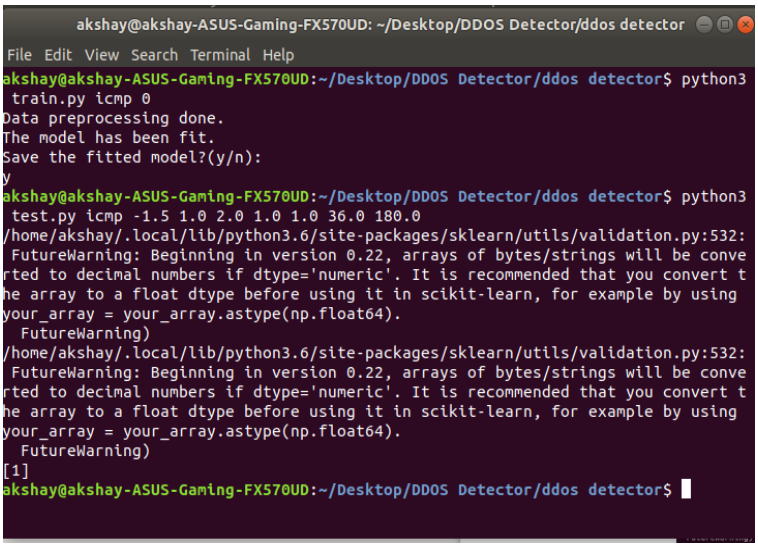
TCP Protocol



UDP Protocol



ICMP Protocol



CONCLUSION:

We have achieved 99% accuracy with the ML Algorithms used in the project. Jupyter Notebooks were used to visualize the accuracy data of the Algorithms used.

```
Accuracy of the model is:  99.93938291810632
Confusion Matrix:
 [[   82    24]
 [    6 49379]]
Report:
              precision    recall  f1-score   support

           0       0.93      0.77      0.85       106
           1       1.00      1.00      1.00     49385
```

```
    accuracy                           1.00     49491
   macro avg       0.97      0.89      0.92     49491
weighted avg       1.00      1.00      1.00     49491


==============***==============
Accuracy of the model is:  99.99393829181064
Confusion Matrix:
 [[  104     2]
 [    1 49384]]
Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       106
           1       1.00      1.00      1.00     49385

    accuracy                           1.00     49491
   macro avg       1.00      0.99      0.99     49491
weighted avg       1.00      1.00      1.00     49491


==============***==============
Accuracy of the model is:  99.90301266897012
Confusion Matrix:
 [[   96    10]
 [   38 49347]]
Report:
              precision    recall  f1-score   support

           0       0.72      0.91      0.80       106
           1       1.00      1.00      1.00     49385

    accuracy                           1.00     49491
   macro avg       0.86      0.95      0.90     49491
weighted avg       1.00      1.00      1.00     49491
```

REFERENCES:

1. S. T. Zargar, J. Joshi, and D. Tipper, “A survey of defense mechanisms against distributed denial of service (DDOS) flooding attacks,” IEEE Communications Surveys and Tutorials, vol. 15, no. 4, pp. 2046–2069, 2013.

2. A. Marzano, D. Alexander, O. Fonseca et al., “The evolution of bashlite and mirai IoT botnets,” in Proceedings of the 2018 IEEE Symposium on Computers and Communications (ISCC), 2018.

3. S. Kottler, “February 28th DDoS incident report,” 2018, https://github.blog/2018-03-01-ddos-incident-report/.
4. Y. Cao, Y. Gao, R. Tan, Q. Han, and Z. Liu, “Understanding internet DDoS mitigation from academic and industrial perspectives,” IEEE Access, vol. 6, pp. 66641–66648, 2018.
5. R. Roman, J. Zhou, and J. Lopez, “On the features and challenges of security and privacy in distributed internet of things,”Computer Networks, vol. 57, no. 10, pp. 2266–2279, 2013.