An Industry Oriented Major Project Report

On

A Hybrid Deep Learning Approach For Botnet Detection In IOT Networks

Submitted in Partial Fulfillment of the Academic Requirement for the Award of Degree of

BACHELOR OF TECHNOLOGY

In

Computer Science and Engineering (AI&ML)

Submitted by:

B. RAJESH	**20R01A6606**
K. KARTHIK	**20R01A6624**
M.YASHASWINI	**20R01A6635**
V.BHAVAN TEJA	**20R01A6657**

Under the Guidance Of

Mr. G.VENU GOPAL RAO

(Assistant Professor, CSE(AI&ML) Dept)

CMR INSTITUTE OF TECHNOLOGY

(UGC AUTONOMUS)

(Approved by AICTE, Affiliated to JNTU, Kukatpally, Hyderabad)

Kandlakoya, Medchal Road, Hyderabad

2023-2024

CMR INSTITUTE OF TECHNOLOGY

(UGC AUTONOMOUS)

(Approved by AICTE, Affiliated to JNTUH, Kukatpally ,Hyderabad)
Kandlakoya , Medchal Road ,Hyderabad

CERTIFICATE

This is to certify that a Mini Project entitled with **"A HYBRID DEEP LEARNING APPROACH FOR BOTNET DETECTION IN IOT NETWORKS"** is submitted by:

B. RAJESH	**20R01A6606**
K. KARTHIK	**20R01A6624**
M.YASHASWINI	**20R01A6635**
V.BHAVAN TEJA	**20R01A6657**

to JNTUH , Hyderabad, in partial fulfillment of the requirement for award of degree of B.Tech in CSE(AIML) and is a record of a bonafide work carried out under our guidance and supervision .The results in the project have been verified and are found to be satisfactory. The results embodied in this work have not been submitted to have any other university for award of any other degree or diploma.

Signature Of Internal Guide	Signature Of Coordinator	Signature Of HOD
Mr G. Venu Gopal Rao	Dr. S. Dhanalakshmi	Prof. P. Pavan Kumar

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We are extremely grateful to **Dr. M. Janga Reddy**, **Director**, **Dr. B. Satyanarayana**, **Principal** and **Mr.P.Pavan kumar**, **Head of Department**, Dept of Computer Science and Engineering (AIML), CMR Institute of Technology for their inspiration and valuable guidance during entire duration.

We are extremely thankful to **Dr. S. Dhanalakshmi** , Mini Project Co-ordinator and internal guide **Mr. G. Venu Gopal Rao**, Dept of Computer Science and Engineering (AIML), CMR Institute of Technology for his constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this project

We express our thanks to all staff members and friends for all the help and coordination extended in bringing out this Project successfully in time.

Finally, we are very much thankful to our parents and relatives who guided directly or indirectly for successful completion of the project.

B. RAJESH	**20R01A6606**
K. KARTHIK	**20R01A6624**
M. YASHASWINI	**20R01A6635**
V. BHAVAN TEJA	**20R01A6657**

TABLE OF CONTENTS

ABSTRACT

Cloud computing is perhaps the most enticing innovation in the present figuring situation. It gives an expense-effective arrangement by diminishing the enormous forthright expense of purchasing equipment foundations and processing power. Fog computing is an additional help to cloud infrastructure by utilizing a portion of the less-registered undertaking at the edge devices, reducing the end client's reaction time, such as IoT. However, most of the IoT devices are resource-constrained, and there are many devices that cyber attacks could target. Cyber-attacks such as bottleneck, Dos, DDoS, and botnets are still significant threats in the IoT environment. Botnets are currently the most significant threat on the internet. A set of infected systems connected online and directed by an adversary to carry out malicious actions without authorization or authentication is known as a botnet. A botnet can compromise the system and steal the data. It can also perform attacks, like Phishing, spamming, and more. To overcome the critical issue, we exhibit a novel botnet attack detection approach that could be utilized in fog computing situations to dispense with the attack using the programmable nature of the software-defined network (SDN) environment.We carefully tested the most recent dataset for our proposed technique, standard and extended performance evaluation measures, and current DL models. To further illustrate overall performance, our findings are cross-validated. The proposed method performs better than previous ones in correctly identifying 99.98% of multi-variant sophisticated bot attacks. Additionally, the time of our suggested method is 0.022(ms), indicating good speed efficiency results.

LIST OF FIGURES

LIST OF SCREENSHOTS

1.INTRODUCTION

1.1 ABOUT PROJECT :

One of the most significant issues for the network system to be efficient and reliable while doing transactions over the IoT is security . The tremendous growth of IoT in different fields, i.e., surveillance, healthcare, transportation, manufacturing industry, education, and others, encourages securing IoT infrastructure to improve its performance. Earlier IoT devices generate data through various types of sensors, and the fog server with malicious intent to lower its performance. Hence, security and protection of the system are among the major issues that can affect the performance of fog computing. In this regard, availability is among the core security requirements for offering services to the actual customer applications according to their interest. However, this is constantly tested by the adversaries by launching different types of attacks, such as DoS or DDoS attacks . An individual or a group can perform these attacks. If a group performs it, it is named ‘‘botnet,’’ while if an individual launches it, it is known as ‘‘bot-master.’’. The bot-master is the attacker node that can launch several types of attacks on the server, such as Phishing, spam, Click fraud, and others. A command-and-control channel remotely controls a botnet. The command-and-control channel is a system the adversary uses to control by sending messages and commands to a compromised system. The adversary can steal the data through these commands and manipulate the infected network . In a botnet attack, some ‘n’ number of compromised nodes are controlled by a bot-master, and they launch an attack on the server from different compromised systems. In the fog computing paradigm security is still challenging task, and various security schemes are proposed to make it resilient against vulnerabilities. However, most of the schemes focus on flexibility and continuous monitoring of the fog server. Software-defined networking (SDN) is used at fog servers to address flexibility, and continuous monitoring issues. SDN is an emerging networking paradigm that assists in making the network more flexible that can help in managing the network, analyzing the traffic, and assisting in the routing control architectures as there is a separate control plan that provides a flexible device management policy. Hence, an SDN-based fog computing environment provides centralized control to the fog computing system. The characteristics of the SDN based fog computing system are discussed below: • SDN can manage the secure connection for thousands of devices connected over the fog for data transmission. • SDN can provide real-time monitoring and awareness with low latency. • SDN can dynamically balance the load with its flexible architecture The proposed technique is evaluated against well-known performance evaluation metrics of the machine and deep learning algorithms known as precision, F1-score, recall, accuracy, and so forth. • For unbiased results, we also applied the technique of 10-fold-cross-validation. The paper's organization is as follows; section II introduces related literature. Section III shows the security issues in Fog computing. Section IV provides information about Deep Learning and its algorithms. Section V details our proposed system and the methodology used for detection and experimentation, such as Dataset, detection phase, evaluation phase, and experiment. While Section VI comprises the experimental results and our assessment results. Finally, section VII provides the conclusion and defines the future map.

1.2 Existing System :

Several researchers are focusing on detecting botnet attacks these days. The main requirement in botnet detection is identifying the infected devices before they can exploit the network by initiating malicious activity. Authors propose numerous methods that claim to secure the network against botnet attacks. These approaches focus on anomaly detection schemes using artificial intelligence, primarily ML and DL algorithms. In various research approaches, authors used ML and hybrid ML techniques for botnet detection such as BayesNet (BN), Support Vector Machine (SVM), J48, Decision Tree (DT), and Naive Bayes (NB). Furthermore, Machine Learning methods are categorized as the supervised, the unsupervised, or the semi-supervised learning.

Parakash *et al.* performed experiments using three well-known machine learning algorithms to detect DDoS packets: K-Nearest Neighbors algorithm (KNN), SVM, and NB. The findings show that the KNN performs better in detecting

DDoS attacks having 97% accuracy, while SVM and NB algorithms achieve 82% and 83% accuracy, respectively. In the authors proposed a detection scheme that uses the SVM algorithm with their own proposed idle timeout adjustment algorithm (IA). They demonstrated the way their proposed methodology outperforms and achieves better results. In another work, uses, NB, SVM and neural network. Results show that the neural network and NB models performed outclass and achieved 100% accuracy, while the SVM model was at 95% accuracy. Ye *et al.* also used the SVM algorithm and achieved an average accuracy of 95.24%. In , authors performed experiments using various algorithms such as Naive Bayesian and decision tree classifier algorithms. They achieved a 99.6% detection accuracy rate.

DL algorithms are the subset of ML. That can deal with large datasets and unstructured data. ML algorithms do not provide better results for extensive data produced by IoT devices and unstructured data. Hence DL algorithms are preferable for IoT compared to traditional ML algorithms such as KNN, SVM, NB, and others. Different DL and hybrid DL approaches are applied for detecting various kinds of malware in IoT devices. In, the authors described a technique for defending the IoT environment against malware and cyber attacks, such as DDoS, brute force, bot, and infiltration. This strategy makes use of DL in SDN.

Disadvantages

- An existing system is not hybrid deep learning detection policy to improve the efficiency and effectiveness of the SDN-based fog computing architecture. Results show that the proposed scheme works better and provides a better detection rate.
- can't customize the policies and applications dues to its programmable nature.

1.3 Proposed System :

- The system suggests an efficient deep learning framework for detecting Botnet attacks in an SDN-based fog computing environment.
- The practical experiment is performed on N_BaIoT Dataset, which comprises both Botnet attack and benign samples.
- The proposed technique is evaluated against well-known performance evaluation metrics of the machine and deep learning algorithms known as precision, F1-score, recall, accuracy, and so forth.
- For unbiased results, we also applied the technique of 10-fold-cross-validation.
- And in this we use a couple of deep learning algorithms like deep neural networks and long short term memory system

Advantages :

a. System can manage the secure connection for thousands of devices connected over the fog for data transmission.
b. System can provide real-time monitoring and awareness with low latency.
c. System can dynamically balance the load with its flexible architecture.

1.4 LITERATURE SURVEY :

Security remained one of the top research areas in networking paradigms whether it is based on cloud computing, fog computing, IoT or SCADA (Supervisory Control and Data Acquisition) systems or others. Several researchers are focusing on detecting botnet attacks these days. The main requirement in botnet detection while the SVM model was at 95% accuracy. Ye et al. also used the SVM algorithm and achieved an average accuracy of 95.24%. In authors performed experiments using various algorithms such as Naive Bayesian and decision tree classifier algorithms. They achieved a 99.6% detection accuracy rate. ML algorithms face challenges like scalability, learning from massive data, and low-value density data. To convert big data into usable intelligence in the face of an ever-growing big data universe, ML must develop and improve. Massive data is developing exponentially, so ML must develop and evolve to turn big data into valuable insight. DL algorithms are the subset of ML. That can deal with large datasets and unstructured data. ML algorithms do not provide better results for extensive data produced by IoT devices and unstructured data. Hence DL algorithms are preferable for IoT compared to traditional ML algorithms such as KNN, SVM, NB, and others. Different DL and hybrid DL approaches are applied for detecting various kinds of malware in IoT devices . In the authors described a technique for defending the IoT environment against malware and cyber attacks, such as DDoS, brute force, bot, and infiltration. This strategy makes use of DL in SDN. They used the CICIDS2018 dataset for the evaluation of the presented scheme. The proposed model achieved 99.87% accuracy, 0.0554% FPR, with a testing time of only 18.9ms. Likewise and intelligence are still needed. The accuracy of botnet malware detection varies for different algorithms applied to different datasets. In Table 2, several ML or DL based detection schemes are presented. It shows that the selected research area is among the emerging research trends in the field of IoT security.

There is ongoing research in this area using different datasets. As per our findings, a thorough study of the DL hybrid combinations is required to explore the possibility of increasing the accuracy and precision in the detection of botnet attacks such that it further achieves lower FPR in consuming less time. Hence, we tested various combinations of DL algorithms in our research work and concluded that the hybrid deep learning algorithm uses DNN and LSTM is effective. It also produces better outcomes compared to other strategies that have been suggested. Additionally, it completely pinpoints sophisticated and devastating multi-attacks in the IoT environment.

2. REQUIREMENT SPECIFICATIONS

2.1 REQUIREMENT ANALYSIS

The project involved analyzing the design of few applications so as to make the application more user friendly .To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible ,the browser version had to be chosen so that it is compatible with the most of the Browsers.

H/W System Configuration:

➢ Processor	- Pentium –IV
➢ RAM	- 4 GB (min)
➢ Hard Disk	- 20 GB
➢ Key Board	- Standard Windows Keyboard
➢ Mouse	- Two or Three Button Mouse
➢ Monitor	- SVGA

SOFTWARE REQUIREMENTS:

❖ **Operating system**	**:** Windows 7 Ultimate.
❖ **Coding Language**	**:** Python.
❖ **Front-End**	**:** Python.
❖ **Back-End**	**:** Django-ORM
❖ **Designing**	**:** Html, css, javascript.
❖ **Data Base**	**:** MySQL (WAMP Server).

2.2 SPECIFICATION PRINCIPLES

2.2.1 SOFTWARE DESCRIPTION

Python:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Python use

Python is usually used for creating sites and programming, task robotization, information investigation, and information representation. Since it's moderately simple to learn, Python has been taken on by numerous non-software engineers like bookkeepers and researchers, for different regular undertakings, such as coordinating funds.

Its standard library is made up of many functions that come with Python when it is installed.[31] On the Internet there are many other libraries libraries have been examinedto provide wonderful ends in varied areas like Machine Learning (ML), Deep Learning, etc.[32] These libraries make it a powerful language; it can do many different things.

Syntax:

Some of Python's syntax comes from C, because that is the language that Python was written in. But Python uses whitespace to delimit code: spaces or tabs are used to organize code into groups. This is different from C. In C, there is a semicolon at the end of each line and curly braces ({}) are used to group code. Using whitespace to delimit code makes Python a very easy-to-read language.

Statements and control flow

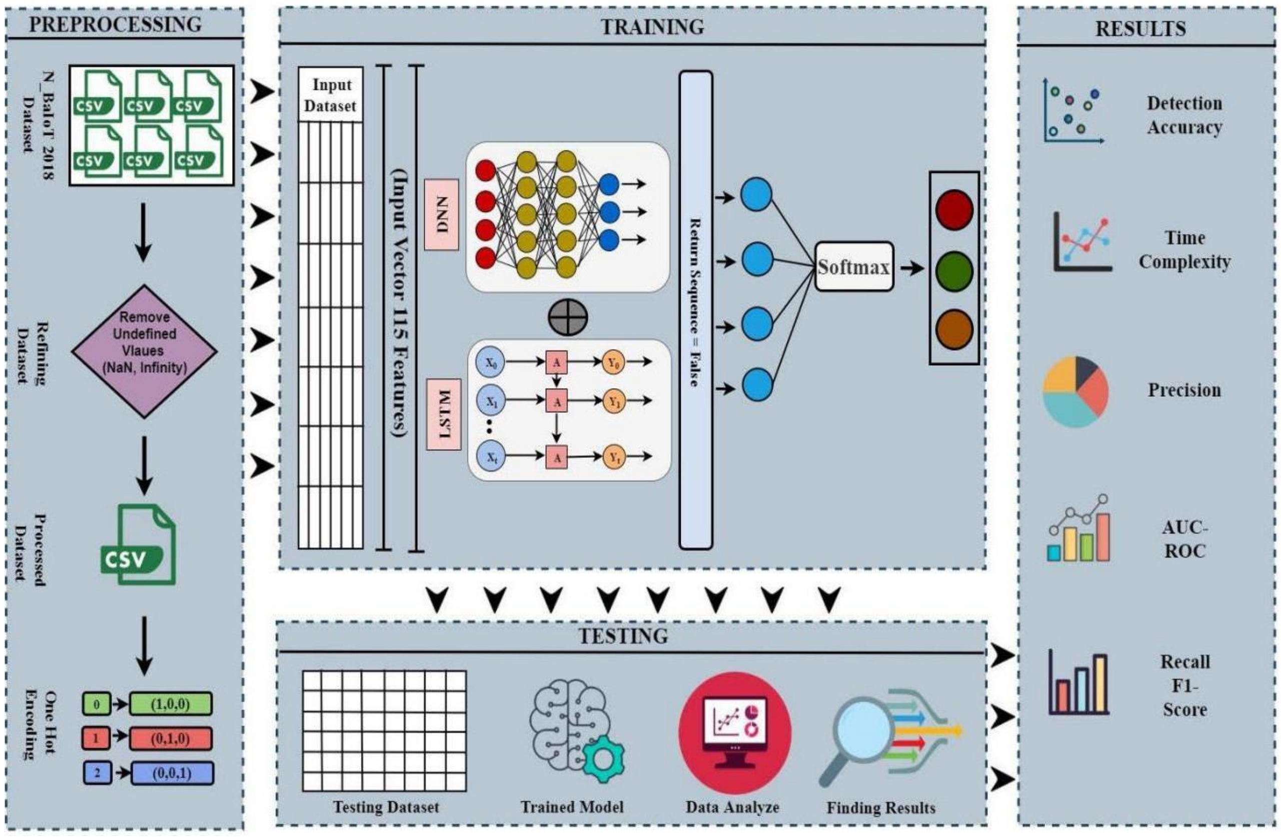
Python's statements include:

- The assignment statement, or the = sign. In Python, the statement `x = 2` means that the name x is bound to the integer 2. Names can be rebound to many different types in Python, which is why Python is a dynamically typed language. For example, you could now type the statement `x = 'spam'` and it would work, but it wouldn't in another language like C or C++.
- The if statement, which runs a block of code if certain conditions are met, along with else and elif (a contraction of else if from other programming languages). The elif statement runs a block of code if the previous conditions are not met, but the conditions for the elif statement are met. The else statement runs a block of code if none of the previous conditions are met.
- The for statement, which iterates over an iterable object such as a list and binds each element of that object to a variable to use in that block of code, which creates a for loop.
- The while statement, which runs a block of code as long as certain conditions are met, which creates a while loop.
- The def statement, which defines a function or method.

- The pass statement, which means "do nothing."
- The class statement, which allows the user to create their own type of objects like what integers and strings are.
- The import statement, which imports Python files for use in the user's code.
- The print statement, which outputs various things to the console.

3.SYSTEM DESIGN

3.1 ARCHITECTURE/BLOCK DIAGRAM :



3.2 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

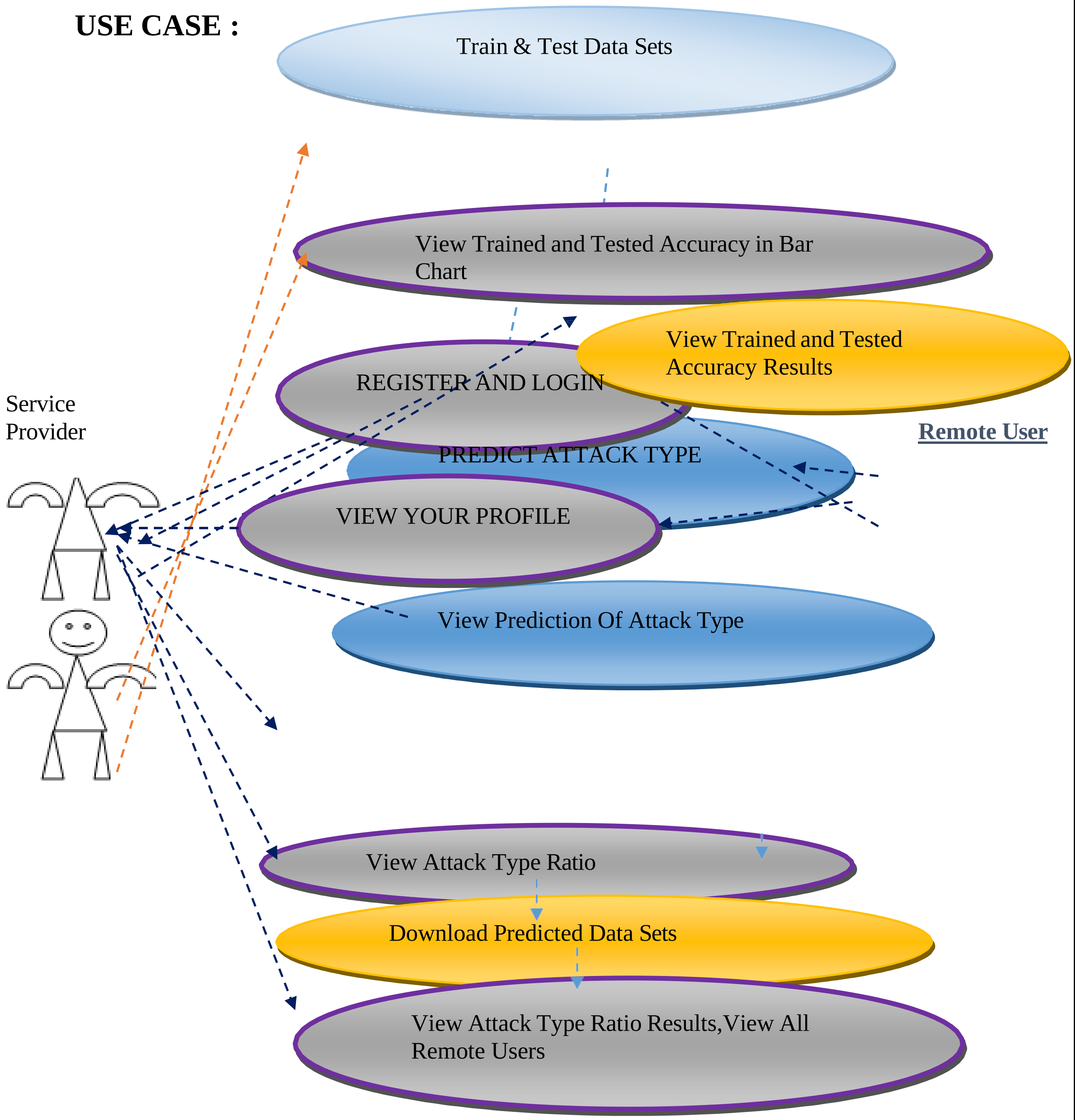
1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.

Provide a formal basis for understanding the modeling language

USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

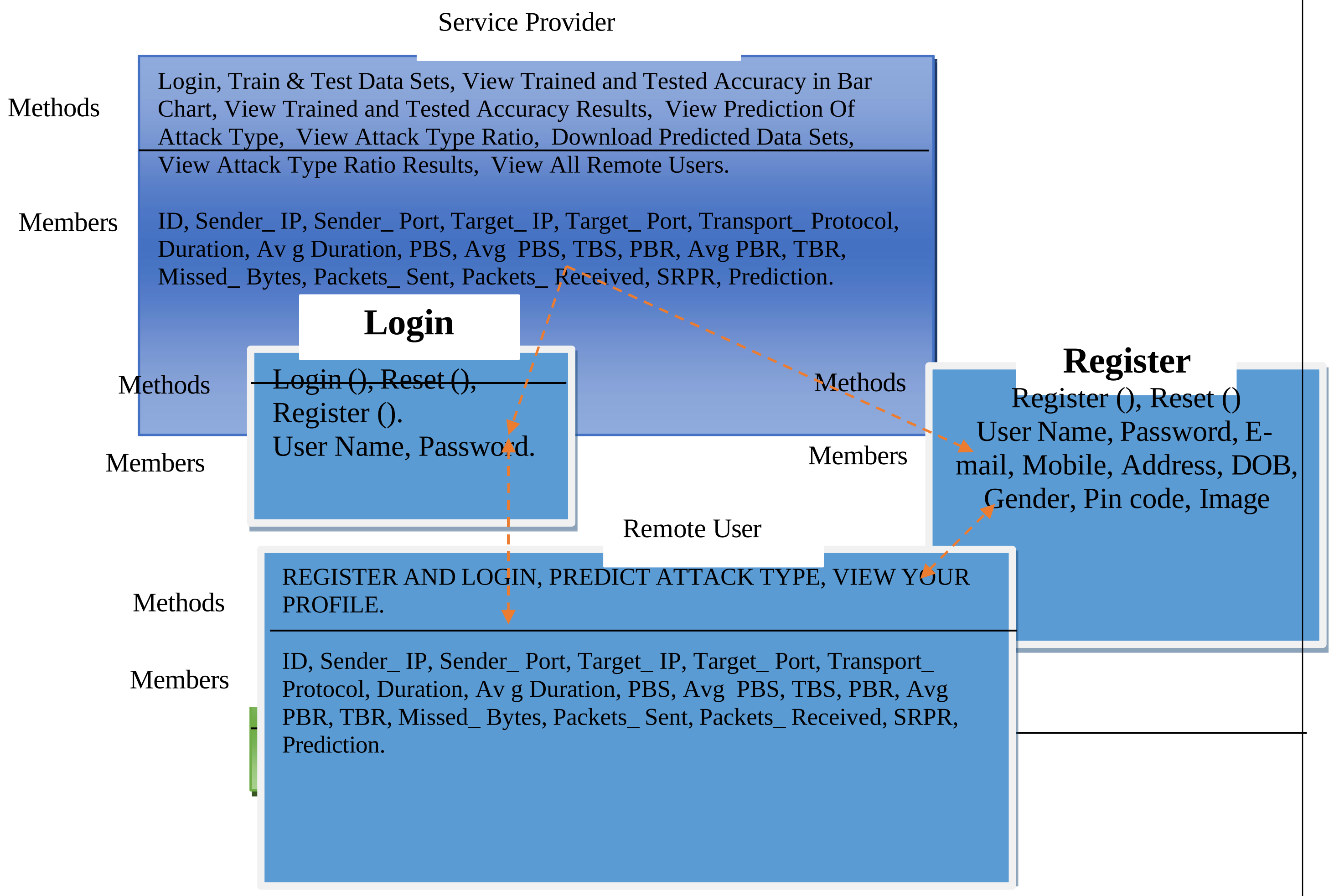
USE CASE :


Train & Test Data Sets
View Trained and Tested Accuracy in Bar Chart
View Trained and Tested Accuracy Results
REGISTER AND LOGIN
Service Provider
Remote User
PREDICT ATTACK TYPE
VIEW YOUR PROFILE
View Prediction Of Attack Type
View Attack Type Ratio
Download Predicted Data Sets
View Attack Type Ratio Results,View All Remote Users

CLASS DIAGRAM :

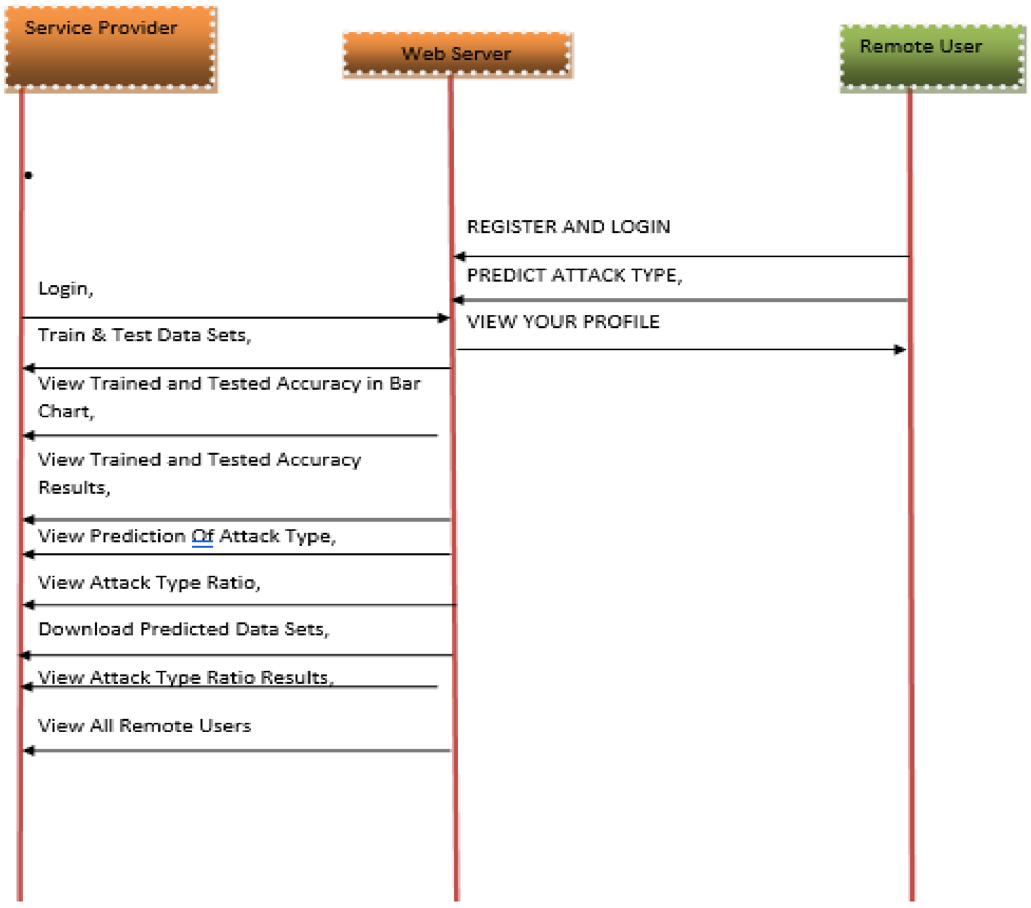
In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

- **Class Diagram :**


Service Provider
Methods
Login, Train & Test Data Sets, View Trained and Tested Accuracy in Bar Chart, View Trained and Tested Accuracy Results, View Prediction Of Attack Type, View Attack Type Ratio, Download Predicted Data Sets, View Attack Type Ratio Results, View All Remote Users.
Members
ID, Sender_ IP, Sender_ Port, Target_ IP, Target_ Port, Transport_ Protocol, Duration, Av g Duration, PBS, Avg PBS, TBS, PBR, Avg PBR, TBR, Missed_ Bytes, Packets_ Sent, Packets_ Received, SRPR, Prediction.
Login
Methods
Login (), Reset (), Register ().
Members
User Name, Password.
Register
Methods
Register (), Reset ()
Members
User Name, Password, E-mail, Mobile, Address, DOB, Gender, Pin code, Image
Remote User
Methods
REGISTER AND LOGIN, PREDICT ATTACK TYPE, VIEW YOUR PROFILE.
Members
ID, Sender_ IP, Sender_ Port, Target_ IP, Target_ Port, Transport_ Protocol, Duration, Av g Duration, PBS, Avg PBS, TBS, PBR, Avg PBR, TBR, Missed_ Bytes, Packets_ Sent, Packets_ Received, SRPR, Prediction.

SEQUENCE DIAGRAM:

- **Sequence Diagram**


Service Provider
Web Server
Remote User
REGISTER AND LOGIN
PREDICT ATTACK TYPE,
VIEW YOUR PROFILE
Login,
Train & Test Data Sets,
View Trained and Tested Accuracy in Bar Chart,
View Trained and Tested Accuracy Results,
View Prediction Of Attack Type,
View Attack Type Ratio,
Download Predicted Data Sets,
View Attack Type Ratio Results,
View All Remote Users

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

IMPLEMENTATION

1) PRE-PROCESSING The N_BaIoT dataset is pre-processed in order to improve the effectiveness and performance of our proposed hybrid deep learning methodology. In order to guarantee the accuracy of the data, we first inspected the dataset and removed any missing nan and infinite values. To hasten the learning process, we used MinMaxScaler to standardise data between 0 and 1. In addition, we trained a deep learning system on target labels using OHE. The steps for pre-processing are as follows: Step 1 (Input): It is the input stage, where the N_BaIoT dataset is loaded that contains 170,000 IoT traffic which consists of six IoT devices to train our algorithm. Here, six CSV files containing data records for each node are loaded. Step 2 (Refining Dataset): In this step, the dataset's NAN and infinite values are eliminated because they are the primary causes of the many errors that can occur when the gradient disappears, slowing down the network and rendering it unsafe. The dataset is refined using MinMaxScaller. Step 3 (Processed Dataset): Here, we get the processed dataset which contains a single CSV file for data collected from all nodes which are free from nan and infinity values. Step 4 (One Hot encoding): OHE is performed to facilitate the DL algorithm to provide better results. In our case, it normalizes the data according to the three categories based on its label (0, 1, 2). 2) TRAINING Step 1 (Input pre-processed dataset): 90% of the processed dataset was given as input to our algorithm for training, which comprises 115 features. Step 2 (Hybrid deep learning phase): In this phase, the hybrid DL technique is applied using two DL algorithms, namely DNN and LSTM. They are executed in parallel on the inputted dataset. Further, the result is merged to get better output. Step 3 (Add Layer): This layer adds a list of inputs. It accepts a list of similar-shaped tensors as input and outputs a single tensor (also of the same shape). Step 4 (Return Sequence: Boolean. Whether the final output in the output sequence should be returned or the entire output sequence should be returned. False is the default value. Step 5 (Softmax function): The softmax function is used as we have multiple classes (Benign, GAFGYT, and MIRAI) that need to be classified properly. As discussed in the results section, it minimizes the prediction errors and improves the detection rate.

4.2 SAMPLE CODE :

MANAGE.PY

```
"""Django's command-line utility for administrative tasks."""
import os
import sys
def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'a_hybrid_deep_learning_approach.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)
```

REMOTE USER:

```
from django.apps import AppConfig

class ClientSiteConfig(AppConfig):
    name = 'Remote_User'
from django import forms
from Remote_User.models import ClientRegister_Model
class ClientRegister_Form(forms.ModelForm):
    password = forms.CharField(widget=forms.PasswordInput())
    email = forms.EmailField(required=True)

    class Meta:
        model = ClientRegister_Model
        fields = ("username","email","password","phoneno","country","state","city")

from django.db import models

# Create your models here.
from django.db.models import CASCADE

class ClientRegister_Model(models.Model):
    username = models.CharField(max_length=30)
    email = models.EmailField(max_length=30)
    password = models.CharField(max_length=10)
    phoneno = models.CharField(max_length=10)
    country = models.CharField(max_length=30)
    state = models.CharField(max_length=30)
    city = models.CharField(max_length=30)
    gender= models.CharField(max_length=30)
    address= models.CharField(max_length=30)

class bottleneck_detection(models.Model):

    ID1= models.CharField(max_length=3000)
    Sender_IP= models.CharField(max_length=3000)
    Sender_Port= models.CharField(max_length=3000)
    Target_IP= models.CharField(max_length=3000)
    Target_Port= models.CharField(max_length=3000)
    Transport_Protocol= models.CharField(max_length=3000)
    Duration= models.CharField(max_length=3000)
    AvgDuration= models.CharField(max_length=3000)
    PBS= models.CharField(max_length=3000)
    AvgPBS= models.CharField(max_length=3000)
    TBS= models.CharField(max_length=3000)
    PBR= models.CharField(max_length=3000)
    AvgPBR= models.CharField(max_length=3000)
    TBR= models.CharField(max_length=3000)
    Missed_Bytes= models.CharField(max_length=3000)
    Packets_Sent= models.CharField(max_length=3000)
    Packets_Received= models.CharField(max_length=3000)
    SRPR= models.CharField(max_length=3000)
    Prediction= models.CharField(max_length=3000)

class detection_accuracy(models.Model):

    names = models.CharField(max_length=300)
    ratio = models.CharField(max_length=300)

class detection_ratio(models.Model):

    names = models.CharField(max_length=300)
    ratio = models.CharField(max_length=300)

from django.db.models import Count
from django.db.models import Q
from django.shortcuts import render, redirect, get_object_or_404

import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import VotingClassifier
# Create your views here.
from Remote_User.models import
ClientRegister_Model,bottleneck_detection,detection_accuracy

def login(request):

     if request.method == "POST" and 'submit1' in request.POST:
        username = request.POST.get('username')
        password = request.POST.get('password')
        try:
            enter = ClientRegister_Model.objects.get(username=username,password=password)
            request.session["userid"] = enter.id

            return redirect('ViewYourProfile')
        except:
            pass
    return render(request,'RUser/login.html')

def index(request):
    return render(request, 'RUser/index.html')

def Add_DataSet_Details(request):

    return render(request, 'RUser/Add_DataSet_Details.html', {"excel_data": ''})

def Register1(request):

    if request.method == "POST":
        username = request.POST.get('username')
        email = request.POST.get('email')
        password = request.POST.get('password')
        phoneno = request.POST.get('phoneno')
        country = request.POST.get('country')
        state = request.POST.get('state')
        city = request.POST.get('city')
        address = request.POST.get('address')
        gender = request.POST.get('gender')
        ClientRegister_Model.objects.create(username=username, email=email,
password=password, phoneno=phoneno,
                                            country=country, state=state,
city=city,address=address,gender=gender)
```

```
            obj = "Registered Successfully"
            return render(request, 'RUser/Register1.html',{'object':obj})
        else:
            return render(request,'RUser/Register1.html')

def ViewYourProfile(request):
    userid = request.session['userid']
    obj = ClientRegister_Model.objects.get(id= userid)
    return render(request,'RUser/ViewYourProfile.html',{'object':obj})

def Predict_Attack_Type_Prediction(request):
    if request.method == "POST":
        if request.method == "POST":
            ID= request.POST.get('ID')
            Sender_IP= request.POST.get('Sender_IP')
            Sender_Port= request.POST.get('Sender_Port')
            Target_IP= request.POST.get('Target_IP')
            Target_Port= request.POST.get('Target_Port')
            Transport_Protocol= request.POST.get('Transport_Protocol')
            Duration= request.POST.get('Duration')
            AvgDuration= request.POST.get('AvgDuration')
            PBS= request.POST.get('PBS')
            AvgPBS= request.POST.get('AvgPBS')
            TBS= request.POST.get('TBS')
            PBR= request.POST.get('PBR')
            AvgPBR= request.POST.get('AvgPBR')
            TBR= request.POST.get('TBR')
            Missed_Bytes= request.POST.get('Missed_Bytes')
            Packets_Sent= request.POST.get('Packets_Sent')
            Packets_Received= request.POST.get('Packets_Received')
            SRPR= request.POST.get('SRPR')

        df = pd.read_csv('Datasets.csv')
```

```
def apply_response(Label):
    if (Label == 0):
        return 0  # No Botnet Attack
    elif (Label == 1):
        return 1  # Botnet Attack

df['Results'] = df['class'].apply(apply_response)

cv = CountVectorizer()
X = df['ID'].apply(str)
y = df['Results']

print("ID")
print(X)
print("Results")
print(y)

X = cv.fit_transform(X)

models = []
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
X_train.shape, X_test.shape, y_train.shape

print("SGD Classifier")
from sklearn.linear_model import SGDClassifier
sgd_clf = SGDClassifier(loss='hinge', penalty='l2', random_state=0)
sgd_clf.fit(X_train, y_train)
sgdpredict = sgd_clf.predict(X_test)
print("ACCURACY")
print(accuracy_score(y_test, sgdpredict) * 100)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, sgdpredict))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, sgdpredict))
models.append(('SGDClassifier', sgd_clf))

print("DEEP NEURAL NETWORK(DNN)")
print("MLP Classifier")
from sklearn.neural_network import MLPClassifier
mlpc = MLPClassifier().fit(X_train, y_train)
y_pred = mlpc.predict(X_test)
print("ACCURACY")
print(accuracy_score(y_test, y_pred) * 100)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, y_pred))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, y_pred))
models.append(('MLPClassifier', mlpc))

classifier = VotingClassifier(models)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

ID1 = [ID]
vector1 = cv.transform(ID1).toarray()
predict_text = classifier.predict(vector1)

pred = str(predict_text).replace("[", "")
pred1 = pred.replace("]", "")

prediction = int(pred1)

if (prediction == 0):
    val = 'No Botnet Attack'
elif (prediction == 1):
    val = 'Botnet Attack'
    print(val)
    print(pred1)

    bottleneck_detection.objects.create(
    ID1=ID,
    Sender_IP=Sender_IP,
    Sender_Port=Sender_Port,
    Target_IP=Target_IP,
    Target_Port=Target_Port,
    Transport_Protocol=Transport_Protocol,
    Duration=Duration,
    AvgDuration=AvgDuration,
    PBS=PBS,
    AvgPBS=AvgPBS,
    TBS=TBS,
    PBR=PBR,
    AvgPBR=AvgPBR,
    TBR=TBR,
    Missed_Bytes=Missed_Bytes,
    Packets_Sent=Packets_Sent,
    Packets_Received=Packets_Received,
    SRPR=SRPR,
    Prediction=val)

    return render(request, 'RUser/Predict_Attack_Type_Prediction.html',{'objs': val})
return render(request, 'RUser/Predict_Attack_Type_Prediction.html')
```

SERVICE PROVIDER:

```
from django.apps import AppConfig


class ResearchSiteConfig(AppConfig):
    name = 'Service_Provider'
from django.db.models import  Count, Avg
from django.shortcuts import render, redirect
from django.db.models import Count
from django.db.models import Q
import datetime
import xlwt
from django.http import HttpResponse
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier

# Create your views here.
from Remote_User.models import
ClientRegister_Model,bottleneck_detection,detection_ratio,detection_accuracy
def serviceproviderlogin(request):
    if request.method  == "POST":
        admin = request.POST.get('username')
        password = request.POST.get('password')
        if admin == "Admin" and password =="Admin":
            detection_accuracy.objects.all().delete()
            return redirect('View_Remote_Users')

    return render(request,'SProvider/serviceproviderlogin.html')
def View_Prediction_Of_Attacker_Type_Ratio(request):
    detection_ratio.objects.all().delete()
    ratio = ""
    kword = 'Botnet Attack'
    print(kword)
    obj = bottleneck_detection.objects.all().filter(Q(Prediction=kword))
    obj1 = bottleneck_detection.objects.all()
    count = obj.count();
    count1 = obj1.count();
    ratio = (count / count1) * 100
    if ratio != 0:
        detection_ratio.objects.create(names=kword, ratio=ratio)

    ratio12 = ""
    kword12 = 'No Botnet Attack'
    print(kword12)
    obj12 = bottleneck_detection.objects.all().filter(Q(Prediction=kword12))
    obj112 = bottleneck_detection.objects.all()
    count12 = obj12.count();
    count112 = obj112.count();
    ratio12 = (count12 / count112) * 100
    if ratio12 != 0:
        detection_ratio.objects.create(names=kword12, ratio=ratio12)

    obj = detection_ratio.objects.all()
    return render(request, 'SProvider/View_Prediction_Of_Attacker_Type_Ratio.html', {'objs': 
obj})

def View_Remote_Users(request):
    obj=ClientRegister_Model.objects.all()
    return render(request,'SProvider/View_Remote_Users.html',{'objects':obj})

def charts(request,chart_type):
    chart1 = detection_ratio.objects.values('names').annotate(dcount=Avg('ratio'))
    return render(request,"SProvider/charts.html", {'form':chart1, 'chart_type':chart_type})

def charts1(request,chart_type):
    chart1 = detection_accuracy.objects.values('names').annotate(dcount=Avg('ratio'))
    return render(request,"SProvider/charts1.html", {'form':chart1, 'chart_type':chart_type})

def View_Prediction_Of_Attacker_Type(request):
    obj =bottleneck_detection.objects.all()
    return render(request, 'SProvider/View_Prediction_Of_Attacker_Type.html',
{'list_objects': obj})

def likeschart(request,like_chart):
    charts =detection_accuracy.objects.values('names').annotate(dcount=Avg('ratio'))
    return render(request,"SProvider/likeschart.html", {'form':charts, 'like_chart':like_chart})

def Download_Predicted_DataSets(request):

    response = HttpResponse(content_type='application/ms-excel')
    # decide file name
    response['Content-Disposition'] = 'attachment; filename="Predicted_Datasets.xls"'
    # creating workbook
    wb = xlwt.Workbook(encoding='utf-8')
    # adding sheet
    ws = wb.add_sheet("sheet1")
    # Sheet header, first row
    row_num = 0
    font_style = xlwt.XFStyle()
    # headers are bold
    font_style.font.bold = True
    # writer = csv.writer(response)
    obj = bottleneck_detection.objects.all()
    data = obj  # dummy method to fetch data.
    for my_row in data:
        row_num = row_num + 1

        ws.write(row_num, 0, my_row.ID1, font_style)
        ws.write(row_num, 1, my_row.Sender_IP, font_style)
        ws.write(row_num, 2, my_row.Sender_Port, font_style)
        ws.write(row_num, 3, my_row.Target_IP, font_style)
        ws.write(row_num, 4, my_row.Target_Port, font_style)
        ws.write(row_num, 5, my_row.Transport_Protocol, font_style)
        ws.write(row_num, 6, my_row.Duration, font_style)
        ws.write(row_num, 7, my_row.AvgDuration, font_style)
        ws.write(row_num, 8, my_row.PBS, font_style)
        ws.write(row_num, 9, my_row.AvgPBS, font_style)
        ws.write(row_num, 10, my_row.TBS, font_style)
        ws.write(row_num, 11, my_row.PBR, font_style)
        ws.write(row_num, 12, my_row.AvgPBR, font_style)
        ws.write(row_num, 13, my_row.TBR, font_style)
        ws.write(row_num, 14, my_row.Missed_Bytes, font_style)
        ws.write(row_num, 15, my_row.Packets_Sent, font_style)
        ws.write(row_num, 16, my_row.Packets_Received, font_style)
        ws.write(row_num, 17, my_row.SRPR, font_style)
        ws.write(row_num, 18, my_row.Prediction, font_style)
    wb.save(response)
    return response
def train_model(request):
    detection_accuracy.objects.all().delete()

    df = pd.read_csv('Datasets.csv')

    def apply_response(Label):
        if (Label == 0):
            return 0  # No Botnet Attack
        elif (Label == 1):
            return 1  # Botnet Attack

    df['Results'] = df['class'].apply(apply_response)

    cv = CountVectorizer()
    X = df['ID'].apply(str)
    y = df['Results']
```

```
print("ID")
print(X)
print("Results")
print(y)

X = cv.fit_transform(X)

models = []
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
X_train.shape, X_test.shape, y_train.shape

print(X_test)

print("SGD Classifier")
from sklearn.linear_model import SGDClassifier
sgd_clf = SGDClassifier(loss='hinge', penalty='l2', random_state=0)
sgd_clf.fit(X_train, y_train)
sgdpredict = sgd_clf.predict(X_test)
print("ACCURACY")
print(accuracy_score(y_test, sgdpredict) * 100)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, sgdpredict))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, sgdpredict))
detection_accuracy.objects.create(names="SGD Classifier", ratio=accuracy_score(y_test,
sgdpredict) * 100)

print("Gradient Boosting Classifier")

from sklearn.ensemble import GradientBoostingClassifier
clf = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=1,
random_state=0).fit(
    X_train,
    y_train)
clfpredict = clf.predict(X_test)
print("ACCURACY")
print(accuracy_score(y_test, clfpredict) * 100)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, clfpredict))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, clfpredict))
models.append(('GradientBoostingClassifier', clf))
detection_accuracy.objects.create(names="Gradient Boosting
Classifier",ratio=accuracy_score(y_test, clfpredict) * 100)
print("DEEP NEURAL NETWORK(DNN)")
print("MLP Classifier")
from sklearn.neural_network import MLPClassifier
mlpc = MLPClassifier().fit(X_train, y_train)
y_pred = mlpc.predict(X_test)
print("ACCURACY")
print(accuracy_score(y_test, y_pred) * 100)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, y_pred))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, y_pred))
models.append(('MLPClassifier', mlpc))
detection_accuracy.objects.create(names="MLPClassifier-DEEP NEURAL
NETWORK(DNN)",ratio=accuracy_score(y_test, y_pred) * 100)

csv_format = 'Results.csv'
df.to_csv(csv_format, index=False)
df.to_markdown
obj = detection_accuracy.objects.all()
return render(request,'SProvider/train_model.html', {'objs': obj})
```

5. TESTING

5.1 SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the
Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are
types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.
Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.

Output : identified classes of application outputs must be exercised.
Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

<u>Integration Testing</u>

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

SYSTEM TESTING

TESTING METHODOLOGIES

The following are the Testing Methodologies:

- **Unit Testing.**
- **Integration Testing.**
- **User Acceptance Testing.**
- **Output Testing.**
- **Validation Testing.**

<u>Unit Testing</u>

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

Integration Testing

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

1. Top Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

2. Bottom-up Integration

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

- The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
- A driver (i.e.) the control program for testing is written to coordinate test case input and output.
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program Structure

User Acceptance Testing

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

Validation Checking

Validation checks are performed on the following fields.

Text Field:

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

Numeric Field:

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

Preparation of Test Data

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are

again uncovered and corrected by using above testing steps and corrections are also noted for future use.

Using Live Test Data:

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

Using Artificial Test Data:

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package “Virtual Private Network” has satisfied all the requirements specified as per software requirement specification and was accepted.

Testing Strategy

A strategy for system testing integrates system test cases and design techniques into a well planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation .A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

System Testing

Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

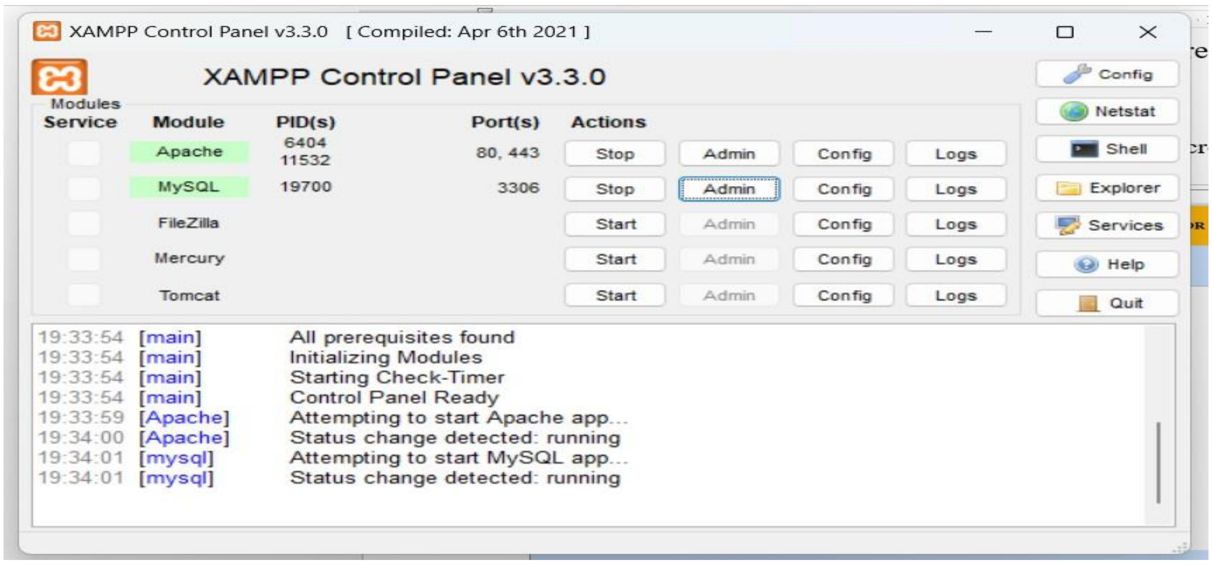
.

In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this. The future holds a lot to offer to the development and refinement of this project.

6. Result

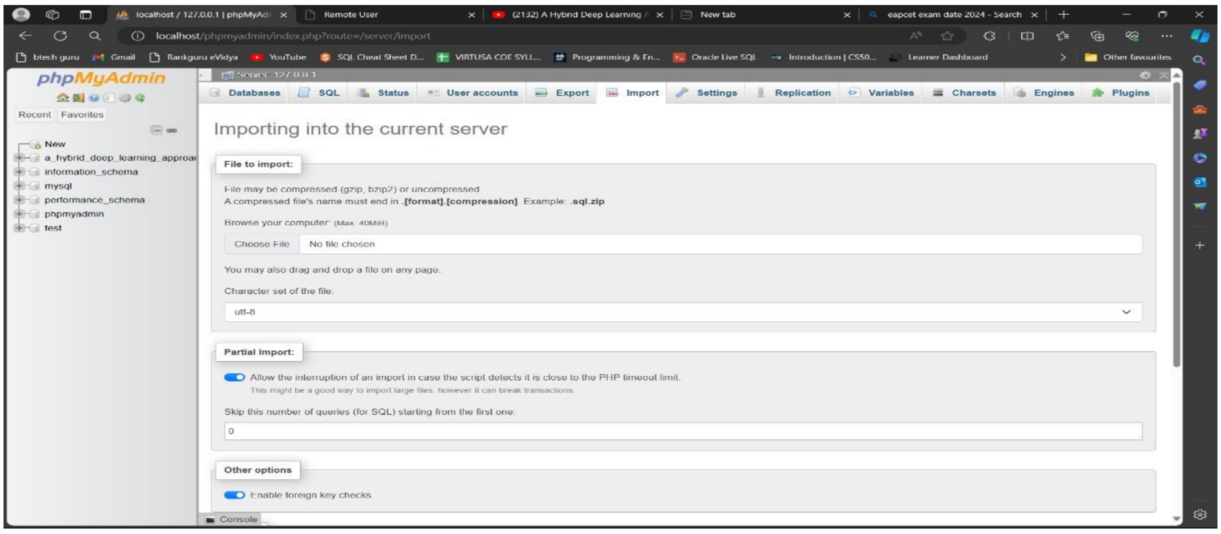
Open xampp control panel and start Apache server and MySQL server

Go to the admin page by clicking on the admin button



Module	PID(s)	Port(s)
Apache	6404 11532	80, 443
MySQL	19700	3306
FileZilla		
Mercury		
Tomcat		

```
19:33:54  [main]      All prerequisites found
19:33:54  [main]      Initializing Modules
19:33:54  [main]      Starting Check-Timer
19:33:54  [main]      Control Panel Ready
19:33:59  [Apache]    Attempting to start Apache app...
19:34:00  [Apache]    Status change detected: running
19:34:01  [mysql]     Attempting to start MySQL app...
19:34:01  [mysql]     Status change detected: running
```



Importing into the current server

File to import:

File may be compressed (gzip, bzip2) or uncompressed.
A compressed file's name must end in **.[format].[compression]**. Example: **.sql.zip**

Browse your computer: (Max: 40MiB)

Choose File   No file chosen

You may also drag and drop a file on any page.

Character set of the file:

utf-8

Partial import:

Allow the interruption of an import in case the script detects it is close to the PHP timeout limit.
This might be a good way to import large files, however it can break transactions.

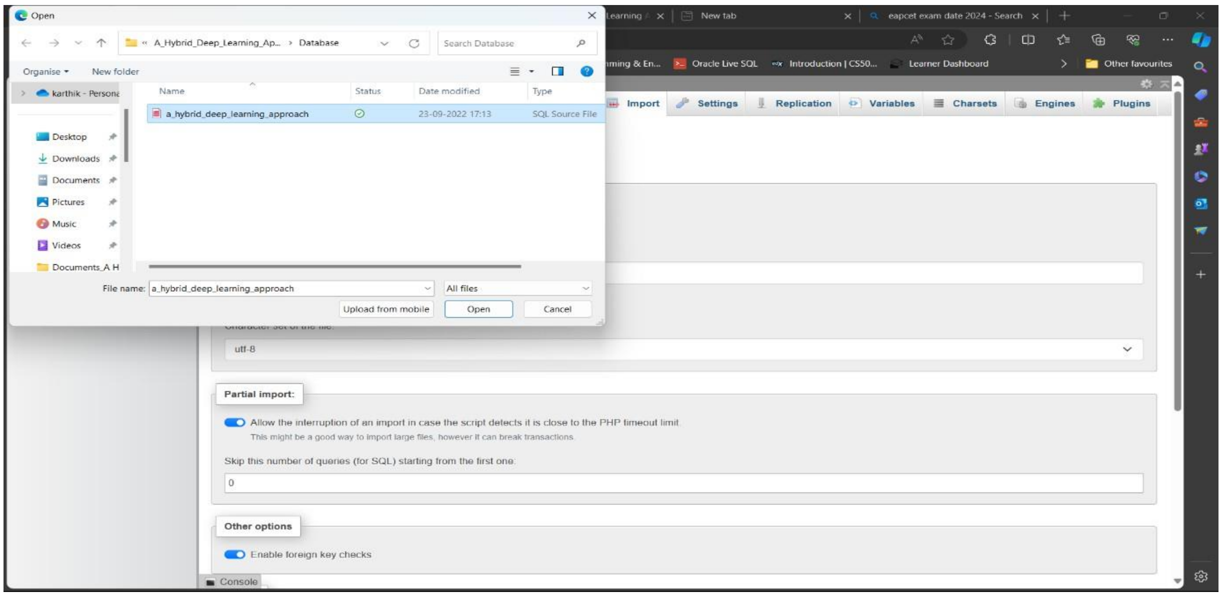
Skip this number of queries (for SQL) starting from the first one:

0

Other options

Enable foreign key checks

Go to import and click on choosefile and import database



Character set of the file: utf-8

Partial import:

Allow the interruption of an import in case the script detects it is close to the PHP timeout limit.
This might be a good way to import large files, however it can break transactions.

Skip this number of queries (for SQL) starting from the first one: 0

Other options

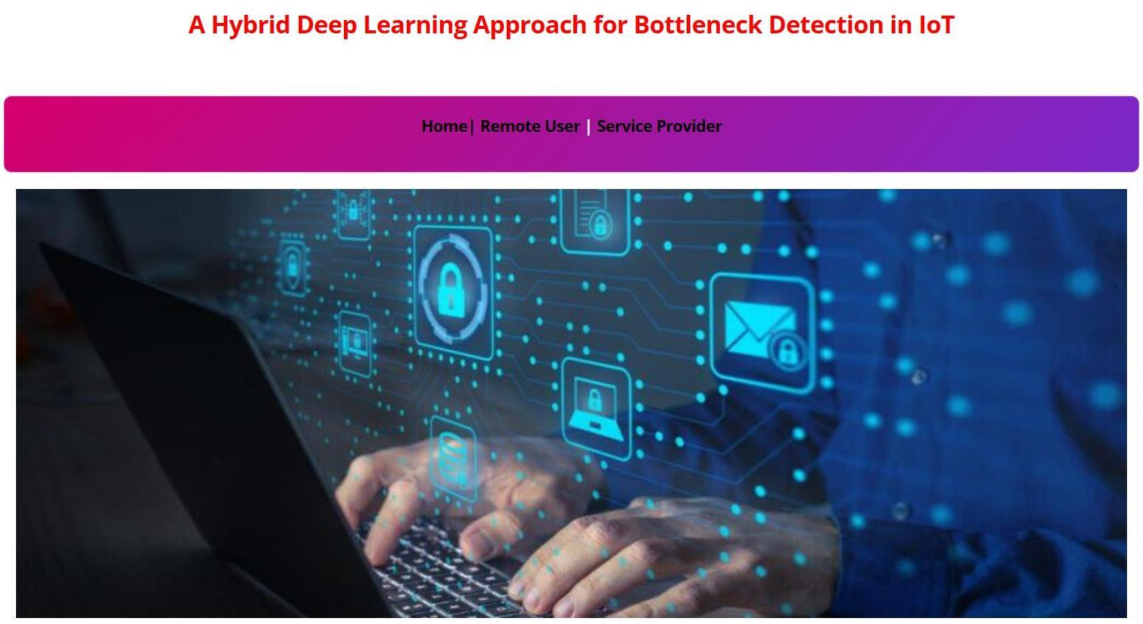
Enable foreign key checks

After importing open the directory in which you are having all the files

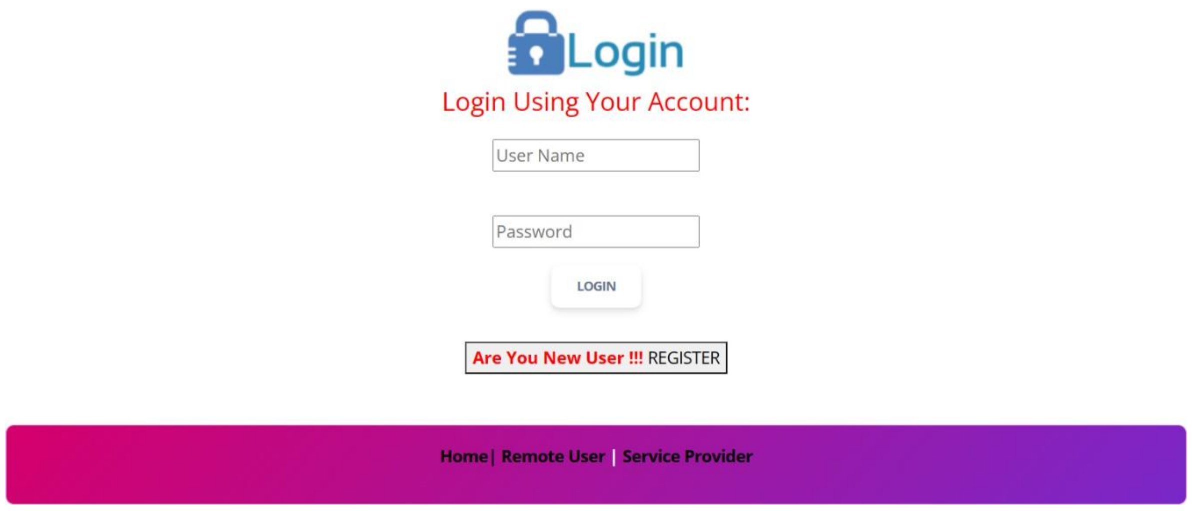
Open command prompt in this directory and by using below command open the web application

:python manage.py runserver

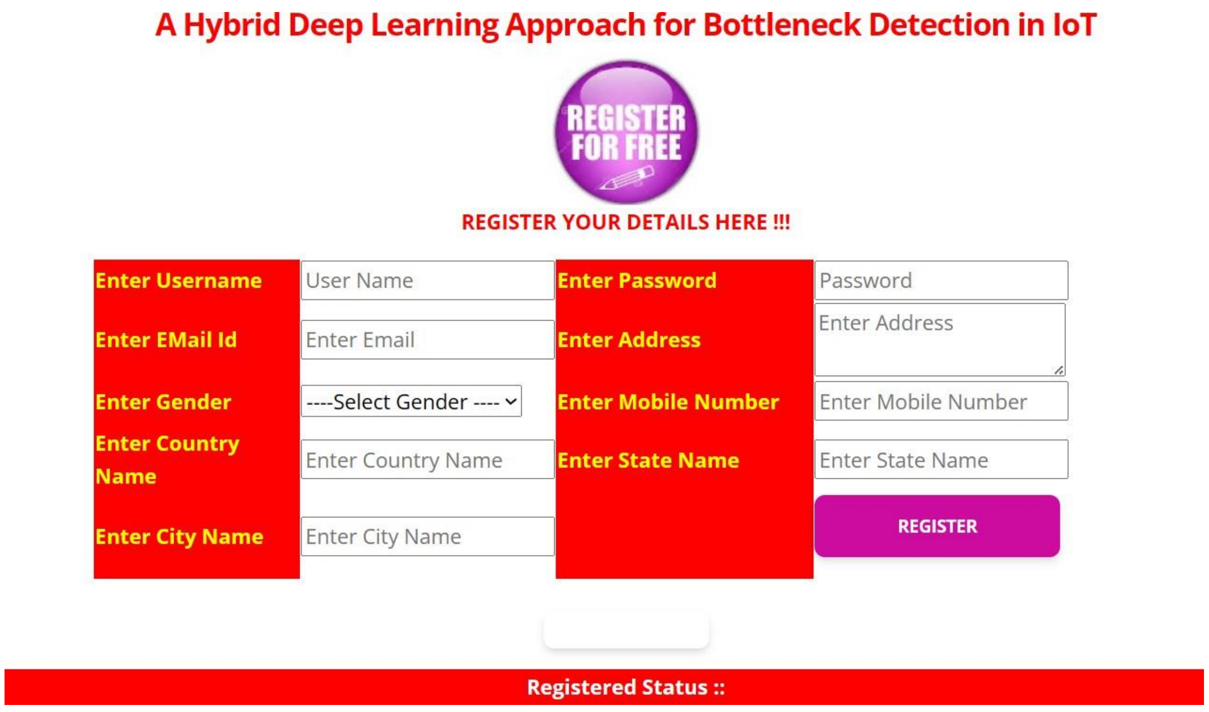
After this you will get a port number click on it


A Hybrid Deep Learning Approach for Bottleneck Detection in IoT
Home | Remote User | Service Provider

Go to remote user


Login
Login Using Your Account:
User Name
Password
LOGIN
Are You New User !!! REGISTER
Home| Remote User | Service Provider

If you have not registered click on the register button


A Hybrid Deep Learning Approach for Bottleneck Detection in IoT

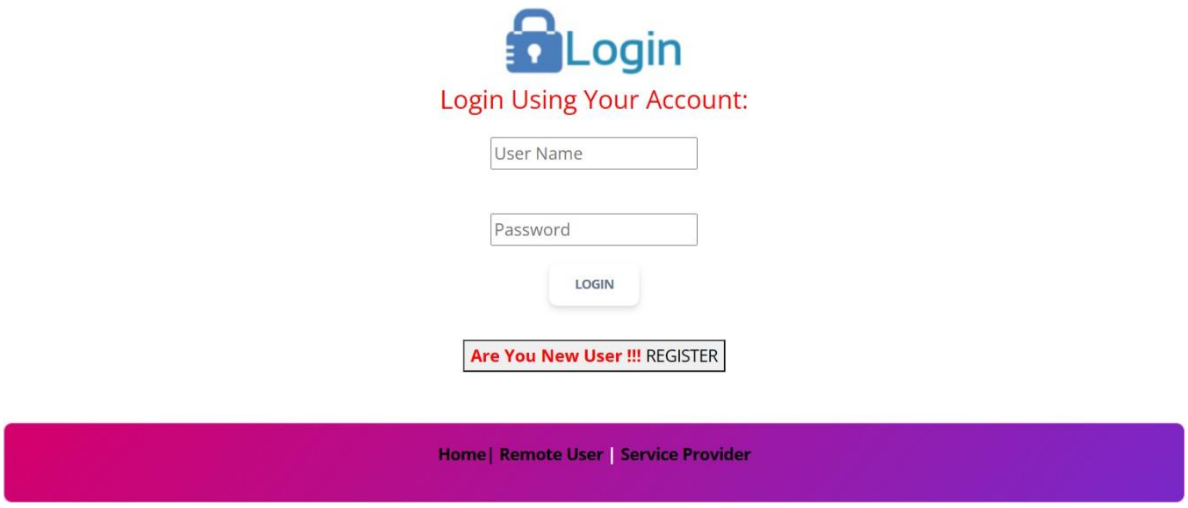
REGISTER YOUR DETAILS HERE !!!

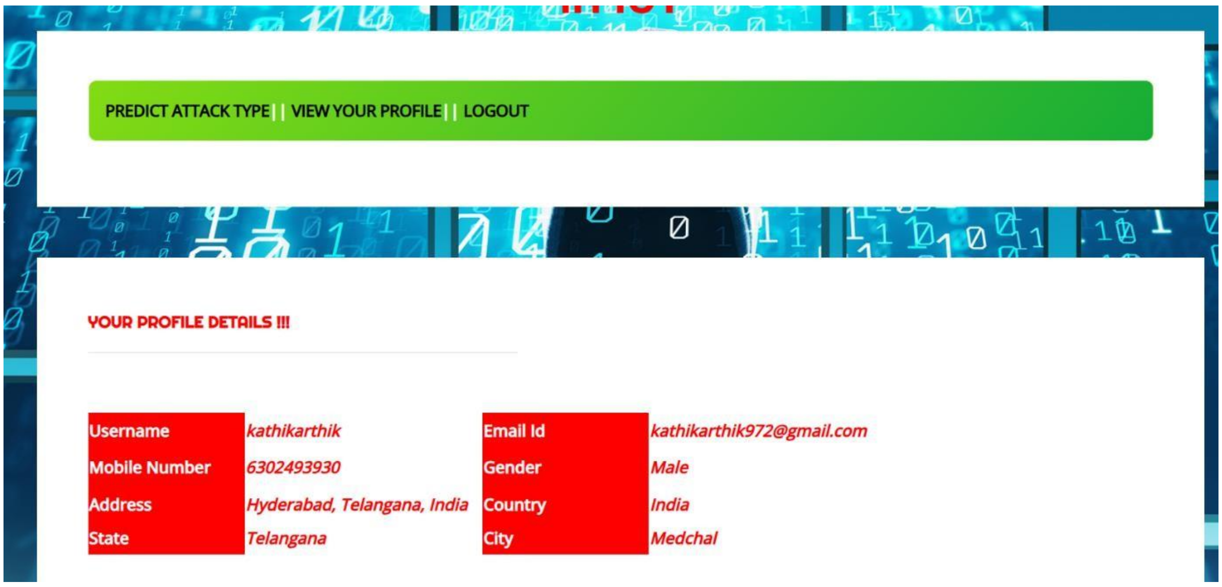
Enter Username	User Name	Enter Password	Password
Enter EMail Id	Enter Email	Enter Address	Enter Address
Enter Gender	----Select Gender ----	Enter Mobile Number	Enter Mobile Number
Enter Country Name	Enter Country Name	Enter State Name	Enter State Name
Enter City Name	Enter City Name		REGISTER

Registered Status ::

Enter all above details and click on register button

And login using your credentials

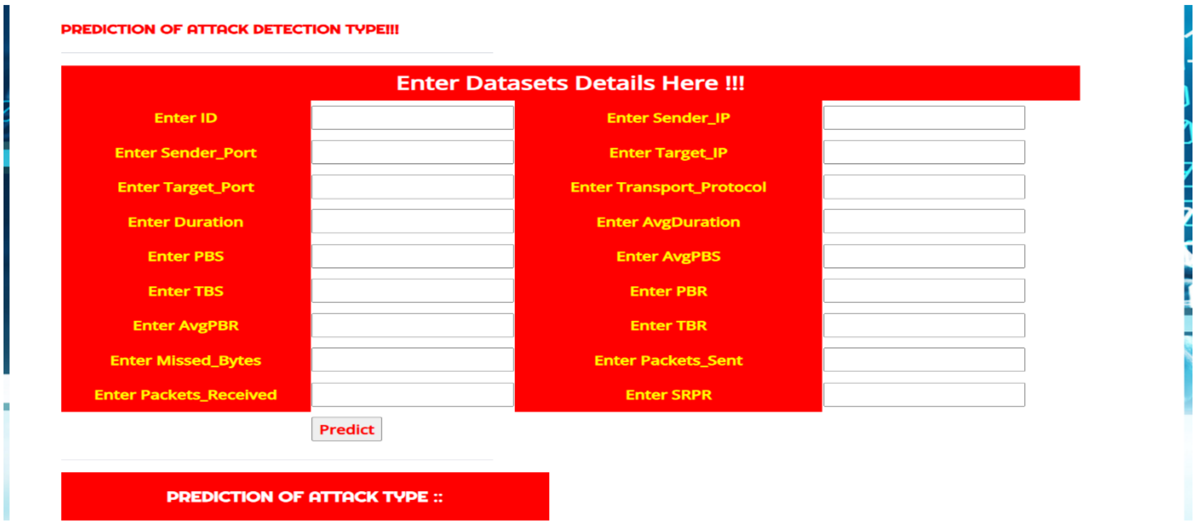

Login
Login Using Your Account:
User Name
Password
LOGIN
Are You New User !!! REGISTER
Home| Remote User | Service Provider



YOUR PROFILE DETAILS !!!

Username	kathikarthik	Email Id	kathikarthik972@gmail.com
Mobile Number	6302493930	Gender	Male
Address	Hyderabad, Telangana, India	Country	India
State	Telangana	City	Medchal

Click on predict attack type


PREDICTION OF ATTACK DETECTION TYPE!!!

Enter Datasets Details Here !!!

Enter ID		Enter Sender_IP	
Enter Sender_Port		Enter Target_IP	
Enter Target_Port		Enter Transport_Protocol	
Enter Duration		Enter AvgDuration	
Enter PBS		Enter AvgPBS	
Enter TBS		Enter PBR	
Enter AvgPBR		Enter TBR	
Enter Missed_Bytes		Enter Packets_Sent	
Enter Packets_Received		Enter SRPR	

PREDICTION OF ATTACK TYPE ::

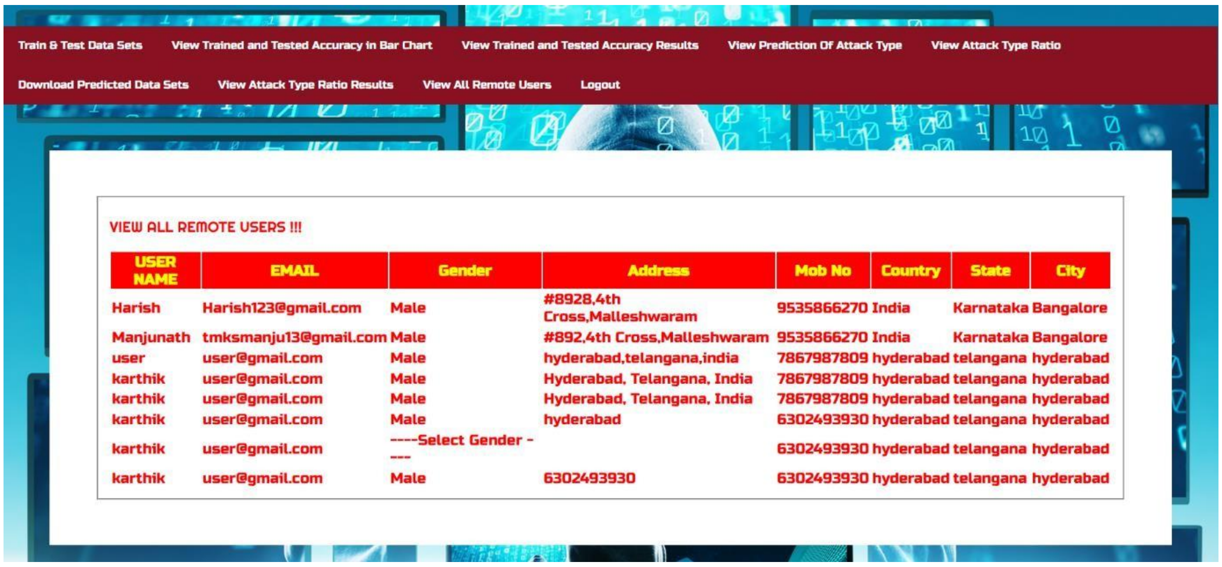
Enter all above details and click on predict

If your networking is under botnet attack it shows that the botnet is detected

And logout from it

Now go into service provider

Give your credentials


Train & Test Data Sets
View Trained and Tested Accuracy in Bar Chart
View Trained and Tested Accuracy Results
View Prediction Of Attack Type
View Attack Type Ratio
Download Predicted Data Sets
View Attack Type Ratio Results
View All Remote Users
Logout

VIEW ALL REMOTE USERS !!!

USER NAME	EMAIL	Gender	Address	Mob No	Country	State	City
Harish	Harish123@gmail.com	Male	#8928,4th Cross,Malleshwaram	9535866270	India	Karnataka	Bangalore
Manjunath	tmksmanju13@gmail.com	Male	#892,4th Cross,Malleshwaram	9535866270	India	Karnataka	Bangalore
user	user@gmail.com	Male	hyderabad,telangana,india	7867987809	hyderabad	telangana	hyderabad
karthik	user@gmail.com	Male	Hyderabad, Telangana, India	7867987809	hyderabad	telangana	hyderabad
karthik	user@gmail.com	Male	Hyderabad, Telangana, India	7867987809	hyderabad	telangana	hyderabad
karthik	user@gmail.com	Male	hyderabad	6302493930	hyderabad	telangana	hyderabad
karthik	user@gmail.com	----Select Gender ----		6302493930	hyderabad	telangana	hyderabad
karthik	user@gmail.com	Male	6302493930	6302493930	hyderabad	telangana	hyderabad

It gives all the infromation about all users who have logedin into our web application

7. CONCLUSION

SDN-based fog computing architectures are the trending networking paradigms for several applications based on the IoT infrastructure. Fog computing systems are vulnerable to various types of Botnet attacks. Hence, there is a need to integrate a security framework that empowers the SDN to monitor the network anomalies against the Botnet attacks. DL algorithms are considered more effective for the IoT-based infrastructures that work on unstructured and large amounts of data. DL based intrusion detection schemes can detect Botnet attacks in the SDN-enabled fog computing IoT system.

We created a framework that utilizes a hybrid DL detection scheme to identify the IoT botnet attacks. It is trained against the dataset that contains normal and malicious data, and then we used this framework to identify botnet attacks that targeted different IoT devices. Our methodology comprises a botnet dataset, a botnet training paradigm, and a botnet detection paradigm.

Our botnet dataset was built using the N_BaIoT dataset, which was produced by driving botnet attacks from the Gafgyt and Mirai botnets into six distinct types of IoT devices. Five attack types, including UDP, TCP, and ACK, are included in both Gafgyt and Mirai attacks. We developed a botnet detection based on three hybrid models_ DNN-LSTM, CNN2D-LSTM, and CNN2D-CNN3D. Using this training model as a foundation, we developed a botnet detection paradigm that can recognise significant botnet attacks. The botnet detection approach is part of a multiclass classification model that can distinguish between the sub-attacks and innocuous data. The fact-finding analysis showed that our hybrid framework DNN-LSTM model had the highest accuracy of 99.98% at identifying the gafgyt and Mirai botnets in the N_BaIoT environment. In 2014 and 2016, the gafgyt and Mirai botnets essentially targeted home routers and IP cameras. The NBaIoT dataset we used for our experiments revealed that rather than the type of IoT devices, the type of training models has a more significant impact on botnet detection performance. We think creating DNN-LSTM-based IoT botnet detection models would be an excellent strategy to enhance botnet identification for different IoT devices.

8. REFERENCES

[1] Z. Hussain, A. Akhunzada, J. Iqbal, I. Bibi, and A. Gani, ``Secure IIoTenabled industry 4.0,'' *Sustainability*, vol. 13, no. 22, p. 12384, Nov. 2021.

[2] R. K. Barik, H. Dubey, K. Mankodiya, S. A. Sasane, and C. Misra, ``Geo-Fog4Health: A fog-based SDI framework for geospatial health big data analysis,'' *J. Ambient Intell. Hum. Comput.*, vol. 10, no. 2, pp. 551_567, Feb. 2019.

[3] S. Khan, S. Parkinson, and Y. Qin, ``Fog computing security: A review of current applications and security solutions,'' *J. Cloud Comput.*, vol. 6, no. 1, pp. 1_22, Dec. 2017.

[4] J. Malik, A. Akhunzada, I. Bibi, M. Talha, M. A. Jan, and M. Usman, ``Security-aware data-driven intelligent transportation systems,'' *IEEE Sensors J.*, vol. 21, no. 14, pp. 15859_15866, Jul. 2021.

[5] Z. Ning, X. Hu, Z. Chen, M. Zhou, B. Hu, J. Cheng, and M. S. Obaidat, ``A cooperative quality-aware service access system for social internet of vehicles,'' *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2506_2517, Aug. 2018.

[6] X. Wang, Z. Ning, M. C. Zhou, X. Hu, L. Wang, Y. Zhang, F. R. Yu, and B. Hu, ``Privacy-preserving content dissemination for vehicular social networks: Challenges and solutions,'' *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1314_1345, 2nd Quart., 2019.

[7] Z. Ning, Y. Li, P. Dong, X. Wang, M. S. Obaidat, X. Hu, L. Guo, Y. Guo, J. Huang, and B. Hu, ``When deep reinforcement learning meets 5Genabled vehicular networks: A distributed of_oading framework for traf_c big data,'' *IEEE Trans. Ind. Informat.*, vol. 16, no. 2, pp. 1352_1361, Feb. 2020.

[8] X.Wang, Z. Ning, and L.Wang, ``Of_oading in internet of vehicles: A fogenabled real-time traf_c management system," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4568_4578, Oct. 2018.

[9] H. Dubey, J. Yang, N. Constant, A. M. Amiri, Q. Yang, and K. Makodiya, ``Fog data: Enhancing telehealth big data through fog computing," in *Proc. ASE BigData Socialinform.*, 2015, pp. 1_6.

[10] W. U. Khan, T. N. Nguyen, F. Jameel, M. A. Jamshed, H. Pervaiz, M. A. Javed, and R. Jäntti, ``Learning-based resource allocation for backscatter-aided vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, early access, Nov. 18, 2021, doi: 10.1109/TITS.2021.3126766.

[11] A. M. Rahmani, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, and P. Liljeberg, ``Exploiting smart e-health gateways at the edge of healthcare Internet-of-Things: A fog computing approach," *Future Gener. Comput. Syst.*, vol. 78, pp. 641_658, Jan. 2018.

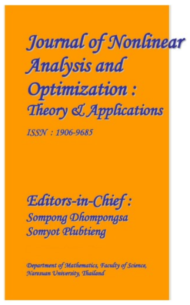
[12] Z. Xiao and Y. Xiao, ``Security and privacy in cloud computing," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 843_859, 2nd Quart., 2013.

[13] Q. Yan, F. R. Yu, Q. Gong, and J. Li, ``Software-de_ned networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 602_622, 1st Quart., 2016.

[14] S. S. C. Silva, R. M. P. Silva, R. C. G. Pinto, and R. M. Salles, ``Botnets: A survey," *Comput. Netw.*, vol. 57, no. 2, pp. 378_403, 2013.

[15] J. Malik, A. Akhunzada, I. Bibi, M. Imran, A. Musaddiq, and S. W. Kim, ``Hybrid deep learning: An ef_cient reconnaissance and surveillance detection mechanism in SDN," *IEEE Access*, vol. 8, pp. 134695_134706, 2020.
022.

Journal of **N**onlinear **A**nalysis and **O**ptimization
Vol. 15, Special Issue. 1, No.01 : 2024
ISSN :**1906-9685**


Journal of Nonlinear Analysis and Optimization : Theory & Applications
ISSN : 1906-9685
Editors-in-Chief :
Sompong Dhompongsa
Somyot Plubtieng
Department of Mathematics, Faculty of Science, Naresuan University, Thailand

HYBRID DEEP LEARNING FOR BOTNET ATTACK DETECTION USING LSTM AND CNN

[1]Maligireddy Yashaswini, [2]Bhukya Rajesh, [3]Kathi Karthik,
[4]Vasikarla Bhavan Teja, [5] G.Venu Gopal Rao,[1,2,3,4]UG Scholar, Department of CSE (AI&ML)
[5]Assistant Professor, Department of CSE (AI&ML)
CMR Institute of Technology, Hyderabad, Telangana, India-501401

ABSTRACT

Cloud computing is perhaps the most enticing innovation in the present figuring situation. It gives an expense-effective arrangement by diminishing the enormous forthright expense of purchasing equipment foundations and processing power. Fog computing is an additional help to cloud infrastructure by utilizing a portion of the less-registered undertaking at the edge devices, reducing the end client's reaction time, such as IoT. However, most of the IoT devices are resource-constrained, and there are many devices that cyber attacks could target. Cyber-attacks such as bottleneck, Dos, DDoS, and botnets are still significant threats in the IoT environment. Botnets are currently the most significant threat on the internet. A set of infected systems connected online and directed by an adversary to carry out malicious actions without authorization or authentication is known as a botnet. A botnet can compromise the system and steal the data. It can also perform attacks, like Phishing, spamming, and more. To overcome the critical issue, we exhibit a novel botnet attack detection approach that could be utilized in fog computing situations to dispense with the attack using the programmable nature of the software-defined network (SDN) environment.We carefully tested the most recent dataset for our proposed technique, standard and extended performance evaluation measures, and current DL models. To further illustrate overall performance, our findings are cross-validated. The proposed method performs better than previous ones in correctly identifying 99.98% of multi-variant sophisticated bot attacks. Additionally, the time of our suggested method is 0.022(ms), indicating good speed efficiency results.

Keywords: Botnet Iot Attack, LSTM, CNN, Auto encoder.

INTRODUCTION: To bring the expected result for Bot-Net attacks in the network devices a new develop dataset is applied. The dataset contains that normal traffic flows and several numerous of cyber-attacks traffic flows in botnets attacks. For the accurate traffic and for the develop effective dataset, the realistic test bed is used for to develop this dataset with the effective information features and also for the improvement of ML model performance and effective prediction model, mostly it were extracted and added it with the extracted features datasets. However, for the best results, the extracted features datasets are labelled as attack flow, categories, and subcategories. Nowadays, the Internet technology is growing up in day to day, and the varies devices are connected with this technology. By introducing this technology, daily life becomes more comfortable and well-organized. On the one side, these technologies are developing numerously but with this rapid development and popularization of internet devices causes the increasing number of cyber-attacks in the desktops is called the Bot-Net Attacks. Still it lacks the security in their internet connection software because most of them have not enough storage and computational resource for robust security mechanisms. In this project, it proposed a machine learning (ML) based botnet attack detection mechanism with sequential detection architecture. Its approach is adopted to implement a lightweight detection system with a high performance. Botnet is a network software bots designed to do malicious activities on the target network which are controlled using command and control protocol by the single unit is called the bot master. Bots are infected computers which is controlled remotely by bot master without any sign of being hacked and are used to perform malicious activities. Botnet size varies from small botnet to the large botnets where small botnet consists of few hundred bots and large botnets consists of 50,000 bots. Hackers can attack the system data without any noticeable indication of their presence. To secure these devices against botnet attacks, ML algorithms have been applied to develop Network Intrusion Detection Systems (NIDS). This NIDS can install at the strategic points within a network. Specifically, Deep Learning, an advanced ML approach, gives the unique capacity for automatic extraction of data from large-scale and high-speed network traffic integrated by interconnecting heterogeneous computer devices. Considering the resource constraints in these devices, NIDS techniques applied in classical networks devices which are not efficient for botnet detection in networks because of high computation and memory requirements. In order to produce an efficient Deep Learning method for botnet attack detection in networks, large network traffic information is required to accept efficient classification performance. The processing and analysing the high-dimensional network traffic data can make the course of dimensionality. Also, training Deep Learning models with high-dimensional data can cause Hughes phenomena. High-dimensional of data process is complex and it requires huge computational resources and memory capacity. Some of the devices do not have sufficient space to store big network traffic data required for Deep Learning. Therefore, there is a need for end-to-end DL-based botnet detection method to reduce the high dimensionality of big network traffic and also detect the complexity and recent botnet attacks accurately based on low-dimensional network traffic information. Currently, Bot-Net dataset is the most relevant publicly available dataset for botnet attack detection in networks. The original feature dimensionality of the Bot-Net dataset is 43, and the memory space required for this network traffic data is 1.085 GB. So far, feature dimensionality reductions methods are applied to the Bot-Net dataset were all based on feature selection techniques.

RELEATED WORKS: Although several types of datasets are available in network intrusion detection. They have several challenges, like lack of reliable labels, redundancy of network traffic, low attack diversity, and missing ground truth. For instance, NSL-KDD and KDD Cup99 datasets are most popularly applied, but they are out dated, and they are not reflect the current normal and the attack scenarios. The usage of the DEFCON-8 dataset is limited because of low number of benign traffic models. These attack scenarios in UNIBS dataset are limited to DOS, Bot-Net dataset is the most related dataset which is publicly available for network devices botnet attack detection in the systems. To realise this dataset, an network test bed was set up to generate benign and malicious network traffic using heterogeneous communication protocols like User Datagram Protocol (UDP), Reverse Address Resolution Protocol (RARP), Transmission Control Protocol (TCP), Internet Control Message Protocol (ICMP), Address Resolution Protocol (ARP), Internet Protocol version6 ICMP (IPv6-ICMP) and Internet Group Management Protocol (IGMP). The test bed setup comprised a variety of IOT devices, including a weather station, smart fridge, remotely-activated garage door and smart thermostat. Also, millions of botnet attack traffic samples were included in Bot-Net. These attack traffic samples can be categorized into four IOT botnet scenarios, namely: DDOS, DOS and information theft. To ensure a fair comparison, feature dimensionality reduction methods that do not include benign network traffic traces and all the four botnet attack scenarios in the Bot-IOT dataset were not included in this paper. For instance, did not consider the DOS attack scenario. Also, the performance of the method in detecting benign network traffic was not reported. In a similar work, the authors did not evaluate the procedure of the proposed method. In another work did not evaluate the procedure of the proposed method with the network traffic data in the BOTNET dataset. In summary, the state-of-the-art methods in the related work focused on the selection of specific features from available network traffic information available in the Bot-IOT dataset. However, this approach may likely affect the efficiency of botnet attack detection in IOT networks because the classifiers will not have access to some relevant network information during training, validation, and testing. Consequently, the feature selection approach may lead to low botnet attack detection accuracy and a high false alarm rate in IOT networks. On the other hand, LAE decreases the dimensionality of big IOT network traffic data and produces a low-dimensional latent space feature representation at the hidden layer without losing useful intrinsic network information.

EXISTING SYSTEM

Several researchers are focusing on detecting botnet attacks these days [28]_[30]. The main requirement in botnet detection is identifying the infected devices before they can exploit the network by initiating malicious activity. Authors propose numerous methods that claim to secure the network against botnet attacks. These approaches focus on anomaly detection schemes using artificial intelligence, primarily ML and DL algorithms. In various research approaches, authors [21]_[23] used ML and hybrid ML techniques for botnet detection such as BayesNet (BN), Support Vector Machine (SVM), J48, Decision Tree (DT), and Naive Bayes (NB). Furthermore, Machine Learning methods are categorized as the supervised, the unsupervised, or the semi-supervised learning.

Parakash *et al.* performed experiments using three well-known machine learning algorithms to detect DDoS packets: K-Nearest Neighbors algorithm (KNN), SVM, and NB. The findings show that the KNN performs better in detecting DDoS attacks having 97% accuracy, while SVM and NB algorithms achieve 82% and 83% accuracy, respectively [33]. In [34], the authors proposed a detection scheme that uses the SVM algorithm with their own proposed idle timeout adjustment algorithm (IA). They demonstrated the way their proposed methodology outperforms and achieves better results. In another work, [35] uses, NB, SVM and neural network. Results show that the neural network and NB models performed outclass and achieved 100% accuracy, while the SVM model was at 95% accuracy. Ye *et al.* [36] also used the SVM algorithm and achieved an average accuracy of 95.24%. In [37], authors performed experiments using various algorithms such as Naive Bayesian and decision tree classifier algorithms. They achieved a 99.6% detection accuracy rate. DL algorithms are the subset of ML. That can deal with large datasets and unstructured data. ML algorithms do not provide better results for extensive data produced by IoT devices and unstructured data [38]. Hence DL algorithms are preferable for IoT compared to traditional ML algorithms such as KNN, SVM, NB, and others. Different DL and hybrid DL approaches are applied for detecting various kinds of malware in IoT devices [39]_[41]. In [42], the authors described a technique for defending the IoT environment against malware and cyber attacks, such as DDoS, brute force, bot, and infiltration. This strategy makes use of DL in SDN.

DISADVANTAGES

- An existing system is not hybrid deep learning detection policy to improve the efficiency and effectiveness of the SDN-based fog computing architecture. Results show that the proposed scheme works better and provides a better detection rate.
- can't customize the policies and applications dues to its programmable nature.

PROPOSED SYSTEM

- The system suggests an efficient deep learning framework for detecting Botnet attacks in an SDN-based fog computing environment.
- The practical experiment is performed on N_BaIoT Dataset, which comprises both Botnet attack and benign samples.
- The proposed technique is evaluated against well-known performance evaluation metrics of the machine and deep learning algorithms known as precision, F1-score, recall, accuracy, and so forth.
- For unbiased results, we also applied the technique of 10-fold-cross-validation.

ADVANTAGES

System can manage the secure connection for thousands of devices connected over the fog for data transmission.

System can provide real-time monitoring and awareness with low latency.

System can dynamically balance the load with its flexible architecture.

CONCLUSION: This system was proposed for efficient botnet detection in networks using deep learning algorithms such as LSTM and CNN. The effectiveness of this method was validated by performing extensive experiments with the most relevant publicly available dataset in binary and multi-class classification scenarios. By using this CNN the result will get more accurate and precision also comparing to the LSTM.

REFERENCES:

1. A. O. Akmandor, Y. Hongxu, and N. K. Jha, "Smart, secure, yet energyefficient, internet-of-things sensors," IEEE Transactions on Multi-Scale Computing Systems, vol. 4, no. 4, pp. 914–930, 2018.

2. D. E. Denning, "An intrusion-detection model," IEEE Transactions on software engineering, no. 2, pp. 222–232, 1987.

3. J. Qiu, L. Du, D. Zhang, S. Su, and Z. Tian, "Nei-tte: Intelligent traffic time estimation based on fine-grained time derivation of road segments for smart city," IEEE Transactions on Industrial Informatics, 2019

4. J. Qiu, Z. Tian, C. Du, Q. Zuo, S. Su, and B. Fang, "A survey on access control in the age of internet of things," IEEE Internet of Things Journal, 2020

5. X.-G. Luo, H.-B. Zhang, Z.-L. Zhang, Y. Yu, and K. Li, "A new framework of intelligent public transportation system based on the internet of things," IEEE Access, vol. 7, pp. 55 290–55 304, 2019.

6. Z. Tian, X. Gao, S. Su, and J. Qiu, "Vcash: A novel reputation framework for identifying denial of traffic service in internet of connected vehicles," IEEE Internet of Things Journal, vol. 7, no. 5, pp. 3901– 3909, May 2020.

1. M.Ganga Eswari, D.Vijayasekar, and, S.Dhanalakshmi,, "Criminal Identification Using Biometric Traits in Image Processing", International Journal of Applied Engineering Research Technology(IJAER), ISSN 0973-4562 Vol. 10 No.85 (2015), October 2015,PP 265-268 (SCOPUS/Annexure-II Journals)

2. D.Vijayasekar, S.Dhivya, and S.Dhanalakshmi, "Wiener Filter Operation on Blurred Images", International Journal of Applied Engineering Research Technology(IJAER), ISSN 0973-4562 Vol. 10 No.85 (2015), October 2015,PP 197-200 (SCOPUS/Annexure-II Journals)

3. Shiva Prasanth.A, and S.Dhanalakshmi,, "Automated Testing Tools for Different Coverage Metrics", International Journal of Advanced Innovative Research (IJAIR), Volume 5, Issue 10, ISSN: 2278-7844, October 2016, PP 120-123

4. K.Sindhu, and S.Dhanalakshmi,, "Disclosure of Malevolent in MANET: A Survey", International Journal of Research in Technological Studies(IJRTS),Volume 4,Issue 1,ISSN:2348-1439,December 2016,PP 31-34 (SCOPUS/Annexure-II Journals)

18. S. M. Babu, P. P. Kumar, B. S. Devi, K. P. Reddy, M. Satish and A. Prakash, "Enhancing Efficiency and Productivity: IoT in Industrial Manufacturing," *2023 IEEE 5th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA)*, Hamburg, Germany, 2023, pp. 693-697, doi: 10.1109/ICCCMLA58983.2023.10346807.
19. Prakash, S. M. Babu, P. P. Kumar, S. Devi, K. P. Reddy and M. Satish, "Predicting Consumer Behaviour with Artificial Intelligence," 2023 IEEE 5th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA), Hamburg, Germany, 2023, pp. 698-703, doi: 10.1109/ICCCMLA58983.2023.10346660.

1.Kumbala Pradeep Reddy; Sarangam Kodati; Thotakura Veeranna; G. Ravi, "6 Machine Learning-Based Intelligent Video Analytics Design Using Depth Intra Coding," in Big Data Management in Sensing: Applications in AI and IoT , River Publishers, 2021, pp.77-86.