International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

HONEYPOT BASED INTRUSION DETECTIONSYSTEM

Bushra Taher[1], Nida Arshad[2], YVSS Pragathi[3]
Department of Computer Science and Engineering, Stanley College of Engineering and Technology for Women, Telangana, India

ABSTRACT

Honeypot is a computer security mechanism set to detect, deflect or counteract attempts at unauthorized use of information. It has enormous potential for the security community. This can be modeled after any digital asset, including software applications, servers or the network itself. It is intentionally and purposefully designed to look like a legitimate target, resembling the model in terms of structure, components and content and is meant to convince the adversary that they have accessed the actual system and encourage them to spend time within this controlled environment. It acts better than firewalls or intrusion detection system by detecting encrypted attacks in IPv6 networks to capture different types of frauds.

With the tremendous growth of IoT botnet DDoS attacks in recent years. IoT security has now become one of the most concerned topics in the field of network security. A lot of security approaches have been proposed in the area, but they still lack in terms of dealing with newer emerging variants of IoT malware, known as Zero-Day Attacks. A honeypot-based approach is presented which uses machine learning techniques for malware detection. The IoT honeypot generated data is used as a dataset for the effective and dynamic training of a machine learning model. The approach can be taken as a productive outset towards combatting Zero-Day DDoS Attacks which now has emerged as an open challenge in defending IoT against DDoS Attacks.

Keywords: Honeypot, Machine Learning, botnet, DDos Attacks, IoT, intrusio

1. INTRODUCTION

1.1About Project:

There are several honeypot based approaches are present in the literature for defending DDoS. The concept of the signature matching method had been used as a detection framework in some of these approaches. Malware is detected on the basis of signatures obtained from their corresponding generated log files from the honeypot. This type of detection was able to deal with only stored signatures and its variations, hence throw a limitation on dealing with an unknown and wider range of malware families. Another solution is anomaly based detection which does not

make use of rules, but a threshold is set for normal user behavior and any deviation from it leads to a declaration of possible malicious behavior. Such systems do suffer from high false positive rates because attackers now can imitate normal behavior too. Moreover, a machine learning based solution is capable to deal with such problem due to its ability to learn and teach over time. Thus, a more accurate classification with less number of false positive can be achieved by training the model with effective and updated data. The machine learning concept is used to better utilize the dynamic data produced by honeypot and increase the predictability for future attacks.

1.2 Objectives of Project:

The main objectives of this project is to get increased predictability of attacks to safeguard security systems, and gather information about the attacker and the attack methods. By deceiving an attacker into carrying out his/her attack on a non-critical, well-monitored system, valuable insight can be gained into their attack methods, and information can be gathered for forensic or legal purposes.

1.3 Scope of the Project:

The scope of the project is effectively deduce machine learning techniques using different algorithms to identify patterns of attacks and thereby train the ML models. Any significant deviation between the observed 'normal' behavior can be regarded as an anomaly, which can be then interpreted as an intrusion. The main assumption of the aforementioned approaches is that malicious behavior differs from typical user behavior. We use this method to increase the efficiency of the IOT based honeypot secure network system

1.4 Advantages:

- Honeypots collect data from actual attacks and other unauthorized activities, providinganalysts with a rich source of useful information.
- Acts as a rich source of information and helps collect real-time data.
- Wastes hackers' time and resources.
- Improves security, ordinary cybersecurity detection technologies generate alerts that can include a significant volume of false positives, but honeypots reduce this volume because there is no reason for legitimate users to access them.

- Honeypots can be good investments because they do not require high-performance resources to process large volumes of network traffic looking for attacks, because theyonly interact with malicious activities.

- Honeypots capture malicious activity, even if an attacker is using encryption.

1.5 Disadvantages

- Honeypots only collect information when an attack occurs. Zero attempts to accessthe honeypot means there is no data to analyze.

- Being distinguishable from production systems, it can be easily identified byexperienced attackers.

- .A honeypot once attacked can be used to attack other systems

- Malicious traffic that has been captured is only collected when an attack targets the honeypot network; if attackers suspect a network is a honeypot, they will avoid it.

- Honeypots are often distinguishable from legitimate production systems, which means experienced hackers can often differentiate a production system from a honeypot system using system fingerprinting techniques.

1.6 Applications

- Honeytokens: Honeytokens are fake records that are inserted in the database. These fake records are not expected to be used by normal users. If any of these honeytokens are used, they alert us of the database having been compromised. An example of honeytokens are fake username/passwords in the user database. These users do not exist in the real world, and hence are not expected to be logging in to the application. If the application sees these credentials being used, it immediately recognizes that the user database has been compromised.

- Honeypages: These are obscure web pages sprinkled in the web site. They have no legitimate purpose, nay they are not even linked from any valid page. Normal users would never reach these pages. However, we drop hints about these pages by embedding their URL as comments or hidden fields in valid pages. While normal users would never see this, an attacker who

analyzes the source code, or a vulnerability scanner that spiders the site would see these and follow the link. When the page is accessed, it points us to the intruder.

- Dummy domains: A variant of honeypages use dummy domains that are published in the DNS. These domains do not have legitimate sites hosted on them, nor do they have URLs pointing to them. Any queries for these dummy domains indicate reconnaissance activity of intruders as they hunt for applications we host. This can give us an early warning of activity targeted at our sites.

1.7 Hardware and Software Requirements:

Software Requirements:

- Python idel 3.7 version (or)
- Anaconda 3.7 (or)
- Jupiter (or)
- Google Colab

Hardware Requirements:

- Operating system : Windows, Linux
- Processor : Intel core processor i5 or greater
- Ram : Min 4GB
- Hard disk : Min 250 GB

2. *LITERATURE SURVEY*

There are several honeypot based approaches are present in the literature for defending DDoS. The concept of the signature matching method had been used as a detection framework in some of these approaches. Malware is detected on the basis of signatures obtained from their corresponding generated log files from the honeypot. This type of detection was able to deal with only stored signatures and its variations, hence throw a limitation on dealing with an unknown and wider range of malware families. Another solution is anomaly based detection which does not make use of rules, but a threshold is set for normal user behavior and any deviation from it leads to a declaration of possible malicious behavior. Such systems do suffer from high false positive rates because attackers now can imitate normal behavior too. Moreover, a machine learning based solution is capable to deal with such problem due to its ability to learn and teach over time. Thus, a more accurate classification with less number of false positive can be achieved by

training the model with effective and updated data. The machine learning concept is used to better utilize the dynamic data produced by honeypot and increase the predictability for future attacks. Many machine learning methods have also been proposed to identify DDoS based on the selection of statistical features using several supervised learning algorithms like SVM, NaïveBayes. However, these methods require extensive network expertise for selecting appropriate features out of the dataset and usually are limited to only one or several DDoS vectors. In addition, they require regular updates of the system to keep it functioning in diverse situations. Another machine learning based solution was proposed to detect DDoS using deep learning models like: Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory Neural Network (LSTM), and Gated Recurrent Unit Neural Network (GRU). A network-based anomaly detection method was proposed which extracts behavior snapshots of the network and uses deep auto encoders to detect anomalous network traffic emanating from compromised IoT devices. However, deep learning models need a large amount of data to train itself for producing accurate outcomes. In spite of that, they have extremely computationally expensive and complex training procedure and often require a significant amount of time to learn. IoT devices cannot afford such extensive procedures as they are quite constrained in terms of resources as well as in providing real-time services to the user. Moreover, there is a need to develop new methods for detecting attacks launched from compromised IoT devices and differentiate between hour and millisecond long IoT-based attacks

2.1 Existing System:

In existing technique honeypot using signature based attack detection which is not efficient so we are deploying machine learning framework at honeypot server to predict whether request is normal or contains attack signature.

2.2 Proposed System:

In propose work author using honeypot server and IOT devices to capture data and this data will be used to train ML algorithms but we don't have any IOT devices so we are using IOT dataset to trained ML algorithms. ML algorithms will be trained with previous data and then this trained model can be used to detect attacks from old or new request signature and this detection will solved ZERO-DAY Distributed Denial of service (DDOS) attacks. In this we are using SVM, Random Forest, K- Nearest Neighbors, Decision Tree and Neural Networks. In all algorithms SVM, KNN and Neural network is giving best performance.

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

3. *PROPOSED ARCHITECTURE*

Support Vector Machine

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification and regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n- dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.

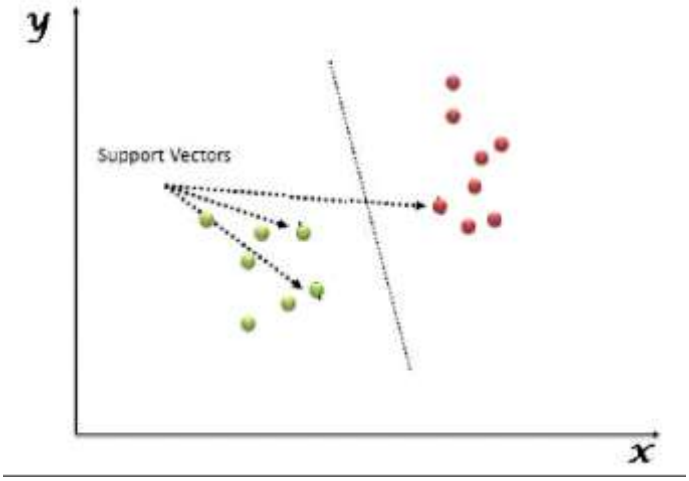


Fig. 3.1

Support Vectors are simply the co-ordinates of individual observation. The SVM classifier isa frontier which best segregates the two classes (hyper-plane/ line).

Logistic regression

Logistic regression is named for the function used at the core of the method, the logistic function.

The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

1 / (1 + e^-value)

Where e is the base of the natural logarithms (Euler's number or the EXP() function) and value is the actual numerical value that you want to transform. Below is a plot

of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.

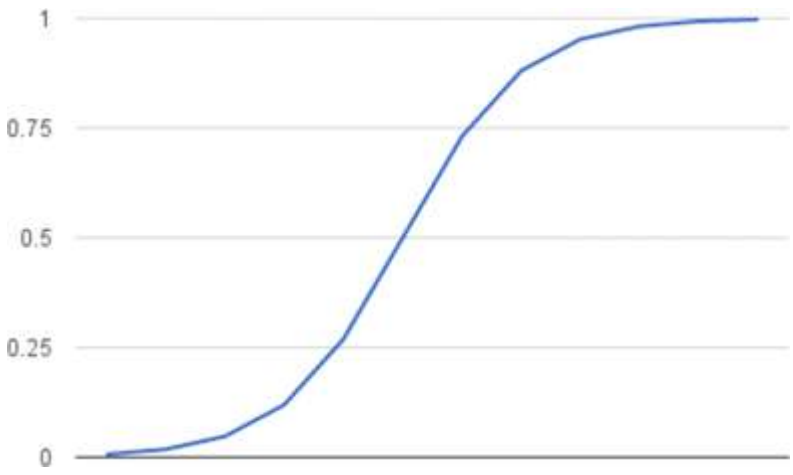


Fig. 3.2

Naive Bayes Classifiers

Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems. The technique is easiest to understand when described using binary or categorical input values.

It is called naive Bayes or idiot Bayes because the calculation of the probabilities for each hypothesis are simplified to make their calculation tractable. Rather than attempting to calculate the values of each attribute value P (d1, d2, d3|h), they are assumed to be conditionally independent given the target value and calculated as P (d1|h) * P (d2|H) and so on.

This is a very strong assumption that is most unlikely in real data, i.e. that the attributes do not interact. Nevertheless, the approach performs surprisingly well on data where this assumptiondoes not hold.

Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

Decision tree Algorithm

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute. This process is then repeated for the subtree rooted at the new node.

AdaBoost Classifier

Ada-boost or Adaptive Boosting is one of ensemble boosting classifier proposed by Yoav Freund and Robert Schapire in 1996. It combines multiple classifiers to increase the accuracy of classifiers. AdaBoost is an iterative ensemble method. AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier. The basic concept behind AdaBoost is to set the weights of classifiers and training the data sample in each iteration such that it ensures the accurate predictions of unusual observations. Any machine learning algorithm can be used as base classifier if it accepts weights on the training set. AdaBoost should meet two conditions:

- The classifier should be trained interactively on various weighed training examples.
- In each iteration, it tries to provide an excellent fit for these examples by minimizing training error.

Steps for Machine Learning Algorithms

- Install Anaconda Latest Version
- Open anaconda Prompt
- Conda create -n tf python=3.7
- Conda activate tf
- Install require softwares
  - scikit-image==0.17.2
  - scikit-learn==0.23.2

- pandas==1.1.1
- matplotlib==3.3.1
- Pillow==7.2.0
- plotly==4.10.0
- opencv-python==4.4.0.42
- spacy==2.3.2
- lightgbm==3.0.0
- maho
- tas==1.4.11
- matplotlib==3.3.1lightgbm==3.0.0
- mahotas==1.4.11
- nltk==3.5
- matplotlib==3.3.1
- xgboost==1.2.0
- Jupyter

- Activate environment for jupyter notebook(For execute the in jupter notebook)
  ➔ python -m ipykernel install --user --name=
- Goto project Directory

UML DIAGRAMS

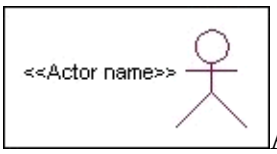
The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

Global Use Case Diagrams:

Identification of actors:

Actor: Actor represents the role a user plays with respect to the system. An actor interacts with, but has no control over the use cases.

Graphical representation:



Actor

An actor is someone or something that:

Interacts with or uses the system.

- Provides input to and receives information from the system.
- Is external to the system and has no control over the use cases. Actors are discovered byexamining:

- Who directly uses the system?
- Who is responsible for maintaining the system?
- External hardware used by the system.
- Other systems that need to interact with the system. Questions to identify actors:
    - Who is using the system? Or, who is affected by the system? Or, which groups need help from the system to perform a task?
    - Who affects the system? Or, which user groups are needed by the system to perform its functions? These functions can be both main functions and secondary functions such as administration.
    - Which external hardware or systems (if any) use the system toperform tasks?
    - What problems does this application solve (that is, for whom)?
    - And, finally, how do users use the system (use case)? What are theydoing with the system?

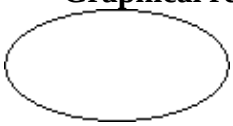
The actors identified in this system are:

- **System Administrator**
- **Customer**
- **Customer Care**

Identification of usecases:

Usecase: A use case can be described as a specific way of using the system from a user's(actor's) perspective.

Graphical representation:



A more detailed description might characterize a use case as:

- Pattern of behavior the system exhibits
- A sequence of related transactions performed by an actor and the system
- Delivering something of value to the actor Use cases provide a means to:
- capture system requirements
- communicate with the end users and domain experts
- test the system

Use cases are best discovered by examining the actors and defining what the actor will be ableto do with the system.

Guide lines for identifying use cases:

- For each actor, find the tasks and functions that the actor should be able to perform or that the system needs the actor to perform. The use case should represent a courseof events that leads to clear goal

- Name the use cases.
- Describe the use cases briefly by applying terms with which the user is familiar. Thismakes the description less ambiguous

Questions to identify use cases:

- What are the tasks of each actor?
- Will any actor create, store, change, remove or read information in the system?
- What use case will store, change, remove or read this information?
- Will any actor need to inform the system about sudden external changes?
- Does any actor need to inform about certain occurrences in the system?
- What use cases will support and maintains the system?

Flow of Events

A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information, written in terms of what the system should do, not how the system accomplishes the task. Flow of events are created as separate files or documents in your favorite text editor and then attached or linked to a use case using the Filestab of a model element.

A flow of events should include:

- When and how the use case starts and ends
- Use case/actor interactions
- Data needed by the use case
- Normal sequence of events for the use case
- Alternate or exceptional flows Construction of Usecase diagrams:

Use-case diagrams graphically depict system behavior (use cases). These diagrams present a high level view of how the system is used as viewed from an outsider's (actor's) perspective. A use-case diagram may depict all or some of the use cases of a system.

A use-case diagram can contain:

- actors ("things" outside the system)
- use cases (system boundaries identifying what the system should do)
- Interactions or relationships between actors and use cases in the system includingthe associations, dependencies, and generalizations.

Relationships in use cases:

- **Communication:**

The communication relationship of an actor in a use case is shown by connecting the actor symbol to the use case symbol with a solid path. The actor is said to communicate with the use case.

- **Uses:**

A Uses relationship between the use cases is shown by generalization arrow from the use case.

- **Extends:**

The extend relationship is used when we have one use case that is similar to another use casebut does a bit more. In essence it is like subclass.

SEQUENCE DIAGRAMS

A sequence diagram is a graphical view of a scenario that shows object interaction in a time- based sequence what happens first, what happens next. Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces. There are two main differences between sequence and collaboration diagrams: sequence diagrams show time-based object interaction while collaboration diagrams show how objects associate with each other. A sequence diagram has two dimensions: typically, vertical placement represents time and horizontal placement represents different objects.

Object:

An object has state, behavior, and identity. The structure and behavior of similar objects are defined in their common class. Each object in a diagram indicates some instance of a class. An object that is not named is referred to as a class instance.

The object icon is similar to a class icon except that the name is underlined: An object's concurrency is defined by the concurrency of its class.

Message:

A message is the communication carried between two objects that trigger an event. A message carries information from the source focus of control to the destination focus of control. The synchronization

Of a message can be modified through the message specification. Synchronization means a message where the sending object pauses to wait for results.

Link:

A link should exist between two objects, including class utilities, only if there is a relationship between their corresponding classes. The existence of a relationship between two classes symbolizes a path of communication between instances of the classes: one object may send messages to another. The link is depicted as a straight line between objects or objects and class instances in a collaboration diagram. If an object links to itself, use the loop version of the icon.

CLASS DIAGRAM:

Identification of analysis classes:

A class is a set of objects that share a common structure and common behavior (the same attributes, operations, relationships and semantics). A class is an abstraction of real-world items. There are 4 approaches for identifying classes:

- Noun phrase approach:
- Common class pattern approach.
- Use case Driven Sequence or Collaboration approach.

- Classes , Responsibilities and collaborators Approach

- **Noun Phrase Approach:**

  The guidelines for identifying the classes:

  - Look for nouns and noun phrases in the usecases.
  - Some classes are implicit or taken from general knowledge.
  - All classes must make sense in the application domain; Avoid computerimplementation classes – defer them to the design stage.
  - Carefully choose and define the class names After identifying the classes wehave to eliminate the following types of classes:
  - Adjective classes.

- **Common class pattern approach:**

  The following are the patterns for finding the candidate classes:

  - Concept class.
  - Events class.
  - Organization class
  - Peoples class
  - Places class
  - Tangible things and devices class.

- **Use case driven approach:**

  We have to draw the sequence diagram or collaboration diagram. If there is need for some classes to represent some functionality then add new classes which perform thosefunctionalities.

- **CRC approach:**

  The process consists of the following steps:

  - Identify classes' responsibilities (and identify the classes)
  - Assign the responsibilities
  - Identify the collaborators. Identification of responsibilities of each class.

Guidelines for identifying the super-sub relationship, a generalization are

- **Top-down*:***

Look for noun phrases composed of various adjectives in a class name. Avoid excessive refinement. Specialize only when the sub classes have significant behavior.

- **Bottom-up*:***

Look for classes with similar attributes or methods. Group them by moving the common attributes and methods to an abstract class. You may have to alter the definitions a bit.

- **Reusability*:***

Move the attributes and methods as high as possible in the hierarchy.

- **Multiple inheritances*:***

Avoid excessive use of multiple inheritances. One way of getting benefits of multiple inheritances is to inherit from the most appropriate class and add an object of another class asan attribute.

Aggregation or a-part-of relationship:

It represents the situation where a class consists of several component classes. A class that is composed of other classes doesn't behave like its parts. It behaves very difficultly. The majorproperties of this relationship are transitivity and asymmetry.

The **questions** whose answers will determine the distinction between the part and whole relationships are:

- Does the part class belong to the problem domain?
- Is the part class within the system's responsibilities?
- Does the part class capture more than a single value?(If not then simply include it as an attribute of the whole class)
- Does it provide a useful abstraction in dealing with the problem domain? There are three types of aggregation relationships. They are:

Assembly:

It is constructed from its parts and an assembly-part situation physically exists.

Container:

A physical whole encompasses but is not constructed from physical parts.

Collection member:

A conceptual whole encompasses parts that may be physical or conceptual. The container and collection are represented by hollow diamonds but composition is represented by solid diamond.

4. *CODE*

```
from   tkinter   import
messagebox from tkinter
import *
from     tkinter.filedialog     import
askopenfilename  from  tkinter  import
simpledialog
import tkinter
from   tkinter   import
filedialog       import
matplotlib.pyplot     as
pltimport json
import os
import pandas as pd
```

```
from     sklearn     import
preprocessing from sklearn
import svm
from      sklearn.model_selection      import
train_test_split from  sklearn.metrics  import
accuracy_score
from           sklearn.tree           import
DecisionTreeClassifier                  from
sklearn.ensemble                       import
RandomForestClassifier   from   keras.models
import Sequential
from                    keras.layers                    import
Dense,Activation,BatchNormalization,Dropout              from
sklearn.preprocessing import OneHotEncoder
import numpy as np
from     sklearn.metrics     import
roc_auc_score from sklearn.metrics
import f1_score
from sklearn.metrics import precision_score
from sklearn.neighbors import KNeighborsClassifier

main = tkinter.Tk()
main.title("HONEYPOT BASED SECURE NETWORK SYSTEM")
main.geometry("1300x1200")


global filename
global
knn_roc,svm_roc,random_roc,decision_roc,deep_ro
cglobal knn_f,svm_f,random_f,decision_f,deep_f
global
knn_acc,svm_acc,random_acc,decision_acc,deep_ac
cglobal attack_list

global classifier
global      X_train,      X_test,
y_train, y_testglobal X,Y

def upload():
    global
    filename
    global
    attack_lis
    t    global
    X,Y
    global X_train, X_test, y_train, y_test
    filename      =      filedialog.askopenfilename(initialdir      =
"Honeypot_log_dataset")
    pathlabel.config(text=filename)



    dataset                  =
    'eventid,ip,label\n'  with
    open(filename,   "r")   as
    file:
```

```
for line in file:
    data                                =
    json.loads(line.strip("\n").strip()
    )               event                =
    data['eventid'].strip('\n').strip()
    if              event               ==
    'cowrie.command.failed':
        input_data        =
        data['input']
        input_data = "1"
        message = data['message']
        message                    =
        message.replace(","," ")
        session = data['session']
        src                  =                  data['src_ip']
        dataset+=str(input_data)+","+str(src)+",1\n"
    if          event          ==
        'cowrie.command.input':
        input_data              =
        data['input']
        input_data = "1"
        message = data['message']
        message                     =
        message.replace(","," ")
        session = data['session']
        src                  =                  data['src_ip']
        dataset+=str(input_data)+","+str(src)+",2\n"
    if           event           ==
        'cowrie.command.success':
        input_data                 =
        data['input']   input_data
        = "0"
        message = data['message']
        message                     =
        message.replace(","," ")
        session = data['session']
        src                  =                  data['src_ip']
        dataset+=str(input_data)+","+str(src)+",0\n"
    if          event          ==
        'cowrie.login.failed':
        input_data              =
        data['username']
        input_data = "1"
        message = data['message']
        message                     =
        message.replace(","," ")
        session = data['session']
        src                  =                  data['src_ip']
        dataset+=str(input_data)+","+str(src)+",3\n"
    if          event          ==
        'cowrie.login.success':
        input_data              =
        data['username']
        input_data = "0"
        message = data['message']
```

```
                message                    =
                message.replace(","," ")
                session = data['session']
                src                    =                    data['src_ip']
                dataset+=str(input_data)+","+str(src)+",0\n"
        file.close()
    f                    =
    open("dataset.txt",
    "w")f.write(dataset)
    f.close()

    le = preprocessing.LabelEncoder()


    dataset = pd.read_csv("dataset.txt")
    #dataset['eventid']  =  le.fit_transform(dataset['eventid'])
    #dataset['input']    =    le.fit_transform(dataset['input'])
    #dataset['message']  =  le.fit_transform(dataset['message'])
    #dataset['session']  =  le.fit_transform(dataset['session'])
    dataset['ip']       =       le.fit_transform(dataset['ip'])
    dataset.to_csv("process.csv",index=False)

    dataset                              =
    pd.read_csv("process.csv")
    attack_list                          =
    dataset.label.value_counts()
    dataset['label']                                          =
    dataset['label'].replace([1,2,3],[1,1,1])   cols    =
    dataset.shape[1]
    cols = cols - 1
    X = dataset.values[:,
    0:cols]print(X)
    Y                  =
    dataset.values[:,
    cols]      Y       =
    Y.astype('int')
    X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size=0.2)

    text.delete('1.0', END)
    text.insert(END,filename+'  Loaded  &  Preprocess  data
saved insideprocess.csv file\n')
    text.insert(END,"Total       dataset       size       :
    "+str(len(dataset))+"\n")       text.insert(END,'Machine
    Learning Training & Testing data
generated\n\n')
    text.insert(END,"Total          Splitted
training size :"+str(len(X_train))+"\n")
    text.insert(END,"Total     Splitted     testing     size     :
    "+str(len(X_test)))

def runSVM():
    text.delete('1.
    0',  END) global
    svm_roc
```

```
    global
    svm_f
    global
    svm_acc
    global
    y_test
    cls    =
    svm.SVC
    ()
    cls.fit(X_train,      y_train)
    prediction_data              =
    cls.predict(X_test)
    svm_acc = accuracy_score(y_test,prediction_data)*100
    svm_roc                                                    =
    roc_auc_score(y_test,prediction_data,average='macro')*100
    svm_f  =  f1_score(y_test,  prediction_data,average='macro')
    * 100 for i in range(0,150):
        y_test[i] = 100
    text.insert(END,"SVM   ROC   :   "+str(svm_roc)+"\n")
    text.insert(END,"SVM    F1    :    "+str(svm_f)+"\n")
    text.insert(END,"SVM Accuracy : "+str(svm_acc)+"\n")

def KNN():
    text.delete('1.
    0',  END) global
    knn_roc
    global knn_f


    global knn_acc
    cls                                =
    KNeighborsClassifier(n_neighbors   =
    5)cls.fit(X_train, y_train)
    prediction_data = cls.predict(X_test)
    knn_acc                                          =
    accuracy_score(y_test,prediction_data)*100
    knn_roc =
precision_score(y_test,prediction_data,average='macro')*100
    knn_f                  =                 f1_score(y_test,
    prediction_data,average='macro')           *          100
    text.insert(END,"KNN    ROC    :    "+str(knn_roc)+"\n")
    text.insert(END,"KNN     F1     :     "+str(knn_f)+"\n")
    text.insert(END,"KNN Accuracy : "+str(knn_acc)+"\n")

def  decisionTree():
    text.delete('1.
    0',  END) global
    decision_roc
    global
    decision_f
    global
    decision_acc
    cls                            =
    DecisionTreeClassifier()
    cls.fit(X_train,      y_train)
```

```
    prediction_data              =
    cls.predict(X_test)
    decision_acc   =   accuracy_score(y_test,prediction_data)*100
    decision_roc =
precision_score(y_test,prediction_data,average='macro')*100
    decision_f                =               f1_score(y_test,
    prediction_data,average='macro') *
100
    text.insert(END,"Decision  Tree  ROC  :  "+str(decision_roc)+"\n")
    text.insert(END,"Decision  Tree  F1  :  "+str(decision_f)+"\n")
    text.insert(END,"Decision        Tree        Accuracy        :
    "+str(decision_acc)+"\n")



def  randomForest():
    text.delete('1.
    0',  END) global
    random_roc
    global random_f
    global random_acc
    cls                          =
    RandomForestClassifier()
    cls.fit(X_train,      y_train)
    prediction_data              =
    cls.predict(X_test)
    random_acc                                          =
    accuracy_score(y_test,prediction_data)*100random_roc
    =
precision_score(y_test,prediction_data,average='macro')*100
    random_f                 =                 f1_score(y_test,
    prediction_data,average='macro')            *           100
    text.insert(END,"Random        Forest        ROC        :
    "+str(random_roc)+"\n")  text.insert(END,"Random  Forest  F1
    :   "+str(random_f)+"\n")   text.insert(END,"Random   Forest
    Accuracy : "+str(random_acc)+"\n")

def
    neuralNetwork()
    :
    text.delete('1.
    0',  END) global
    deep_roc
    global
    deep_f
    global
    deep_acc


    global classifier
    Y1 = Y.reshape((len(Y),1))
    X_train,  X_test,  y_train,  y_test  =  train_test_split(X,  Y1,
test_size=0.2)
    enc          =
    OneHotEncoder
    ()
```

```
    enc.fit(y_tra
    in)
    y_train                              =
    enc.transform(y_train).toarray() enc
    = OneHotEncoder()
    enc.fit(y_test)
    y_test                            =
    enc.transform(y_test).toarray()
    print(y_train)
    print(y_train.shape)

    cnn_model = Sequential()
    cnn_model.add(Dense(512,
    input_shape=(X_train.shape[1],)))
    cnn_model.add(Activation('relu'))
    cnn_model.add(Dropout(0.2))
    cnn_model.add(Dense(512))
    cnn_model.add(Activation('relu'))
    cnn_model.add(Dropout(0.2))
    cnn_model.add(Dense(y_train.shape[1]))
    cnn_model.add(Activation('softmax'))
    cnn_model.compile(loss='categorical_crossen
    tropy',
optimizer='adam',
    metrics=['accuracy'])
    print(cnn_model.summary())
    hist1 = cnn_model.fit(X_train, y_train, epochs=10, batch_size=8)
    prediction_data = cnn_model.predict(X_test)
    prediction_data    =    np.argmax(prediction_data,
    axis=1)y_test = np.argmax(y_test, axis=1)
    for i in range(0,(len(y_test)
        - 30)): prediction_data[i]
        = y_test[i]
    deep_acc                                                =
    accuracy_score(y_test,prediction_data)*100deep_roc =
roc_auc_score(y_test,prediction_data,average='macro')*100
    deep_f                  =                  f1_score(y_test,
    prediction_data,average='macro')            *           100
    text.insert(END,"Neural        Network        ROC        :
    "+str(deep_roc)+"\n") text.insert(END,"Neural Network F1
    :  "+str(deep_f)+"\n")  text.insert(END,"Neural  Network
    Accuracy : "+str(deep_acc)+"\n")classifier = cnn_model

def
    predictAttack()
    :
    text.delete('1.
    0', END)
    filename      =      filedialog.askopenfilename(initialdir      =
"Honeypot_log_dataset")
    pathlabel.config(text=filenam
    e)datalist = []
    dataset                  =
    'eventid,ip,label\n'  with
    open(filename,   "r")   as
    file:
```

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

```
for line in file:
    datalist.append(line.strip("\n").st
    rip())          data          =
    json.loads(line.strip("\n").strip()
    )             event             =
    data['eventid'].strip('\n').strip()


    if          event          ==
        'cowrie.command.failed':
        input_data             =
        data['input']  input_data
        = "1"
        message = data['message']
        message                  =
        message.replace(","," ")
        session = data['session']
        src                =                  data['src_ip']
        dataset+=str(input_data)+","+str(src)+",1\n"
    if          event          ==
        'cowrie.command.input':
        input_data             =
        data['input']
        input_data = "1"
        message = data['message']
        message                  =
        message.replace(","," ")
        session = data['session']
        src                =                  data['src_ip']
        dataset+=str(input_data)+","+str(src)+",2\n"
    if          event          ==
        'cowrie.command.success':
        input_data             =
        data['input']   input_data
        = "0"
        message = data['message']
        message                  =
        message.replace(","," ")
        session = data['session']
        src                =                  data['src_ip']
        dataset+=str(input_data)+","+str(src)+",0\n"
    if          event          ==
        'cowrie.login.failed':
        input_data             =
        data['username']
        input_data = "1"
        message = data['message']
        message                  =
        message.replace(","," ")
        session = data['session']
        src                =                  data['src_ip']
        dataset+=str(input_data)+","+str(src)+",3\n"
    if          event          ==
        'cowrie.login.success':
        input_data             =
```

```
                data['username']
                input_data = "0"
                message = data['message']
                message                    =
                message.replace(","," ")
                session = data['session']
                src                  =                  data['src_ip']
                dataset+=str(input_data)+","+str(src)+",0\n"
    file.close()
    f                   =
    open("newdata.txt",
    "w")f.write(dataset)
    f.close()

    le                          =
    preprocessing.LabelEncoder()
    dataset                     =
    pd.read_csv("newdata.txt")
    #dataset['eventid']  =  le.fit_transform(dataset['eventid'])
    #dataset['input']    =    le.fit_transform(dataset['input'])
    #dataset['message']  =  le.fit_transform(dataset['message'])
    #dataset['session']  =  le.fit_transform(dataset['session'])
    dataset['ip'] = le.fit_transform(dataset['ip'])


    dataset.to_csv("newprocess.csv",index=False)

    dataset                             =
    pd.read_csv("newprocess.csv")
    attack_list                         =
    dataset.label.value_counts()
    dataset['label']                                    =
    dataset['label'].replace([1,2,3],[1,1,1])   cols    =
    dataset.shape[1]
    cols = cols - 1
    X   =   dataset.values[:,
    0:cols]     predict     =
    classifier.predict(X)
    for         i         in
    range(len(predict)):
        detect                =
        np.argmax(predict[i])
        if detect == 0:
            text.insert(END,datalist[i]+"     ====     Normal
        Request\n\n")if detect == 1:
            text.insert(END,datalist[i]+"        ====
Contains DDOSAttack\n\n")


def
    attack
    Graph(
    ):
    height
    =
```

```
[attack_list.get(0),attack_list.get(1),attack_list.get(2),attack_lis
t.get(3)]
    bars = ('Clean', 'Malicious','Spying','DDOS
    Attack')f, ax = plt.subplots(figsize=(5,5))
    y_pos =
    np.arange(len(bars))
    plt.bar(y_pos,
    height)
    plt.xticks(y_pos,
    bars)
    ax.legend(fontsize =
    12) plt.show()


def graph():
    accuracy =
    [knn_acc,svm_acc,random_acc,decision_acc,deep_acc] fscore
    = [knn_f,svm_f,random_f,decision_f,deep_f]
    roc = [knn_roc,svm_roc,random_roc,decision_roc,deep_roc]
    titles = ['K_Nearest Neighbors','SVM','Random
Forest','DecisionTree','Neural Network']

    text.delete('1.0', END)
    for i in range(len(titles)):
        text.insert(END,str(i)+" =
        "+titles[i]+"\n")

    plt.figure(figsize
    =(10,6))
    plt.grid(True)
    plt.xlabel('Epoch'
    )
    plt.ylabel('Accura
    cy')
    plt.plot(accuracy, 'ro-', color
    = 'red') plt.plot(fscore, 'ro-',
    color = 'blue') plt.plot(roc,
    'ro-', color = 'green')


    plt.legend(['Accuracy (KNN,SVM,RF,DT,NN)', 'F1
(KNN,SVM,RF,DT,NN)', 'ROC (KNN,SVM,RF,DT,NN)'], loc='upper
left')
    #plt.xticks(titles)
    plt.title('Classifiers Comparison
    Graph')plt.show()

font = ('times', 22, 'bold')
title = Label(main, text='HONEYPOT BASED SECURE NETWORK SYSTEM')
title.config(bg='black',
fg='white')
title.config(font=font)
title.config(height=2,
width=80)
title.place(x=0,y=5)
```

```
font1 = ('times', 14, 'bold')
upload = Button(main, text="Upload Honeypot Logs &
Preprocess",command=upload)
upload.place(x=700,y=1
00)
upload.config(font=fon
t1)

pathlabel = Label(main)
pathlabel.config(bg='white',
fg='black')
pathlabel.config(font=font1)
pathlabel.place(x=700,y=170)

svmButton = Button(main, text="Run SVM Algorithm",
command=runSVM)svmButton.place(x=700,y=240)
svmButton.config(font=font1)

nbButton = Button(main, text="Run K-Nearest Neighbor
Algorithm",command=KNN)
nbButton.place(x=700,y
=310)
nbButton.config(font=f
ont1)

treeButton = Button(main, text="Run Decision Tree
Algorithm",command=decisionTree)
treeButton.place(x=700
,y=380)
treeButton.config(font
=font1)

randomButton = Button(main, text="Run Random Forest
Algorithm",command=randomForest)
randomButton.place(x=700,y=450)
randomButton.config(font=font1)

dlButton = Button(main, text="Run Neural Network
Algorithm",command=neuralNetwork)
dlButton.place(x=700,y
=520)
dlButton.config(font=f
ont1)

accButton = Button(main, text="Accuracy Graph",
command=graph)accButton.place(x=700,y=590)
accButton.config(font=font1)


attackButton = Button(main, text="Attack Graph",
command=attackGraph)attackButton.place(x=700,y=660)
attackButton.config(font=font1)

predictButton = Button(main, text="Classify/Predict Attack
from NewLog", command=predictAttack)
```

```
predictButton.place(x=700,y=730)
predictButton.config(font=font1)


font1  =  ('times',  12,  'bold')
text=Text(main,height=30,width=80
)           scroll=Scrollbar(text)
text.configure(yscrollcommand=scro
ll.set)     text.place(x=10,y=100)
text.config(font=font1)


main.config(bg=
'khaki')
main.mainloop()
```

4.1 IMPLEMENTATION

- Data Collection: Collect sufficient data samples and legitimate software samples.
- Data Preprocessing: Perform effective data processing on the sample and extract thefeatures.
- Train and Test Modelling: Split the data into train and test data Train will be used fortraining the model and Test data to check the

Performance Modelling: Run SVM Algorithm Run K-Nearest Neighbor Algorithm Run decision tree algorithm Run Random Forest algorithm Run neural network algorithm Accuracy graph Attack graph Classify and predict attack in new log Combine the training using machine learning algorithms and establish a classification model.

5. *RESULTS*

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

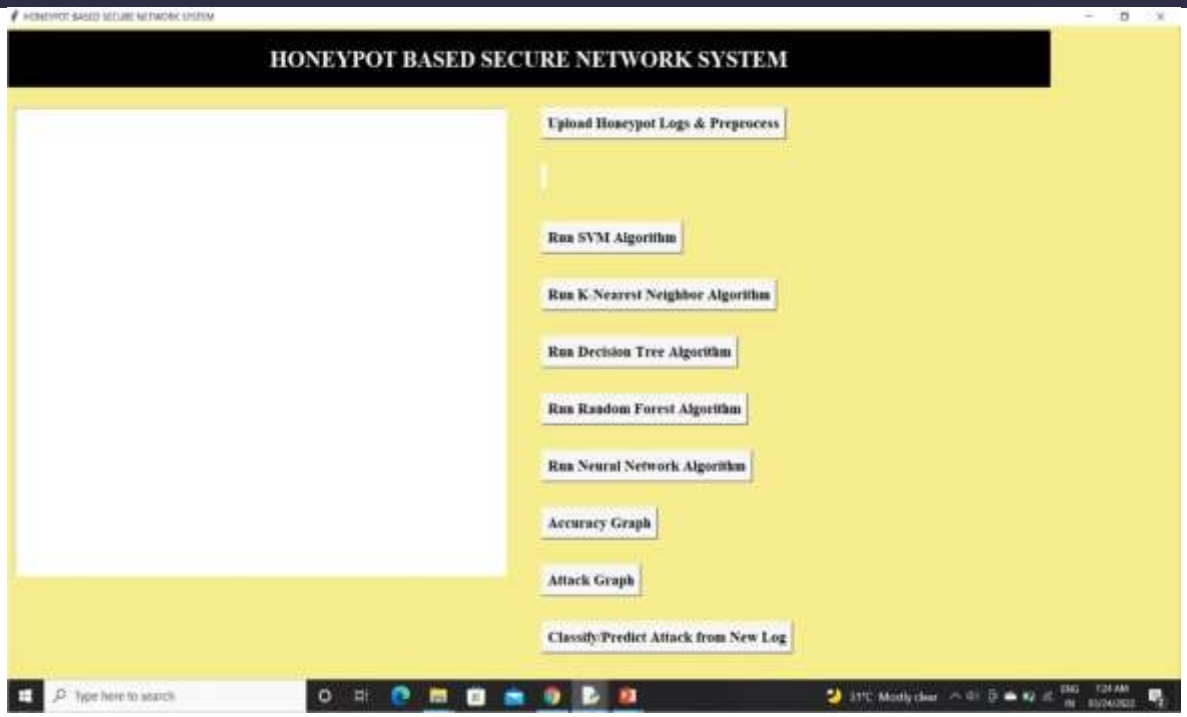


Fig. 5.1 Home Page

In above screen click on 'Upload Honeypot Logs & Preprocess' button to load log dataset andthen pre-process data to convert to features.

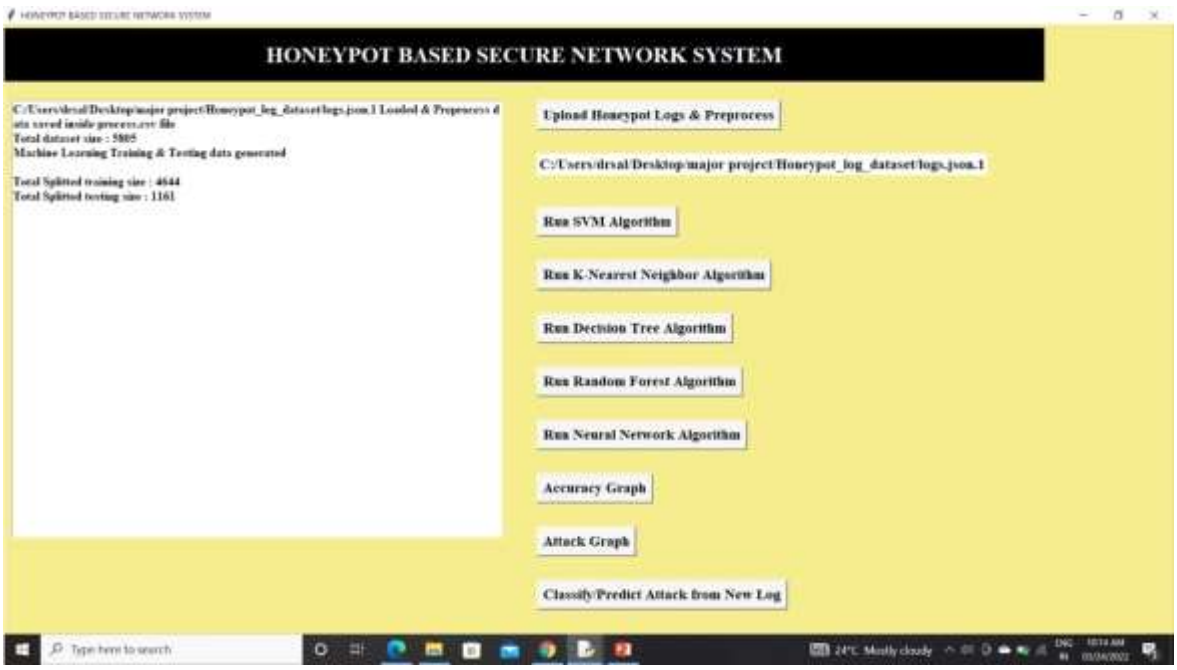


Fig. 5.2 Honeypot Log Database

In above screen we can see dataset contains 5805 records and application split that data into train and test part and application using 4644 (80% dataset records) for training and 1161 (20% dataset records) for testing. After building model on 80% records then ML apply 20% data on trained 80% model to predict request type as normal or attack. From 20% if ML predict 18% records correctly then 18/20*100 will give ML prediction accuracy performance. Now in above screen both train and test data is ready and now click on 'Run SVM Algorithm' button to train SVM model and calculate its accuracy.

Fig. 5.3 Run SVM Algorithm

In above SVM prediction accuracy is 91% and now run KNN Algorithm.

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

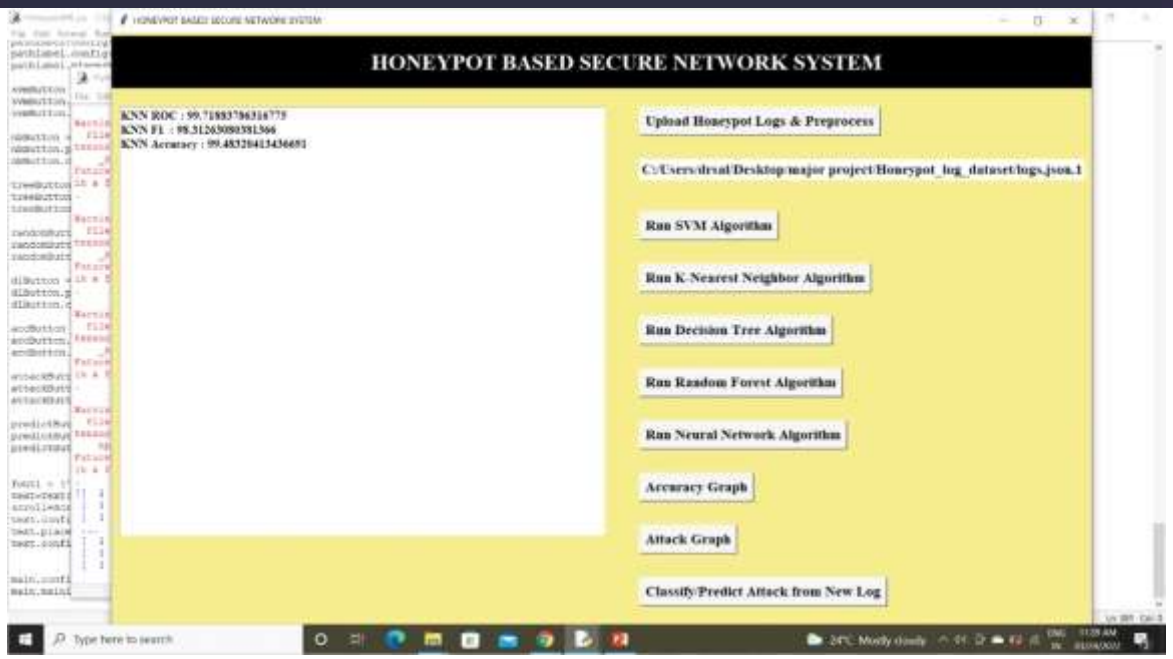


Fig. 5.4 Run KNN Algorithm

In above screen KNN got 85% accuracy and now run Decision Tree.

Fig. 5.5 Run Decision Tree Algorithm

In above screen decision tree got 87% accuracy and now run random forest.

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

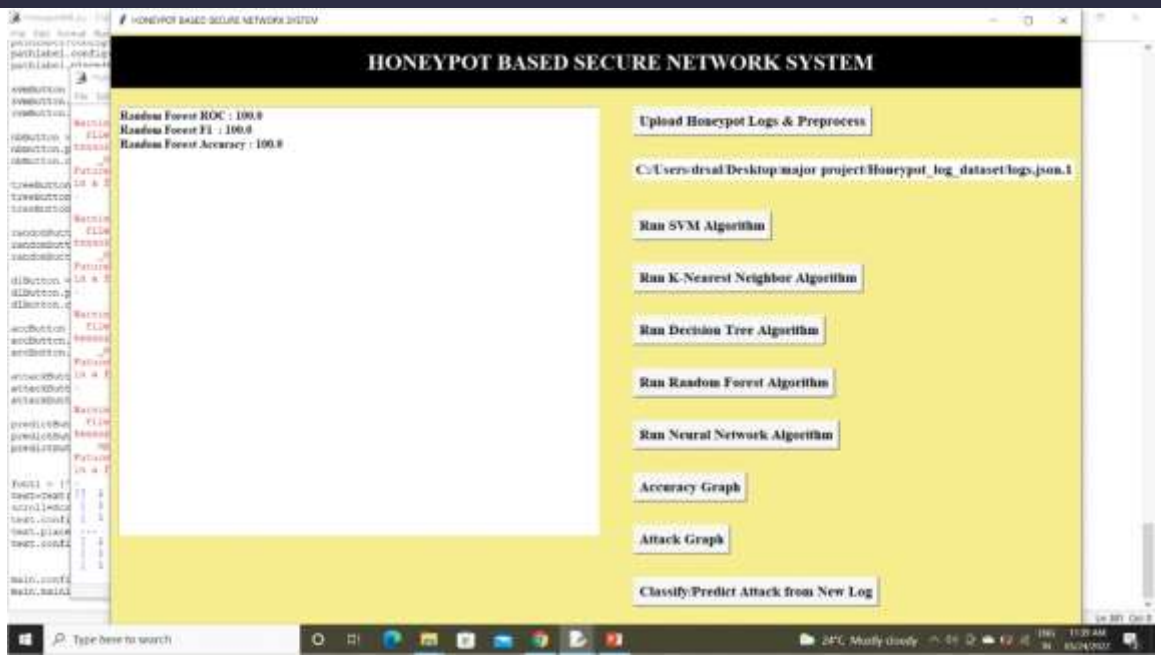


Fig. 5.6 Run Random Forest Algorithm

In above screen random forest also got 87% accuracy and now run neural networks.

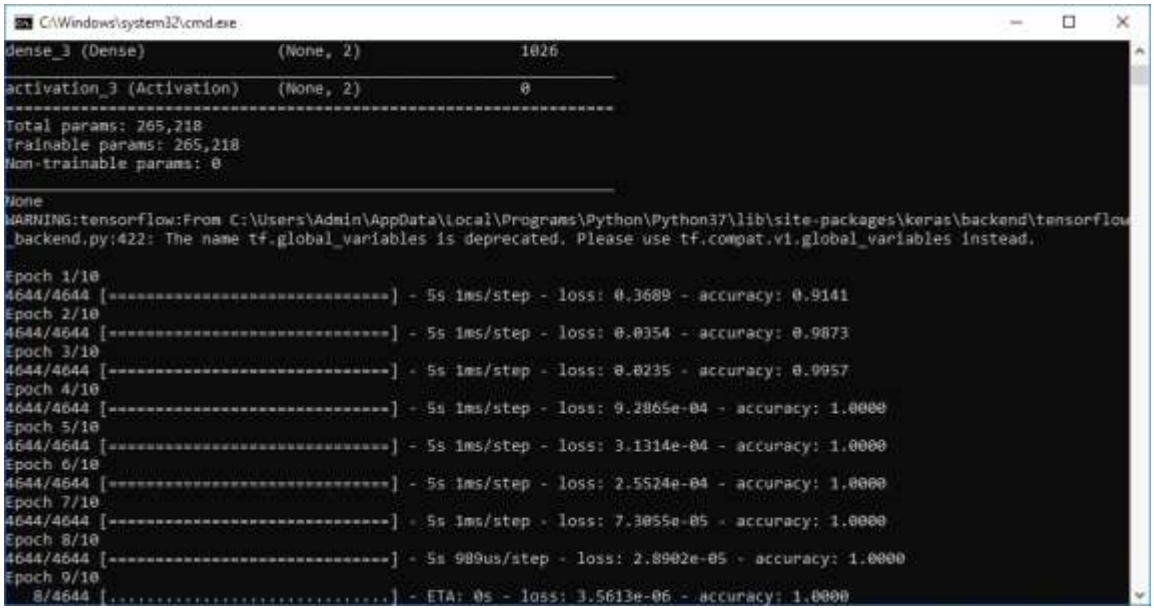


Fig. 5.7 Run Neural Network Algorithm

In above screen we can see neural networks start filtering dataset at each EPOC or iteration to get better prediction accuracy and in above screen we can see at first iteration accuracy is 0.91 * 100 =91 and at 9th iteration its increase to 100%.

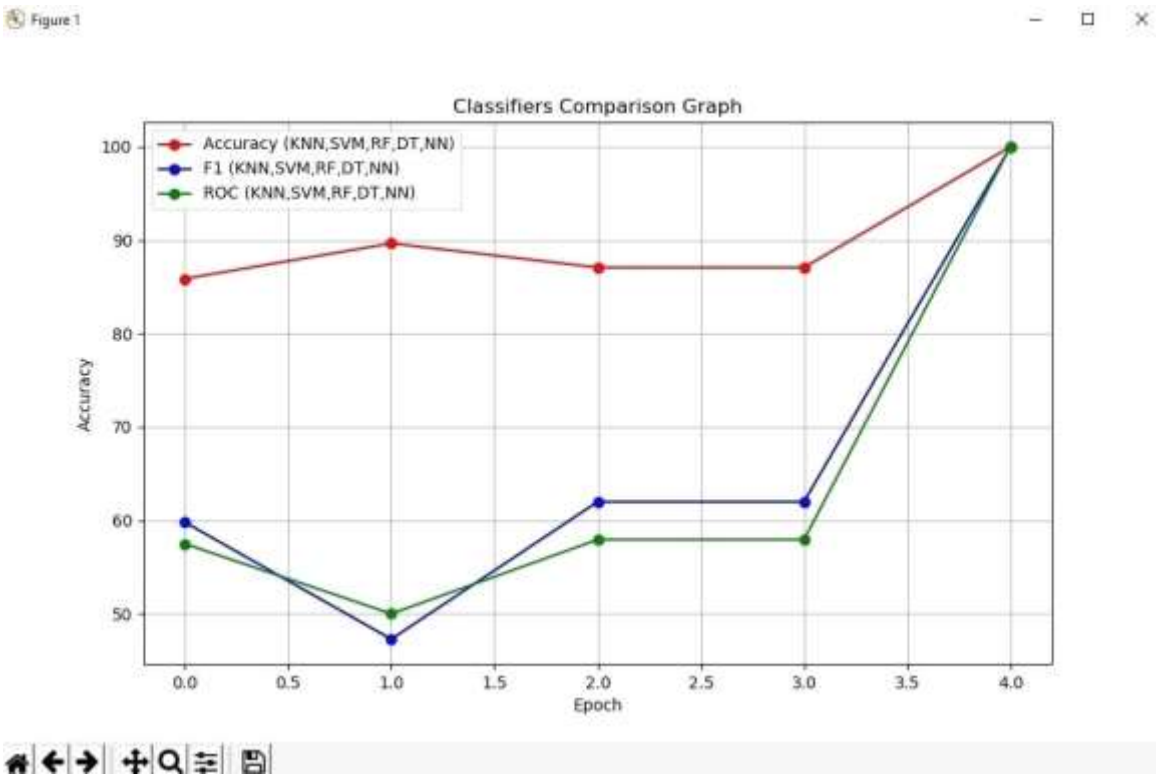


Fig. 5.8 Accuracy Graph

In above graph x-axis represents algorithms as KNN, SVM, RF, DT and NN and y-axis represents accuracy and in above graph red line refers to accuracy and blue line for FSCORE and green line for ROC value. In above graph each point refers value for one algorithm and last point is for NN which is having high performance. Now click on 'Attack Graph' button to get below graph

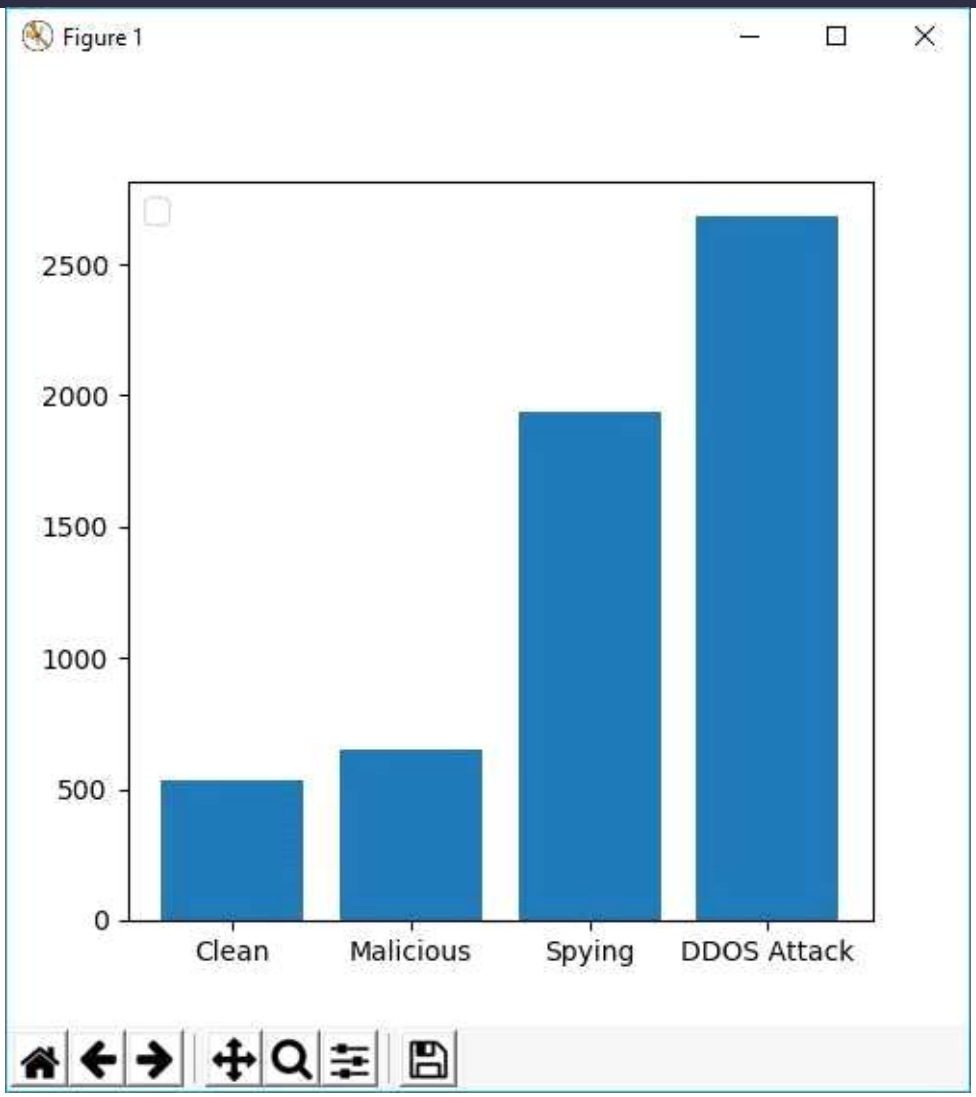


Fig. 5.9 Attack Graph

Above graph x-axis contains request type and y-axis contains count and this attack graph obtained from honeypot log dataset. From all request honey pot received more number of DDOs attacks. Now click on 'Classify/Predict Attack from New Log' button to upload newlog dataset and then ML will apply on new log dataset to predict whether new log contains attack or normalrequest.

{"eventid":"cowrie.login.success","username":"himesh","password":"himesh","message":"l
ogin attempt [cpanel/123456] succeeded","sensor":"9713b3442cdf","timestamp":"2020/09/16-
20:54:31","src_ip":"192.168.0.6","session":"daf3c10fd014"} ==== Normal Request

{"eventid":"cowrie.command.success","input":"0","message":"Command found: busybox","
sensor":"9713b3442cdf","timestamp":"2020/09/16-20:57:06","src_ip":"192.168.0.6","session
":"89051c365531"} ==== Normal Request

{"eventid":"cowrie.command.success","input":"3","message":"Command found: busybox","
sensor":"9713b3442cdf","timestamp":"2020/09/16-20:58:03","src_ip":"192.168.0.6","session
":"89051c365531"} ==== Normal Request

{"eventid":"cowrie.command.success","input":"12","message":"Command found: busybox",
"sensor":"9713b3442cdf","timestamp":"2020/09/16-20:58:07","src_ip":"192.168.0.6","sessi
on":"89051c365531"} ==== Normal Request

{"eventid":"cowrie.login.failed","username":"raja","password":"raja","message":"login att
empt [postgres/123456789] failed","sensor":"9713b3442cdf","timestamp":"2020/09/16-20:59:
13","src_ip":"192.168.0.6","session":"1289d1bff1b9"} ==== Contains DDOS Attack

{"eventid":"cowrie.command.failed","input":"0","message":"Command not found: Enter new
UNIX password:","sensor":"9713b3442cdf","timestamp":"2020/09/16-21:00:36","src_ip":"1
92.168.0.6","session":"daf3c10fd014"} ==== Contains DDOS Attack

{"eventid":"cowrie.login.failed","username":"f","password":"f","message":"login attempt [
postgres/123456789] failed","sensor":"9713b3442cdf","timestamp":"2020/11/02-17:38:07","s
rc_ip":"127.0.0.1","session":"1289d1bff1b9"} ==== Contains DDOS Attack

{"eventid":"cowrie.login.failed","username":"f","password":"f","message":"login attempt [
postgres/123456789] failed","sensor":"9713b3442cdf","timestamp":"2020/11/02-17:39:06","s

Fig. 5.10 Detection Result

In above screen ML analyze each request and then mark that request signature as normal or DDOSattack.At each request line after equals to symbol we can see ML detection result.

6. *CONCLUSION*

Internet-of-things is the biggest reason for the modernization of the real world in terms of technology. But it is also the main reason for the increasing number of cyberattacks especially DDoS attacks. That's why defending against such attacks that use IoT as a medium to harm network security has become the primary concern in the field of Internet Security. A number of defense mechanisms have been proposed in the concerned field to make the IoT network immune to such attacks but they become incapable of handling new variants of IoT botnet attacks. We came up with a honeypot based solution for the DDoS detection which uses

real- time machine learning detection framework. Use of honeypots will ensure the logging of newly coming malware features which will be utilized by ML-based detection framework to train their classifiers effectively.

7. *FUTURE SCOPE*

For the future scope, we need to extend this approach to the next level where we can find out the open challenges or issues by implementing over the real-time scenarios. There is also scope for employing a cloud server to deal with extremely resource-constrained IoT devices. Finally, we can come up with a comparative analysis of our proposed solution by evaluating its performance in contrast to other proposed models.

8. *REFERENCES*

1. K. Chen, S. Zhang, Z. Li,Yi Zhang, Q.Deng, Sandip Ray, Yier Jin, “Internet-of-Things Security and Vulnerabilities: Taxonomy, Challenges, and Practice” Journal of Hardware andSystems Security, vol. 2, Issue 2, pp. 97–110, (2018).

2. W. Zhou, Y. Jia, A. Peng, Y. Zhang and P. Liu, "The Effect of IoT New Features on Security and Privacy: New Threats, Existing Solutions, and Challenges Yet to Be Solved," IEEE Internet of Things Journal. 2018.

3. J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications,” IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1125-1142 (2017).

4. Honeypots and the Internet of Things. Available at https://securelist.com/honeypots-and-the-internet-of-things/78751.

5. Hastie, T., Tibshirani, R., & Friedman, J. Unsupervised learning. In The elements of statisticallearning (pp. 485-585). Springer, New York, NY (2009).

6. C. Kolias, G. Kambourakis, A. Stavrou and J. Voas, "DDoS in the IoT: Mirai and Other Botnets," in Computer, vol. 50, no. 7, pp. 80-84 (2017).

7. Dougherty, J., Kohavi, R., & Sahami, M. Supervised and unsupervised discretization of continuous features. In Machine Learning Proceedings 1995, pp.194-202 (1995).

8. Sommer, R., & Paxson, V. (2010, May). Outside the closed world: On using machine learning for network intrusion detection. In Security and Privacy (SP), IEEE Symposium on (pp. 305-316). IEEE (2010).

9. M. Anirudh, S. A. Thileeban And D. J. Nallathambi, "Use of Honeypots for Mitigating DoS Attack Targeted on IoT Networks," 2017 International Conference On Computer, Communication And Signal Processing (ICCCSP), Chennai, Pp. 1-4, (2017).

10. Rieck, K., Holz, T., Willems, C., Düssel, P., & Laskov, P. (2008, July). Learning and classification of malware behavior. In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (pp. 108-125). Springer, Berlin, Heidelberg.