

P2P Botnet Detection Using Machine Learning Algorithm: The Tools

Blessing Iduh, Raphael Okonkwo, Obiajulu Ositanwosu, Obinna Iwegbuna

Abstract— Creating Botnet detection systems have become very imperative, due to the continued creation of newer Botnet toolkits cyber criminals. A Botnet is a network of compromised computerized devices that are connected to a central controller called a Botmaster. These devices are usually used to carry out malicious activities like identity theft, sending of spam mails, DOS attacks and other damaging acts without the knowledge of the actual owner of the device. Botnet detection using advanced techniques has become very necessary as Botmasters continue to device new means of attack. This paper therefore, presents some relevant tools and procedures involved in creating a Botnet detection system, and how to apply these tools using machine learning algorithms. This paper also shows the steps involved in applying these tools.

Index Terms— Botnet, Botmaster, Botnet Detection, Machine Learning, Cyber Security, C&C Channel, Malware, Decision Tree Classifier

1 INTRODUCTION

Botnets have in recent times, become a very major challenge in the cyberspace. The word Botnet according to [1] is a combination of the words roBot and NETwork. It is used to describe a group of compromised computer systems that are usually connected to a central controller called a Botmaster. The Botmaster uses command and control channels, to manipulate these infected computers. The difference between Botnets and other malwares according to [2], is the use of command and control (C&C) channels by Botnets. The C&C channel allows Bots to receive commands and perform malicious activities. A single infected system is known as a Bot, while a network of infected devices is referred to as a Botnet. Botnets are created by the Botmaster for communication infrastructure to perform malicious activities like email spamming, click fraud, identity theft, phishing attacks, denial of service attacks, information theft and distributed denial of service attacks. Systems that are connected to the internet have the chances of getting infected and become part of a Botnet. According to [3], in their Survey of HTTP Botnet, a Botnet was described as a group of cooperated computers which are remotely controlled by hackers to launch various network attacks, such as DDoS attack, junk mail, click fraud, identity theft and information phishing. [4], in their Overview of Peer-to-Peer Botnets noted that Botnets have recently been identified as one of the most important threats to the security of the Internet. In the work of [5], they explained that Botnets have five states in their life cycle: the Injection state where the malware gets into the host system

and which is achieved when the host download the malware through e-mail, trojan software and click fraud techniques. The user of the system innocently clicks on these items and a download and infection is initiated; being the connection state where the Bot will be made to connect with the Command & Control (C&C) server; the third state being the waiting state where the Bot waits for the request from its master; the execution state where the received request is performed or treated by the Bot and Finally, the maintenance and upgrading state where the Botmaster upgrades their attacking techniques in order to bypass any detection method. A user can get infected by visiting an infected site or accessing resources from such sites. Also, an infected system on a network can infect other systems on that same network. Botnets have continued to cause serious threats to the society including private and government organizations, national infrastructure and the general internet community. Botnet detection involves the identification of Bots in the machine or network so that it can be managed. In recent years Botnet detection has been a hot topic in the research community due to increase in the malicious activity. According to [6], the key features and characteristics of Bots are considered as a critical step when dealing with Botnet detection.

2. LITERATURE REVIEW

Several researchers like [7], [8], [9], have worked on Botnet Detection and management. In general, two major approaches exist for detection of Botnets these include; the signature-based and anomaly based detection methods. Researcher like [10] and [11] among others, have studied the signature-based detection and their results are applicable for known Bots. In their approach, every packet is monitored and compared to the pre-configured signatures and attack patterns in the database. Even though their approach can detect some Botnets, the signature database needs to always be updated to detect the new Bots. In addition, Bot-masters obfuscate the Bots by novel packers to avoid detection by the signature-based approaches. Also, a signature-based Botnet detection technique uses the

- Blessing Iduh is currently pursuing PhD degree program in Data Communication and Networking at Nnamdi Azikiwe University, Awka, Anambra State, Nigeria. E-mail: bn.iduh@unizik.edu.ng
- Raphael Okonkwo is currently a Professor of computer Science, Nnamdi Azikiwe University, Awka, Anambra State, Nigeria. E-mail: ro.okonkwo@unizik.edu.ng
- Ositanwosu Obiajulu is currently pursuing PhD degree program in Information Technology, Machine learning and IOT Soochow China University, Guangzhou, PR China. E-mail: oe.ositanwosu@unizik.edu.ng
- Obinna Iwegbuna is currently pursuing PhD degree program in Information Technology at Ebonyi State University, Ebonyi State, Nigeria. E-mail: @unizik.edu.ng

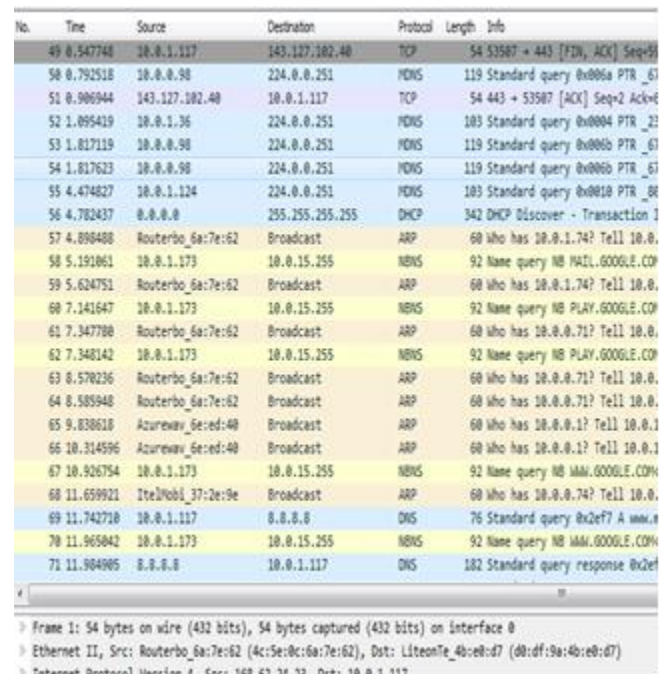
signatures of current Botnets for its detection. Other Botnet detection techniques include; anomaly based detection, network based detection, data mining base detection and Machine learning based Botnet detection techniques. In this work, the focus is on the use of machine learning algorithms for Botnet detection. [12], proposed a machine learning technique for Botnet detection that uses network statistics. These statistics involves bytes per second, packet duration per second of some protocols used for chatting such as IRC protocol. [10], presented a detection approach that examines the use of network flow characteristics like bandwidth, packet timing, and burst duration to show evidence of Botnet activity by filtering traffic that are unrelated to Botnets, to reduce the amount of data that is being processed.

3 IMPLEMENTATION

3.1. The Packet Capture Phase

This phase involves the collection of dataset for the Botnet Analysis. In this work, the dataset was collected from a campus network. The setup for the data capture is shown in fig. 1. The Wireshark software was used as the tool at this phase. The Wireshark software was installed into a host computer and connected to the enterprise network. The host needed to be on the same network that the analysis and detection is to be carried out so as to get a live capture of activities that are going on on the network at selected time frames. In this case, the traffic and activities on the network were captured for 20 Minutes. Note that the longer the time spent in capturing the data, the larger the file. And the larger the file, the heavier it is to analyze. A sample capture of the Wireshark interface is shown in figure 1. The information gathered about individual packets on the captured network include, the time of the capture, the source and destination address of the packet, the internet protocol type of the packet which shows whether it is a transmission control protocol (TCP) packet, which is the protocol that controls uploading and downloading of files on the internet; an Address Resolution Protocol Protocol (ARP) packet, which is the protocol that performs the Internet Protocol (IP) routing. ARP locates the hardware address, which is also known as the Media Access Control (MAC) address of a host from its known IP address. The Multicast Domain Name Service (MDNS) protocol as shown in figure 1, shows all packets that has to do with printing from a network printer or such network devices in the network. .

The packets were captured based on the activities going on in the network at the time of the capture, which involves the sites visited by different users in that network. The system was able to capture both benign and infected traffics on the network for analysis.

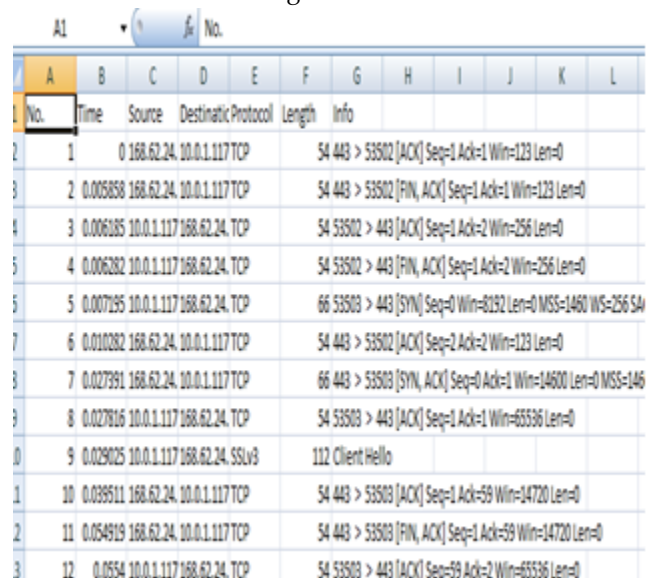


No.	Time	Source	Destination	Protocol	Length	Info
49	0.347748	10.0.1.117	143.127.182.40	TCP	54	53587 → 443 [FIN, ACK] Seq=...
50	0.792518	10.0.0.98	224.0.0.251	MDNS	119	Standard query 0x000a PTR _61...
51	0.906944	143.127.182.40	10.0.1.117	TCP	54	443 → 53587 [ACK] Seq=2 Ack=...
52	1.095419	10.0.1.36	224.0.0.251	MDNS	103	Standard query 0x0004 PTR _23...
53	1.817119	10.0.0.98	224.0.0.251	MDNS	119	Standard query 0x000b PTR _63...
54	1.817623	10.0.0.98	224.0.0.251	MDNS	119	Standard query 0x000b PTR _63...
55	4.474827	10.0.1.124	224.0.0.251	MDNS	103	Standard query 0x0010 PTR _80...
56	4.782437	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction 1...
57	4.894888	Routerbo_6a:7e:62	Broadcast	ARP	60	Who has 10.0.1.74? Tell 10.0....
58	5.191861	10.0.1.173	10.0.15.255	NBNS	92	Name query NB MAIL.GOOGLE.CO...
59	5.624751	Routerbo_6a:7e:62	Broadcast	ARP	60	Who has 10.0.1.74? Tell 10.0....
60	7.141647	10.0.1.173	10.0.15.255	NBNS	92	Name query NB PLAY.GOOGLE.CO...
61	7.347788	Routerbo_6a:7e:62	Broadcast	ARP	60	Who has 10.0.0.71? Tell 10.0....
62	7.348142	10.0.1.173	10.0.15.255	NBNS	92	Name query NB PLAY.GOOGLE.CO...
63	8.578236	Routerbo_6a:7e:62	Broadcast	ARP	60	Who has 10.0.0.71? Tell 10.0....
64	8.585948	Routerbo_6a:7e:62	Broadcast	ARP	60	Who has 10.0.0.71? Tell 10.0....
65	9.838618	Azurewav_5e:ed:40	Broadcast	ARP	60	Who has 10.0.0.1? Tell 10.0.1...
66	10.314596	Azurewav_5e:ed:40	Broadcast	ARP	60	Who has 10.0.0.1? Tell 10.0.1...
67	10.926754	10.0.1.173	10.0.15.255	NBNS	92	Name query NB MAIL.GOOGLE.CO...
68	11.659921	Te2Nobi_37:2e:9e	Broadcast	ARP	60	Who has 10.0.0.74? Tell 10.0....
69	11.742710	10.0.1.117	0.0.0.0	DNS	76	Standard query 0x2eff A www.8...
70	11.965842	10.0.1.173	10.0.15.255	NBNS	92	Name query NB MAIL.GOOGLE.CO...
71	11.984985	0.0.0.0	10.0.1.117	DNS	182	Standard query response 0x2ef...

Figure 1. Wireshark capture interface

3.2 . Flow Characteristics Extraction

This stage involves converting the captured files into an appropriate format for the analysis. The Wireshark application, captures its data in .pcap format. This format is not recognizable by the system. Therefore, it was extracted and converted to a comma delimited format (.csv) and exported to Microsoft excel. This is shown in figure 2.



A	B	C	D	E	F	G	H	I	J	K	L
No.	Time	Source	Destination	Protocol	Length	Info					
1	0	168.62.24.10.0.1.117	TCP	54	443 > 53582 [ACK] Seq=1 Ack=1 Win=123 Len=0						
2	0.005858	168.62.24.10.0.1.117	TCP	54	443 > 53582 [FIN, ACK] Seq=1 Ack=1 Win=123 Len=0						
3	0.006185	10.0.1.117	168.62.24.10.0.1.117	TCP	54	53582 > 443 [ACK] Seq=1 Ack=2 Win=256 Len=0					
4	0.006282	10.0.1.117	168.62.24.10.0.1.117	TCP	54	53582 > 443 [FIN, ACK] Seq=1 Ack=2 Win=256 Len=0					
5	0.007195	10.0.1.117	168.62.24.10.0.1.117	TCP	66	53583 > 443 [SYN] Seq=0 Win=0 Len=0 MSS=1460 WS=256 SA...					
6	0.010282	168.62.24.10.0.1.117	TCP	54	443 > 53582 [ACK] Seq=2 Ack=2 Win=123 Len=0						
7	0.027991	168.62.24.10.0.1.117	TCP	66	443 > 53583 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=146...						
8	0.027816	10.0.1.117	168.62.24.10.0.1.117	TCP	54	53583 > 443 [ACK] Seq=1 Ack=1 Win=65536 Len=0					
9	0.029025	10.0.1.117	168.62.24.10.0.1.117	SSLv3	112	Client Hello					
10	0.039511	168.62.24.10.0.1.117	TCP	54	443 > 53583 [ACK] Seq=1 Ack=59 Win=14720 Len=0						
11	0.054919	168.62.24.10.0.1.117	TCP	54	443 > 53583 [FIN, ACK] Seq=1 Ack=59 Win=14720 Len=0						
12	0.0554	10.0.1.117	168.62.24.10.0.1.117	TCP	54	53583 > 443 [ACK] Seq=59 Ack=2 Win=65536 Len=0					

Figure 2. Captured data in .csv format

3.3 . Classification Stage

The classification stage of this work was done, using Machine learning algorithms. This was achieved with the Python machine learning module. Some of the Python tools and libraries that were put together for the classification include;

i) NUMPY – Numerical Python (Numpy) is a library for the Python programming language, it has support for large, multi-dimensional arrays and matrices. Numpy has a large collection of high-level mathematical functions to operate on these arrays. It was used in this work, by python to handle the multidimensional arrays and functions that were needed for the classification.

ii) SCIKIT-LEARN – Scikit-Learn is a machine learning library for Python programming language. It was used for this work because it supports the supervised and unsupervised machine learning algorithm which were used for this work for handling encrypted C&C channels. It builds on two basic libraries of Python, NumPy and SciPy which was why we needed to first import the Numpy library to accommodate the scikit-learn. It was one of the tools used to perform the clustering, regression and classification in this system.

iii) TENSORFLOW - TensorFlow is a machine learning library, and it was used in this work as a tool to communicate with the hardware and read details from the file. With Tensorflow, after writing the codes in python, the program was compiled and run on the CPU directly. Tensorflow uses a system of multi-layered nodes that allowed the training and deployment of the system irrespective of its large datasets

iv) KERAS- keras is also a machine learning tool that works with python. It provided a high-level API. Keras is written in Python and it is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R Programming and Theano. It is designed to enhance fast experimentation with the deep neural networks. It was used in this work because it is user friendly, modular, and far reaching. It provided the building blocks to create and train the datasets.

v) THEANO- Theano is a Python library that enables the evaluation of mathematical operations as well as multi-dimensional arrays in an efficient manner. It is very useful in machine learning and deep learning.. Theano makes good use of the Graphic Processing Units (GPUs) as oppose to Central Processing Units. Theano has the ability to convert structures very efficient codes that works with Numpy. It is specially designed to take care of the types of computation that is needed for Machine Learning. Theano is for numerical computation, and is similar to NumPy. It is the tool that was used to define, optimize, and evaluate the mathematical expressions

involving multi-dimensional arrays efficiently in this system. It did all the mathematical analysis internally. This allowed it to make data-intensive calculations up to 100 times faster than when run on the CPU alone.

vi) PANDAS- Pandas is a Python library that provides high-level data structures which are simple to use as well as intuitive. It is a software that is specifically created for the Python programming language to be used for analysis and manipulation of data. Pandas specifically offers structures and operations of data that can be used to manipulate time series and numerical tables. Pandas is a free and open software. The name is gotten from term "panel data". In this paper, Pandas was used for grouping, combining and filtering the data, as well as performing time series analysis in the system. Pandas was used to fetch the dataset in Python from the CSV, Excel, JSON files and manipulated the data to perform operations on it.

vii) MATPLOTLIB – is used in Python programming environment, as the plotting library. It works together with Numpy, which is the Python numerical mathematics extension. Matplotlib, creates an object-oriented API that can be used to embed plots into an applications. Matplotlib was used for the data visualization of this work. This is because, it has the ability as a standard python library, to be used to 2D plots and graphs. It was imported in this work to create a graph of the dataset.

viii) SEABORN - Seaborn is a graphic library that is built on the platform of Matplotlib. It enables you make better charts, create better visuals for your data. Seaborn data visualization library was imported in this work. It helped to build on the Matplotlib's foundations.

ix) PICKLE - The pickle module was used to implements binary protocols for serializing and de-serializing the Python object structure. It is the process whereby a Python object hierarchy is converted into a byte stream, while unpickling is the inverse operation, and it's an operation whereby a byte stream from a binary file is converted back into an object hierarchy.

3.4. Database Development Tools

Some of the tools used for the data gathering and database development include;

- i) Wireshark
- ii) Anaconda

i. Use of the Wireshark tool

Wireshark is a network packet analyzer. It was used to capture the network packets and to display the packet data in details. The dataset used for this work was the CTU The dataset (Garcia, Grill, Stiborek, & Zunino, 2014). Some dataset was also gathered from Nnamdi Azikiwe University Network flow. The data set was sniffed from the 'nitdawireless' network 'connection2'. The wireshark packet sniffer was used to sniff the data both for analysis. The sniffing was done at five (5) different occasions to enable proper capture of both Botnet and benign traffic. The dataset were captured for different durations between 15mins to 3hours. The unizikCapture3 file had the highest amount of data as its capture time lasted for 2hours 10 minutes.

ii. Use of Ananconda

Anaconda also known as Conda, is an open-source package manager, environment manager, and distribution of the Python programming language. It was the environment on which the entire system was built. It gave the capability to import all the needed libraries that were used to carry out all the various stages of this work. Because it's designed for data science and machine learning workflows, it was the appropriate tool used to handle the large datasets used in this work.

3.5 Algorithm for the Botnet Detection System Begin

Step one - Load the data using pandas as usual:

Step two - import pandas as pd

Step three - data = pd.read_csv("capture20110816-3.binetflow.xz")

Step four - data['Label'] = data.Label.str.contains("Botnet")

Step five - Data Exploration

Step six - data.columns

Step seven - Index(['StartTime', 'Dur', 'Proto', 'SrcAddr', 'Sport', 'Dir', 'DstAddr',

'Dport', 'State', 'sTos', 'dTos', 'TotPkts', 'TotBytes', 'SrcBytes', 'Label'],

dtype='object')

step Eight - first script DataPreparation.py.

step Nine - from __future__ import division

step Ten - import os, sys

step Eleven - import threading

Step Twelve - Create class called Prepare to select training and testing data:

Step Thirteen - Class Prepare(threading.Thread):

Step fourteen - **def __init__(self, X, Y, XT, YT, accLabel=None):**

Step fifteen - threading.Thread.__init__(self)

Step Sixteen - self.X = X

Step Seventeen - self.Y = Y

Step Eighteen - self.XT=XT

Step Nineteen - self.YT=YT

Step Twenty - self.accLabel= accLabel

Step Twenty-One - def run(self):

Step Twenty - Two- X = np.zeros(self.X.shape)

Step Twenty - Three - Y = np.zeros(self.Y.shape)

Step Twenty-four- XT = np.zeros(self.XT.shape)

Step Twenty- Five- YT = np.zeros(self.YT.shape)

Step Twenty - Six - np.copyto(X, self.X)

Step Twenty - Seven - np.copyto(Y, self.Y)

Step Twenty - Eight- np.copyto(XT, self.XT)

Step Twenty Nine - np.copyto(YT, self.YT)

Step Thirty - **for i in range(9):**

step Thirty-One- Train the dataset.

Step Thirty- Two - First, load the data from the pickle file

Step Thirty three- import the previous scripts

Step Thirty-four - import LoadData

Step Thirty five - import DataPreparation

Step Thirty six - import pickle

Step Thirty seven - file = open('flowdata.pickle', 'rb')

Step Thirty Eight - data = pickle.load(file, encoding = 'latin1')

Step Thirty Nine - import pickle

Step fourty - print(pickle.__doc__)

Step fourty one - Create portable serialized representations of Python objects.

Step fourty-one -apply machine learning algorithms

Step fourty two - Import the required modules to use four machine learning algorithms from sklearn:

Step fourty - three - from sklearn.linear_model import

Step fourty four- from sklearn.tree import

Step fourty five- from sklearn.naive_bayes import

Step fourty six - from sklearn.neighbors import

Step fourty seven - Prepare the data by using the previous module build.

Step fourty Eight- import DataPreparation by typing import

DataPreparation:

Step fourty Nine - DataPreparation. Prepare (Xdata, Ydata, XdataT, YdataT)

Step Fifty- train the models=

Step Fifty one - we now train the model with different techniques so that we can later select the most suitable machine learning technique for our project.

Step Fifty-two - prepare the data and select the features

Step Fifty Three- define the machine learning algorithm

Step Fifty Four - fit the model,

Step fifty five - print out the score after defining its variable.

Step fifty six - Decision tree model:

Step fifty seven - clf = DecisionTreeClassifier()

Step fifty eight - clf.fit(Xdata, Ydata)

Step fifty nine - paPrediction = clf.predict(XdataT)

Step sixty - Score = clf.score(XdataT, YdataT)

Step sixty - one - print ("The Score of the Decision Tree Classifier is", Score * 100)

Step sixty one- Neural network model:
Step sixty two - from keras.models import *
Step sixty three - from keras.layers import Dense, Activation
Step sixty four - from keras.optimizers import *
End

Algorithm for Classification

Begin
Step 1: import numpy
Step 2: import pandas library
Step 3: import tensorflow
Step 4: read in botnet data .csv
Step 5: is data accurate?
if data in the appropriate format, continue to (6)
else
 go back to (4)
step 6: assign labels to the .csv file
step 7: explore data by checking columns and rows

Perform Data Preparation
Step 8: import operating system
Step 9: import threading
Step 10: create a class called preparation
Step 11: select training data
Step 12: select testing data
Step 13: build the data loader
Step 14: prepare the machine learning algorithm (decision tree algorithm)
Step 15: import loadData
Step 16: import DataPreparation
Step 17: import pickle
Step 18: read in pickle data .pickle
Step 19: Functions:
 dump(object, file)
 dumps(object) -> string
 load(file) -> object
 loads(string) -> object
step 20: call the machine learning classifiers

step 21: import sklearn
from sklearn.linear_model import
from sklearn.tree import
from sklearn.naive_bayes import
from sklearn.neighbors import
step 22: train the model
step 23: prepare data and select its features
step 24: define a machine learning algorithm
step 25: fit the model
step 25: show results
end

4. CONCLUSIONS

This paper presented some of the relevant tools needed for a botnet detection and management system. Some related literatures were reviewed. Some tools were presented for each phases of the system implementation and design. These phases include the packet capture phase, the extraction of flow characteristics phase and the classification Phase. Some of the tools presented include; Numpy, Scikit-learn, Tensorflow, Keras, Theano, Pandas, Matplotlib, Seaborn and Pickel. The database management tool presented include, Wireshark and Anaconda. This paper presented a brief step and guide on how some of the tools were applied. This paper is very significant in creating Botnet Detection Systems as it gives a guide on the tools to use.

REFERENCES

- [1]. *Analysis of Botnet Classification and Detection Technique: A Review*. Iduh, Blessing Nwamaka and Okonkwo, Obikwelu Raphael. 10, October 2018, Journal of Emerging Technologies and Innovative Research, Vol. 5, pp. 195-201.
- [2]. *HTTP BASED BOT-NET DETECTION TECHNIQUE USING APRIORI ALGORITHM WITH ACTUAL TIME DURATION*. Khillari, Ashwini and Augustine, Archana. 2017, International Journal of Computer Engineering and Applications, Volume XI, Issue III, March 17, www.ijcea.com ISSN 2321-3469, pp. 13-18.
- [3]. *A Survey of HTTP Botnet Detection*. Chaware, Saurabh P. and Bhingarkar, Sukhada. 2016, International Research Journal of Engineering and Technology (IRJET), pp. 713-714.
- [4]. *Peer-to-Peer Botnets: Overview and Case Study*. Grizzard, Julian B., et al. 2012, Computer Based Learning Unit, University of Leeds., pp. 1-6.
- [5]. *General framework for detection of botnet using Random forest in real time*. Selvam, Nandhini, Vanitha and Sumathi. 2015, International Journal of Scientific & Engineering Research, Volume 6, Issue 4, pp. 469-479.
- [6] *Botnet Detection and Countermeasures- A Survey*. Tesfahun, Abebe and Bhaskari, D.Lalitha. 2013, International Journal of Emerging Trends & Technology in Computer Science, pp. 309-314.
- [7]. *Botnet Detection based on Anomaly and Community Detection*. Jing Wang, Ioannis Ch. Paschalidis,. 2016, IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 523-531.
- [8]. *Holistic Botnet Detection Framework Independent of Botnet Protocols and Architecture*. Rostami, M. R., Idris, N. B., & Ismail, Z. A. 2017.
- [9]. *BOTNET DETECTION BASED ON COARSE GRAINED PEER-TO-PEER TECHNIQUE*. M.Muthulakshmi, M.Grace Ananthi, S.Suganya & P.Raghavan. 2017, International Research Journal Of Engineering Sciences (IRJES), pp. 86-89.

- [10]. **Chumachenko, Kateryna.** *MACHINE LEARNING METHODS FOR MALWARE DETECTION AND CLASSIFICATION*. India : XAMK University of Applied Science, 2017.
- [11]. *A Survey on Data Mining Methods for Malware Detection* . **Kuber, Ms. Shital Balkrishna.** 2014, International Journal of Engineering Research and General Science Volume 2, Issue 6, pp. 672-675.
- [12]. **Pedersen, Matija Stevanovic and Jens Myrup.** *On the Use of Machine Learning for Identifying Botnet NetworkTraffic*. Department of Electronic Systems, Wireless Communication Networks Section, y5430. Aalborg Denmark : s.n., 2016.

IJSER