Botnet Detection Technique Using Artificial Neural Network

Suleiman Muhammad Nasir, Olalere Morufu Olajide.
Ojeniyi Joseph. A.
Cyber Security Science Department
Federal University of Technology
Minna, Nigeria
suleiman.pg611993@st.futminna.edu.ng,
lerejide@futminna.edu, ojeniyija@futminna.edu.ng

Mebawondu Jacob Olorunshogo
Computer Science Department
Federal Polytechnic
Nasarawa, Nigeria

ABSTRACT - As the cyber space grows so also are it challenges. The most severe security challenges bothering cyber security researchers around the globe is botnet: a network of systems that is taken over by a hacker to launch attack or perform an unwholesome activity. Botnet as a means of cyber-attack delivery has more far reaching effect that any other means. It is a prime factor in the delivery of Distributed Denial of Service (DDoS), SPARM (mail or fake review), and click fraud among other cyber related crimes. It is equally implicated in most of the activities that are termed cyber warfare. The paper reviews the incidence and purulence of botnet and proposes an Artificial Neural Network based Botnet Detection and classification system. The model was implemented on an ISCXbotnet dataset and an accuracy of 98% was achieved in detection and classification.

KEYWORDS: Botnet, command & Control channel, botmaster/botherder, zombie,

INTRODUCTION

Botnet is coined from the words roBOT and NETwork. It describes a kind of attack where a set of connected computer systems (possibly over the internet) is taken over by a distance or remote system to launch a systematic and simultaneous attack on a target system. [1, 2,3]. The controlled systems are referred to as the bot while the controlling system is called the botmaster. The bots are mostly the end user host systems which are a priori infested or compromised while the botmaster can be a human user at a remote end or a system operating the various hosts[4, 5]. The malware system deployed to enable a remote control of a system is also be called a bot.

Botnet employs vulnerable machines using techniques that are employed by other types of malware (remote software vulnerability exploitation, social engineering etc). The defining chrematistics of a botnet is that is that it uses *Command and Control Channels*. This channel enhances the dissemination of botmaster's commands to the various bots or armies[6, 7].

Botnets happens to be the most severe security challenges bothering cyber security researchers around the globe [8]. The botnet can be used in implementing so many hazardous effects in an organization. It can be used to transfer secret information to a competitor or an adversary, to spread denial of Service attack (DoS) for phishing activities, key logging activities forward spam activities and click fraud etc. Unlike other classes of malware that are generally used to exhibit technical prowess botnets are mostly employed for illegal and illicit undertakings. These undertakings may take the form of extortion, identity theft and software piracy. Hence the reason for a concerted research in this area.

Recently many of the cyber-attacks have employed botnet in one way or the other [6, 9]. It signifies a vivid and cogent treat to the information systems across the globe.[10]. Botnet has the ability to obfuscate the identity of the attacker and easily multiply the source of attack. It has evolved from a simple spam factory to and elaborate big masquerade behind mammoth criminal activities. According to [11] the number of computers enrolled as bot increases by 8% per seek. Extracting from this every computer system is a potential victim. Another study has reported a botnet of up to 400,000 bots.

A botnet can be defined as a coordinated group of compromised machines controlled via Command & Control (C&C) communication channels that are connected to some C&C servers/peers managed by botmaster/botherder. In other words, it is an army of infested end-user systems under the influence of a remote instructor, which has the capability to unleash havoc or spread malicious software or conduct a malicious activity on a target system [12, 13, 1]. We can also define a botnet as an *assemblage* of *malware* instances that are *controlled* through a C&C communication channels

Through the botnet attack the identity of the botmaster is effectively concealed while the master can effectively conduct a multiple and simultaneous attack on the target. The bots can generally be used to conduct any attack as the botmaster deems fit. The essential characteristics of a botnet are that the bots communicate with some C&C servers/peers, enact some activities that can be said to be malicious, and these activities are in a similar or correlated the C&C

In the botnet world according to [14] there are three actors: the botmaster, the vulnerable systems and the defenders. The botmaster is actively involved in recruiting the bots and controlling them to perform his nefarious act. His motive is always economic gain. The vulnerable systems are the end users' systems whose vulnerability could be employed by the botmaster to get them recruited as zombies into the botnet. These system, if unpatched, remain tools in

the hands of a botmaster in conducting his criminal activities. The defender intends to prevent, detect, track host and trace the botmaster. He is always on the side of the law and protected by the law enforcement agency.

The botnet life circle divided into four phases: formation, C&C, attack and post attack phases [9]. At the formation phase a botmaster attempts to and compromise any available vulnerable machines. It uses all the available methods including social Engineering to recruit vulnerable hosts to become part of his network of zombies. At the C&C phase, the master communicates with the bot through the C&C mechanism. The botmaster always keep track of the bots by relaying regular instruction which may include when and how to attack. At this point a defender may intervene and disrupt the botmaster activity thereby binging the botnet system down.

The attack phase starts when the botmaster issues instruction to the bots to launch attack. The victim may be the host system itself or another system entirely depending on the mode of attack in the intention of the botmaster. During the post attack phase the bots may get exposed to the defender and it is brought down and the system patched. However, if the botnet remains active the botmaster increases the number of bots in his botnet by requiring more vulnerable systems for his future attack.

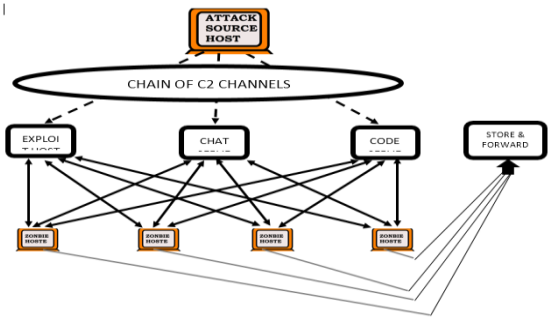

Figure 1 Chat based botnet architecture using a C2 char service

A. Architectures

There are two basic topologies available in thorough which a botnet can be deployed. These are Centralized Command and Control (C&C) architecture or distributed Command and Control (C&C) architecture [7]. Centralized C&C architecture is a simple and widely used architecture. In (C&C) architecture the botmaster simply communicates with the bots using the C&C server to send commands to the botnet. The architecture is one way and therefore suffers a single point failure. Once the C&C server is brought down the whole botnet is down. The famous approaches used in centralized botnet include Internet Relay Chat (IRC) and Hypothec Transfer Protocol (HTTP).[15]

Decentralized C&C is a relatively new architecture aimed at elevating the single point of entry setback suffers by a Centralized C&C architecture. In Decentralized C&C infrastructure all peers in the network works like a bot and a C&C Server at the same time. Here the botmaster's role is only to send command to any recruited peer. The life cycle of a distributed botnet (particularly the P2P architecture) consists of initial infection, peer propagation, secondary injection and attack[16; 15, 8 5]

B. History

The first observable botnet-like behavior was in 1988 when Robet Marris, released the internet first worm which was designed to "phone home" to a command & Control Server. In 1999 Sub7 – a Trojan and Pretty Perk – a worm were released. These malwares are known to connect infected systems to an IRC channel to listen to malicious commands. In 2000 was the emergence of the Global Threat bot GTbot which was known to to run custom scripts in response to IRC event. Agabot emerged in 2002 and in 2004 its variant phanbot was released. Phanbot Rbot was among the first known P2P based botnet.

2003 witnessed the rise of spybots with new features like key logging and data miming and Rbot (a family of bots which used compression and encryption algorithms to evade detection). Others in the series include Zeus (2006) Grum (2008) and Gameover Zeus (2011)

C. Artificial Neural Network

Artificial neural network is an information processing approach that is inspired by the way biological neurons process information. It is a computing system made up of simple highly connected elements (the neurons), which process information by their dynamic state response to external input working together to solve a particular problem [34, 35, 37]. As human being learn by examples, so is the Artificial Neural Network does. ANN can be configured for applications such as pattern recognition, data classification etc. through a learning process. ANN has a remarkable ability to derive meanings from complicated or impressive data. Hence it can be used to extract patterns and detect trends that are too complex to be noticed by either man or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. The expert can then be used to provide trajectory given a new situation of interest [35, 37]

As human being learns by examples, repeating the examples to create relationships so is the Artificial Neural Network does. The process of learning is referred to as training. The training problem is modeled as a minimization of loss function. This function is in general composed of and error term and regularization term. The error term evaluates how neural network fits the data set while the regulator tern

is sued to prevent out fitting by controlling the effective complexity of the network [37]

The loss function depends on the adaptive parameters (biases and synaptic) and can be conveniently grouped together into a single n-dimensional weigh factor w. The fundamental element of ANN is the neuron. Each neuron handles:

i. The multiplication of the network inputs, x1, x2, x3, . . .xn (from original data, or from the output of other neurons in a neural network) by the associated input weights,

ii. The summation of the weight and input product to the bias value associated with the neuron, and

iii. The passage of the summation result, u, through a linear or nonlinear transformation called the activation function, φ. The neuron's output, y, is the result of the action of the activation function.

$$\varphi = f(u) \quad (1)$$

$$y = \varphi(\sum_{i=1}^{n} \quad X_i W_i + b) \quad (2)$$

$$y = \varphi(W^T X + b) \quad (3)$$

where b is the bias value (or external threshold), W_i, is the weight of the respective inputs xi, u is the argument of the activation function and W^T is a transpose of the weight vector. The weight and bias are adjustable parameters of the neuron that causes the network to exhibit some desired or interesting behaviours. Fig 2 shows an illustration of an artificial neuron. [37]

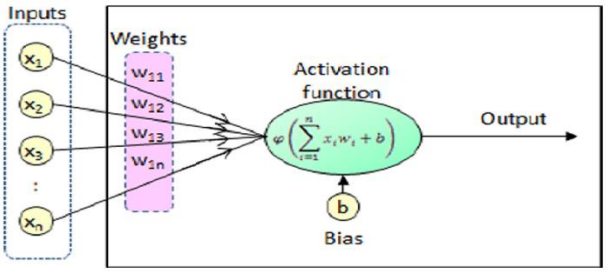

Figure 2 ANN Network [35, 37]

D. Training algorithms

There are five popular training algorithms. These are:

i. Gradient Decent (Steepest descent)
ii. Newton's Method
iii. Conjugate Gradient
iv. Quasi Newton
v. Levenberg Marquidt [35, 37]

II RELATED WORKS

The authors of [23] proposed a BotSniffer: an anomaly base botnet Command and Control detection system, although it is mainly sued for detecting centralized command and control activities. A BotMiner described and proposed by [22] uses a horizontal correlation approach that examines correlation between multiple host. BotMiner is independent of the botnet C&C protocol, structure and infection mode. The frame work of BotMiner targeted both centralized IRC and P2P botnets. The design of a BotMiner woks on the assumption that bots are coordinated malware that exhibit similar communication pattern and similar behavior both malicious and otherwise. Its major drawback is that it targets a group of compromised machines within a monitored network. However, it may fail if only one or two hosts are infected in a network but the hosts belong to a larger group of botnet with spans the monitored network. Again, it may fail to detect bots that exchange covert C&C messages without undertaking any malicious activity.

Another approach was introduced by the authors of [24]. This was based on the analysis of network behavior. The approach focused on the characteristics of IRC flows. Hence the approach is divided in to four stages. The first sage filtered off the not likely varying bot C&C data (given some prior knowledge of IRC bot commands) the second stag clustered the remaining flows into predefined network application clusters using machine learning. The third stage, the correlator stage, where clustered chat-like applications are clustered again in to a group of flows showing similar characteristic. Then the correlated flows are passed in to topology analysis to determine flows with common controllers. The last stage, an offline stage where a human factor analyzed the flows to determine whether they are botnet or not. This last stage spelled the setback of the proposed system.

The authors of [33] in their work identified some TCP related features for the detection of HTTP botnets. With these features a Multi-Layer Feed Forward Neural Network training model using Bold Driver Back-propagation learning algorithm was created. This algorithm had the advantage of dynamically changing the learning rate parameter during weight updating process. With this approach, Spyeye and Zeus botnets are efficiently identified.

In [25] the authors' work is focused on detecting botnet C&C commination on an end host. This is based on the fact that a recruited host need to keep in touch with the botmaster to remain relevant. This is always done by frequent communication between the bot and the botmaster such that such communication exhibits some temporal regularity over a period of irregular large time period. Evaluation of this approach using a real network traces yields low false positive rate.

The authors of [26] proposed a 2P2 botnet detaching technique using multi-phased flow model. In its proposal, 2P2 botnets are identified by observing similar flow occurring between a group of hosts in the network on a regular basis. Flow with similar behavior are groped and a transition model of the grouped flow is constructed using a probability matrix. Then a likelihood ratio is computed and used for bot detection. An accuracy of above 95 was achieved with experimental evaluation

In [27] the authors proposed a P2P botnet detection model using data mining technique. They explained the possibility of detaching host infected with a bot by analyzing bot traffic behavior. They introduced a set of metrics that could be used to differentiate between bot traffic, normal traffic, gaming traffic and general internet traffic. The evaluation of the proposed approach involved studying one 2P2 botnet virus known as Trojanpeacomm and achieved a detection rate of over 87%. However, the approach has not been applied on other categories of botnet to see how it performs. However according to [9] the approach had not been applied on other categories of botnet to see how it performs.

Since the botnet technology is evolving there is the need to consider distributed approach to bot dissemination. Hence the authors of [28] considered other types of network and streaming technologies and proposed artificial neural network for botnet detection.

The authors of [9] proposed a new approach to characterize and detect network traffic (Particularly the P2P bots) before it launches an attack. Using machine learning technique, they extracted and analyzed a set of C&C traffic behavior and its characteristics. It applied all the five machine learning techniques and compared them. The work in [29] observed that focusing on statistics network flow rather than packet content was unable to differentiate between IRC traffic and benign graphic. He pointed the setbacks on previous methods as Principle component analysis, (PCA), Correlation Feature Selection (CFS), minimum redundancy minimum relevance (mRmR) and improper evaluation of feature sets on test bed datasets. He built a dataset which incorporates different varieties of botnet of different protocols in realistic environment.

In their work [3], the authors modeled classifier using an assemble algorithm for botnet classification. It employed K-nearest Neighbor and decision tree and bagging and Ada Boost for soft voting.

III. METHODOLOGY

A. Proposed ANN Model for Based Botnet Detection

The proposed MLP network for botnet detection consists of 11 nodes at the input layer, one hidden layer and 4 nodes at the output layer. In the proposed model, 3 most frequently used activation functions have been considered. These are:

i. Logistic sigmoid activation function also known as logsig.

$$f(u) = \frac{1}{1+e^{-u}} \quad (4)$$

ii. Hyperbolic tangent sigmoid activation function also known as tansig.

$$f(u) = \frac{2}{1+e^{-2u}} - 1 \quad (5)$$

iii. Linear activation function also known as purelin.

$$f(u) = u \quad (6)$$

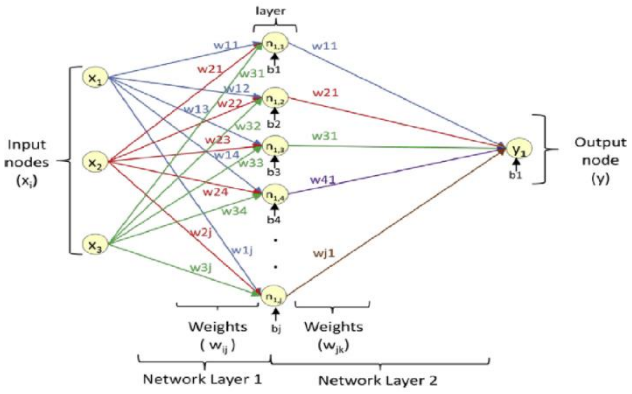

Figure 3

A schematic of the proposed ANN network with variable neurons in the hidden layer is shown inin fig 4 .where x_i ($1 \le i \le 11$) are the set of inputs; w_{ij} and w_{jk} are adjustable weight values: w_{ij} connects the ith input to the jth neuron in the hidden layer, w_{jk} connects the jth output in the hidden layer to the kth node in the output layer; yk ($1 \le k \le 5$) are the output.

Each neuron and output node has associated adjustable bias values: bj (where j = number of neurons) is associated with the jth neuron in network layer 1, bk ($1 \le k \le 5$) is associated with the node in the network layer 2. Within each network layer are: the weights, w, the multiplication and summing operations, the bias, b, and the activation function, u.

B. Data Acquisition and Pre-Processing

For the purpose of the work the use iscxbot2014 dataset. The dataset was originally created by [29]. It is made up of data set from three different sources. These sources are:

i. ISOT data set created by fusing different available data sets [30]. (French chapter of honeynet project, Eriksson Research in Hungary and Lawrence Berkeley National Library).
ii. ISCX 2012 IDS dataset generated in a physical testbed implementation using real devices that generated real traffic that mimic user behavior [31]
iii. Botnet traffic generated by the malwares capture facility project. A research project with the purpose of generating and capturing botnet traces in long term. [32.]

The dataset was extracted using Wireshark: a foremost and widely used network analyzer to generate an excel version of the set. The data set contain Forty-two (42] attributed out of which 11 were selected for our purpose.

C. Model Development

MATLAB was used to write the script files for the developed botnet detection and classification model and performance analysis to determine the weight and bias values, number of neurons and activation function type to be used in the optimal model equation. The script files were written to compare the relative effect of number of hidden layer neurons and activation function type on the performance of a designed network. A feedforward network topology and the default Matlab Neural Network Toolbox learning algorithm, Levenberg– Marquardt, were used.

The number of neurons in the hidden layer was varied from 5 to 33 in incremental steps of 2. Logsig, purelin and tansig type of activation functions were used to create 9 different pairs of activation functions. Thus, each of the 15 different numbers of neurons was used with 9 different pairs of activation functions. Each run of the script file generates 135 networks. For networks in which activation function pairs with logsig or tansig functions were used in the output layer, the input and target output data were pre-processed into 0–1 or _1 to +1 range using bellow respectively.

$$X_{norm\_0\_1} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (7)$$

$$X_{norm\_-1\_1} = 2 * \frac{X - X_{min}}{X_{max} - X_{min}} - 1 \quad (8)$$

The network outputs from the simulation process were then post processed to the original range. To compare the relative effect of number of runs on network performance, the script file was run 20 times and 20 runs generated 2700 trained networks for performance evaluation. The flow diagram of the ANN script file is shown in Fig. 3 flowchart.

Out of the over 1million iscxbot2014 dataset downloaded and extracted through Wireshark into excel file, 9040 samples were used while training the network and 4040 were used for testing. During the training process, the input and target output data were applied to the network and the network computed its output. The initial weight and bias values and their subsequent adjustments were done by the Matlab Neural Network Toolbox software. For each set of output in the output data, the error, e, (the difference between the target output, t, and the network's output, y,) was computed. The computed errors were used by the network performance function to optimize the network and the default network performance function for feedforward networks is mean squared error, MSE (the mean of the sum of the squared errors) which is given by:

$$MSE = \frac{1}{N}\left(\sum_{i=1}^{N} (e_i)^2\right) \quad (9)$$

$$MSE = \frac{1}{N}\left(\sum_{i=1}^{N} (t_i - y_i)^2\right) \quad (10)$$

where N is the number of sets in the output data. The weight and bias values are adjusted so as to minimize the mean squared error and thus increase the network performance. After the adjustments, the network undergoes a retraining process, the mean square error is recomputed and the weight and bias values are readjusted. The retraining continues until the training data achieves the desired mapping to obtain minimum mean square error value.

D) Developed Model

Mathematically, fig 4can be represented as:

$$y_i = \varphi_2\left(\sum_{j=1}^{m} w_{j1}\varphi_1\left[\sum_{i=1}^{11} w_{ij}x_i + b_j\right] + b_1\right) \quad (11)$$

where m is the total number of neurons in the hidden layer. The operations within an N layered network can be mathematically represented by;

$$y_i = \frac{\varphi_N\left(\sum_{j=1}^{p} w_{ki} \cdots \frac{\varphi_2\left(\sum_{j=1}^{m} w_{jk}\frac{\varphi_1\left(\sum_{i=1}^{n} w_{ij}x_i + b_j\right)}{Layer\ 1} + b_k\right)}{Layer\ 2} + \cdots + b_i\right)}{Layer\ N} \quad (12)$$

where i is the number for the ith neuron in layer N, p is the maximum number of neurons in layer N and N is the total number of network layers.

For linear activation function in both hidden and output layers and the use of m number of neurons in the hidden layer, eqn. (10)is transformed into:

$$y = LW[IW * X + b_I] + b_2 \quad (13)$$

$$y = [LW * IW] * X + [LW * b_1] + b_2 \quad (14)$$

Where

Layer weights, LW = [1,m] matrix

Input weights, IW = [m,11] matrix
Layer 1 bias, b1 = [m,1] matrix
Layer 2 bias, b2 = c
Input vector, X = [11,1] matrix

Thus

$$[LW * IW] = [\alpha\ \beta\ \gamma\ ] \quad (15)$$

$$[LW * IW] * X = [\alpha\ \beta\ \gamma\ ][TRD] \quad (16)$$

The proposed model equation is:

$$y = \alpha T + \beta R + \gamma D + c \quad (17)$$

IV RESULT AND DISCUSSION

In order to get best performance of the intelligent model, dynamic thresholding algorithm was developed to get the best threshold value using training dataset between 0.4 and 0.6 with an increment 0f 0.05. The best three threshold values ware recorded: 0.5, 0.505 and 0.58 as displayed in table 4.1

Table 4.1: Dynamic thresholding for training dataset

Thres-hold	Accuracy						Average Accuracy	Standard Deviation
0.485	99	96	99.14286	98.85714	95.85714	97.77143	97.77142857	1.686077443
0.49	99.85714286	99.2857143	99.71429	98.28571	98.42857	99.11429	99.11428571	0.724216677
0.495	98.85714286	99.2857143	99.57143	99.28571	98.71429	99.14286	99.14285714	0.349927106
0.5	99.71428571	99.7142857	99.42857	99.71429	99.42857	99.6	99.6	0.156492159
0.505	99.85714286	99.4285714	98.85714	99.71429	99.28571	99.42857	99.42857143	0.391230398
0.51	99	97.4285714	99.71429	99.57143	99.57143	99.05714	99.05714286	0.950832074
0.515	99.28571429	99.2857143	99.57143	98.57143	99.14286	99.17143	99.17142857	0.369776546
0.52	99.71428571	96.2857143	99.71429	99.71429	99.85714	99.05714	99.05714286	1.550510122

Table 4.2. Performance Evaluation of Activation Function Combinations and DTA

Acti-vation Functio n Pair	Accu-racy (%)	Correct Rate (%)	Error Rate (%)	Last Correct Rate (%)	Last Error Rate (%)	Inco n clusi ve Rate (%)	Classi fied Rate (%)	Sensi -tivity (%)	Speci -ficity (%)	False Positi ve Rate (%)	False Negati ve Rate (%)
PP	66.113	79.18	20.82	79.18	20.82	0	100	71.78	87.65	12.35	28.22
PT	66.113	79.51	20.49	79.51	20.49	0	100	71.37	88.84	11.16	28.63
PL	65.448 5	79.29	20.71	79.29	20.71	0	100	70.95	88.84	11.16	29.05
TP	99.6	99.89	0.11	99.89	0.11	0	100	99.79	99.52	0.48	0.21
TT	75.083 1	86.16	13.84	86.16	13.84	0	100	89.63	82.19	17.81	10.37
TL	66.113	79.51	20.49	79.51	20.49	0	100	71.37	88.84	11.16	28.63
LP	79.734 2	84.05	15.95	84.05	15.95	0	100	80.29	88.36	11.64	19.71
LT	82.059 8	84.83	15.17	84.83	15.17	0	100	82.16	87.89	12.11	17.84
LL	81.395 3	84.05	15.95	84.05	15.95	0	100	82.57	85.75	14.25	17.43

The performance evaluation of the activation function pairs and the Dynamic Thresholding Algorithm (DTA) ispresented in table 4.2 above. The metrics used in evaluating the performance are accuracy, correct rate, error rate, last correct rate, last error rate, inconclusive rate, classified rate, sensitivity, specificity, false positive rate and false negative rate. The performance evaluation report clearly indicates that all the traffic data were classified. There is no unclassified traffic. With reference to all the performance metrics, tansig-purelin (TP) activation function combination achieved the best values. The accuracy, sensitivity and specificity of TP pair are 99.6678%, 99.79% and 100%. With these values, the false positive rate of 0% and false negative rate of 0.21% were attained.

V CONCLUSION AND RECOMMENDATION

A. Conclusion

An Artificial neural network classifier for botnet detection and classification have been presented to for the. The model and its implantation on the chosen data set has shown to be of a comparative very efficiency given an average accuracy of 98%. The SCIX data set has also shown to be very relevant in the researches on botnet detection and classification.

B. Recommendation

We hereby recommend that more than one training algorithm could be combined to generate of more flexible and robust detection system.

REFERENCES

[1] M. Khosroshahy, M. K. M. Ali,& D. "SIC botnet lifecycle model: A step beyond traditional epidemiological models". *Computer Networks*, 2013, *57*(2), 404–421.

[2] E. Alparslan, A Karahoca, & D. Karahoca, "BotNet Detection : Enhancing Analysis by Using Data Mining Techniques", 2012.

[3] A. Bijalwan., N Chand, E. Shubhakar, & C. R. Krishna, "Botnet analysis using ensemble classifier". *Perspectives in Science*, 2016. PP *8*, 502–504.

[4] J. Zarfoss, "A Multifaceted Approach to Understanding the Botnet", 2006. PP 41–52.

[5] P.Torres, & C. Catania, "An Analysis of Recurrent Neural Networks for Botnet Detection Behavior". 2017.

[6] P. Wang, & C.C Zou, "An Advanced Hybrid Peer-to-Peer Botnet". 2010.

[7] M. Alauthaman, N. Aslam, L. Zhang, & R. Alasem, "A P2P Botnet detection scheme based on decision tree and adaptive multilayer neural networks" in. *Neural Computing and Applications*. 2016. https://doi.org/10.1007/s00521-016-2564-5

[8] P. Barthakur, B.T. Park, & C.V.Ramannagar, "A Framework for P2P Botnet Detection Using SVM", 2012. https://doi.org/10.1109/CyberC.2012.40

[9] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D, Zhao,W. Lu, & P. Hakimia. "Detecting P2P Botnets through Network Behavior Analysis and Machine Learning". 2011.

[10] S.S.C Silva, R.M.P. Silva, R.C.G. Pinto, & R.M. Salles, "Botnets : A survey", *57*, 378–403. 2013. https://doi.org/10.1016/j.comnet.2012.07.021

[11] A. Hardikar, & F. Bégin, BYOB: "Build Your Own Botnet Botnets". *SANS Institute InfoSec Reading Room*. 2011.

[12] D. Dagon, C. Zou & W. Lee. "Modeling botnet propagation using timezones". In *13th Annual Network and Distributed System Security Symposium NDSS'06,* 2006.

[13] F.Haddadi, J. Morgan, E.G. Filho & A.N. Zincir-heywood, "Botnet Behaviour Analysis using IP Flows", 2014. PP 1–6. https://doi.org/10.1109/WAINA.2014. 19

[14] J. Leonard, S. Xu & R. Sandhu, "A framework for understanding botnets". *Proceedings - International Conference on Availability, Reliability and Security, ARES*, 2009. PP 917–922. https://doi.org/10.1109/ARES.2009.65

[15] M. Bailey, E. Cooke, F. Jahanian, Y. Xu, A. Arbor & A. Arbor, "A Survey of Botnet Technology and Defenses" 2006.

[16] S. Anwar, & A. Karim, "A Statistics Approach towards Mobile Botnet Detection", 563–567. (2016).

[17] F. Freiling, Holz, T., & G.Wicherski, "Botnet tracking: Exploring a root-cause methodology to prevent denial of service attacks" In *European symposium on research in compute security*,2005.

[18] M. Reiter, & T. F Yen, "Traffic Aggregation for Malware Detection" 2007.1–22. https://doi.org/10.1007/978-3-540-70542-0_11

[19] A. Karasaridis, B. Rexroad, & D. Hoeflin:Wide-scale Botnet Detection and Characterization". 2007.

[20] J. Goebel, & T. Holz, & Rishi: "identify bot contaminated hosts by IRC nickname evaluation" *HotBots'07 Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets*, 2007.PP 8. https://doi.org/10.1.1.177.8170

[21] G. Gu, P. Porras, V. Yegneswaran, M. Fong & W. Lee, "Bothunter:Detecting malware infection through IDS-driven dialog correlation". In *16th USENIX Security Symposium*, 2007.

[22] G. Gu, R. Perdisci, J. Zhang, & W. Lee, « BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection" 2008.

[23] G. Gu, J. Zhang & W. Lee, "BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic", 2008.

[24] T. Strayer, & R. Walsh, "Botnet Detection", *36*(October), 2008 PP 0–29..https://doi.org/10.1007/978-0-387-68768-1

[25] F. Giroire, J.N. Chandrashekar, E. Taft, Schooler, & D. Papagiannaki, "Exploiting temporal persistence to detect covert botnet channels". *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009. *5758 LNCS*, 326–345, https://doi.org/10.1007/978-3-642-04342-0_17

[26] S.K Noh, J.H. Oh, J.S. Lee, B.N. Noh, & H.C Jeong, "Detecting P2P botnets using a multi-phased flow model". *Proceedings of the 3rd International Conference on Digital Society*, (2009).*ICDS 2009*, PP 247–253. https://doi.org/10.1109/ICDS.2009.37

[27] D. Liu, Y. Li, Y. Hu, & Z. Liang, "A P2P-botnet detection model and algorithms based on network streams

analysis", *International Conference on Future Information Technology and Management Engineering, FITME 2010*, PP *1*, 55–58. https://doi.org/10.1109/FITME.2010.5655788

[28] P. Salvador, C. Santiago, De & N. Siemens, "A Botnet Detection System based on Neural Networks", 2010. PP 57–62. https://doi.org/10.1109/ICDT.2010.19

[29] E.B. Beigi, H.H. Jazi, N. Stakhanova & A. A Ghorbani, "Towards Effective Feature Selection in Machine Learning-Based Botnet Detection Approaches", 2014. PP 247–255.

[30] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, & D. Garant, "Botnet detection based on traffic behavior analysis and flow intervals". *Computers & Security*, 2013. PP *39*, 2–16. https://doi.org/10.1016/j.cose.2013.04.007

[31] A. Shiravi, H. Shiravi, M. Tavallaee & A.A Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection". *Computers & Security*, 2011. *31*(3), 357–374. https://doi.org/10.1016/j.cose.2011.12.012

[32] S. Garcia. (n.d.). "Malware capture facility project, cvut university". Retrieved February 12, 2017, from https://mcfp.agents.fel.cvut.scz

[33] G. Kirubavathi Venkatesh & R. Anitha Nadarajan, "HTTP Botnet Detection Using Adaptive Learning Rate Multilayer Feed-Forward Neural Network". *Information Security Theory and Practice. Security, Privacy and Trust in Computing Systems and Ambient Intelligent Ecosystems*,2012. - *5*, *7322*, 38–48. https://doi.org/10.1007/978-3-642-30955-7_5

[34] J.O Eichie, O.D. Oyedum, M.O. Ajewole, & A.M. Aibinu, "Artificial Neural Network model for the determination of GSM Rxlevel from atmospheric parameters" in. *Engineering Science and Technology, an International Journal*, *20*(2), 795–804. 2017. https://doi.org/10.1016/j.jestch.2016.11.002

[35] I. Masood, N.Z. Abidin, N.R., Roshdi, N.A Rejab & M.F. Johari, "Design of an Artificial Neural Network Pattern Recognition Scheme Using Full Factorial Experiment". *Applied Mechanics and Materials*, 2014. *466*, 1149–1154. https://doi.org/doi: I0.4028/www.scient~c.net/AMM.465-466.1

[36] S. Stockt, & X. Van Der., "A Generic Neural Network Framework Using Design Patterns". *University of Pretoria, Pretoria*, 2017 (October).