I.J.Modern Education and Computer Science, **2013, 10, 9-18**
Published Online November 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2013.10.02

An Efficient Machine Learning Based Classification Scheme for Detecting Distributed Command & Control Traffic of P2P Botnets

Pijush Barthakur[1], Manoj Dahal[2], Mrinal Kanti Ghose[1]
[1]Department of Computer Science and Engineering, Sikkim Manipal Institute of Technology, Sikkim, India
[2]Novell IDC, Bagmane Tech Park, C V Ramannagar, Bangalore, India
pijush.barthakur@gmail.com, mdahal@novell.com, mkghose2000@yahoo.com

***Abstract*—** Biggest internet security threat is the rise of Botnets having modular and flexible structures. The combined power of thousands of remotely controlled computers increases the speed and severity of attacks. In this paper, we provide a comparative analysis of machine-learning based classification of botnet command & control(C&C) traffic for proactive detection of Peer-to-Peer (P2P) botnets. We combine some of selected botnet C&C traffic flow features with that of carefully selected botnet behavioral characteristic features for better classification using machine learning algorithms. Our simulation results show that our method is very effective having very good test accuracy and very little training time. We compare the performances of Decision Tree (C4.5), Bayesian Network and Linear Support Vector Machines using performance metrics like accuracy, sensitivity, positive predictive value(PPV) and F-Measure. We also provide a comparative analysis of our predictive models using AUC (area under ROC curve). Finally, we propose a rule induction algorithm from original C4.5 algorithm of Quinlan. Our proposed algorithm produces better accuracy than the original decision tree classifier.

***Index Terms*—** Botnet, Peer- to- Peer (P2P), WEKA, Linear support vector machine, J48, Bayesnet, ROC curve, AUC

I. Introduction

A Bot is a program that once gets installed in to a computer leads to establishment of some sort of command-and-control (C&C) channel with its C&C server and keeps listening to the C&C channel for future commands. C&C servers are the control centers from where commands from command-set of the botnet are issued to perform coordinated tasks and attacks. The attackers in internet have included worm like abilities to bot by including propagation component to spread through different infection vectors. This leads to creation of botnets that gives power to the bot-herder to access the infected computers and to carry-out different types of attacks like stealing of sensitive information, Distributed Denial of Service (DDoS) attacks, spam campaign, click frauds etc.

Botnets have evolved through time to imitate different topographical structures like IRC, HTTP, P2P etc. IRC and HTTP based botnets are centralized in nature and run the risk of single point of failure i.e. if C&C head is detected and taken down the botnet cripples. However, they have their relative merits as well. IRC based botnets being simple in setup and maintenance, continue to evolve. HTTP based botnets are difficult to detect as its C&C traffic can hide behind normal web traffic to evade detection mechanisms. P2P based botnets are the newest of its kind that mimic Peer-to-Peer (P2P) technologically and thus follow a distributed C&C structure. Absence of fixed C&C server in P2P botnets makes it very difficult to detect and dismantle such structures. Some of the known P2P bots are Storm [1], Nugache [2] and Waledac [3].

In this paper we propose a payload independent approach that can detect botnet even in case of encrypted C&C traffic. We use machine learning algorithms for classification and prediction of large volume of C&C traffic flows generated by P2P botnets. A network flow provides essential information in a network like who is talking to whom i.e. conversation between any two hosts in the network in any specific moment of time. The algorithms that are used to classify network flows during normal C&C operations of botnets are J48, BayesNet and Linear Support Vector Machines. Our approach can be used for proactive detection of P2P botnets by correlating similar bot flows of the same botnet. Our investigation is based on following assumptions: (i) P2P botnet establishes numerous small sessions, (ii) a P2P bot needs to keep communicating for having the malicious network alive, (iii) to avoid detection, a P2P bot passes minimal amount of information in each session, (iv) in every session between a pair of P2P bots, data flow happens in both the directions, (v) Every botnet has its own specific set of commands and C&C interactions of bots are preprogrammed to the set of commands they receive, and (vi) P2P botnets usually runs each session for a very small duration. Some of our assumptions are similar to the one used in paper [4][5].

Based on the above mentioned characteristic features of P2P botnets, we propose a rule generation algorithm for botnet traffic classification. We used an indirect method of deriving the initial rule set from decision tree

generated using C4.5 algorithm. Then, we followed a step-by-step approach for optimization of the rule set. Our final rule set has a uniform structure providing significant insight in to similarities within P2P botnet C&C traffic.

Rest of the paper is organized as follows: Section II provides a brief overview of related works. In Section III, we discuss the inherent differences between botnet C&C flows and normal network flows. In Section IV we provide the architectural overview of our classification process and the approach we have followed for dataset preparation. In Section V, we briefly describe the algorithms used in our classification task and also provide a detail analysis of the result of our classification models. In Section VI, we propose a rule generation algorithm for P2P botnet traffic classification and finally we conclude in Section VII.

II. Related works

A botnet operator needs to spend a sufficiently long interval of time with its newly acquired bot, before it is finally deployed for attack. This pre-attack period involves a sequence of stages like initial infection of the computer with bot code, the process of "rallying" i.e. the procedure adopted by a botnet for self identification of newly created bots so that it can initiate contact with Command & Control (C&C) server, and sequence of measures taken for the newly created bot client to make it secure. Measures taken to make a bot client secure normally involve deployment of anti-antivirus tools and Rootkit or similar tools in order to hide itself from applications already installed by security agencies. In a newly created bot client, the hacker also employs tools to retrieve details of the computer (e.g. processor speed, memory, network speed etc.) and to search for location of any leftover tools by an earlier infection [6]. It is imperative to study botnet behavior during these early phases of exploitation in order to neutralize a bot before it takes part in any attack. We may term such detection approaches as "proactive". However, most detection techniques developed till date are reactive in nature. In a recently published work [7], the author emphasizes on the use of network flows with incident handler for detection of network threats. It states that the flow data collected from all critical points throughout the network infrastructure provides needed visibility that can help to identify systems that are infected with malware or participating in a botnet. They can also reveal an attacker's targets, systems that are already compromised, and even the attackers themselves. A machine learning based approach for botnet detection was proposed in paper [8] albeit for IRC botnets. It uses classification using flow characteristics that can be employed for proactive detection of IRC bots. Zhao et. al. [9] have proposed a machine learning based classification scheme for detection of P2P botnets based on a set of network traffic attributes observed during a selected time window. They used Bayesian Network and Decision Tree classifiers with 12 selected traffic attributes. However, we achieved better accuracy with our selected set of attributes. Wernhuar et. al. [10] have proposed a mechanism to quickly identify P2P botnet traffic flows during the connection stage. They used Response to Intervention (RTI) method to observe the traffic flows of normal P2P applications and P2P botnets. Then they used decision tree model for classification, and information obtained were used for identification of abnormal traffic flows and the location of zombie computers. The detection technique proposed in our earlier work [11], also emphasizes on a similar approach for detection of P2P botnet. However, the approach is based solely on non-linear Support Vector Machines (SVM) and hence takes long time duration to complete the classification task.

Apart from machine learning based approaches, DNS based approaches may also be used for proactive detection of botnets. Botnet DNS traffic exhibits some unique properties like sudden and abnormal increase in DNS request rates mainly due to group activities of bots within a botnet, use of Dynamic DNS (DDNS) and in many cases use of fast-flux service network (FFSN) that results in rapidly changing DNS entries [12][13]. Moreover, most botnets today uses DNS to find C&C server. An approach for detection of algorithmically generated domain names of some recent botnets using DNS based "domain fluxing" for command-and-control has been proposed in the literature [14]. Botnets that rely on "domain fluxing", generate domain names algorithmically.

Some other detection approaches are worth mentioning. Masud et. al. [15] proposed a flow based approach to classify C&C and normal flow to learn temporal correlation between an incoming packet and one of the following logged events: (i) an outgoing packet (ii) a new outgoing connection and (iii) an application startup. Any incoming packet correlated with one of these logged events is considered a possible botnet command packet. Another approach called BotMiner[16] uses Data Mining for botnet detection through cross cluster correlation. BotMiner is based on essential properties of a botnet like bots within the same botnet exhibits similar communication pattern and similar malicious activity patterns. These detection approaches proposed in [15] and [16] are more of reactive in nature than being pro-active. Hossein et.al. [17] have proposed a P2P botnet detection model using a new bio-inspired model like Artificial Immune System (AIS). The AIS system provides a multilayered protection mechanism to discriminate between the malicious and safe activities. AIS based detection involves two main steps: First, to train the detector using a training data set that contain malicious patterns and system's normal activities. Second, is to monitor real system's traffic for malicious activities using detector sets. Finally, the malicious hosts are identified through identification of similar communication patterns and similar malicious activities. Hang et. al. [18] have proposed Entelecheia, an approach of P2P botnet detection using graph mining through exploitation of "social" behavior of the botnet during its

waiting stage. This they have done in two broad steps, first they created a graph through network-wide interactions of hosts and then they filtered and clustered hosts based on flow information. Shishir et. al. [19] proposed BotGrep that exploits spatial relationship among P2P botnet's communication traffic through graph analysis. This algorithm iteratively partitions the communication graph into a faster-mixing and a slower-mixing piece and then narrows it down on to the fast-mixing component as fast-mixing represents a property of the P2P botnet C&C graph. However, BotGrep has to be combined with some other malware detection scheme for effectively distinguishing botnet communication structure from other applications using P2P communications. Babak et. al. [20] have proposed PeerRush, a generic classification approach that can accurately detect different types of legitimate and malicious P2P traffic. An application profile is initially created by learning traffic samples of known P2P applications. The network traffics generated by P2P hosts within monitored network are then matched with the learned application profile for accurate detection and categorization of P2P applications. Li et. al. [21] proposed a P2P botnet detection framework by identifying similar patterns of P2P botnet flows such as outbound network degree, connection failure rate etc. that occurs at irregular phased intervals. It is called irregular phased similarity (IPS) and used it to determine flow clusters. Then a distance is derived between such flow clusters and compared it with a threshold value for the distance to determine the number of flow clusters that are closer. Finally the ratio of similar clusters is measured and compared it with a predefined threshold to identify a suspicious P2P bot. However, this technique is still in theory only and needs to be practically evaluated.

III. PROBLEM DESCRIPTION AND ASSUMPTIONS

The modern network traffic involves various data types, such as files, e-mails, Web contents, real-time audio/video data streams, etc. Each of these data types either use TCP or UDP as transport layer protocol depending on the type of transmission needed. For example, for transfer of files, e-mails, Web contents etc., the Transmission Control Protocol (TCP) appears to be suitable for its reliability. On the other hand, for transfer of real-time audio/video data streams, which is time-sensitive, the User Datagram Protocol (UDP) is typically used. Applications using TCP establishes full-duplex communication and also flow control i.e. while establishing connection, the ACK sent back to the sender by the receiving TCP, indicates to the sender the number of bytes it can receive beyond the last received TCP segment, without causing overrun and overflow in its internal buffers. Most P2P applications use UDP protocol for communication. Therefore, when we capture data from various applications in the internet, we find non uniformity in terms of volume, time etc. and in many cases are also unidirectional in nature.

Traffic flows captured from P2P bots (in our case Nugache) mostly uses TCP. A bot is a program and therefore every command issued by a bot in its normal C&C operations is followed by a response from either a server in its hierarchy in the botnet or from some other bot in its peer group. In other word, C&C interactions in P2P botnets must follow a strict command-response pattern. Also, a P2P bot needs to keep communicating to have their malicious network working. That is, a P2P bot needs to keep itself updated about other bots that are still active in its network. In normal C&C operations P2P bot establishes numerous small sessions. More specifically, they keep changing communicating ports for normal C&C interaction or until they lunch attack. Therefore, the number of packets in each of the bot generated flow during normal C&C operation is usually small. Finally we observe that the packets in bot generated flows are small in size. In our observation, we find all bot packets have size less than 500 bytes. Moreover, among the few packets transferred in a bot flow only one or two packets carry highest bytes, whereas, the normal P2P traffic carries most of the packets to the size of MTU. All these observations have led us to formulate our problem which is subsequently put to classification using machine learning algorithms.

IV. ARCHITECTURAL OVERVIEW AND DATA SET PREPARATION

Our classification scheme contains four broad modules namely, data acquisition, extraction, filtering & scaling and botnet C&C traffic classification. We show the pipeline diagram in Fig. 1.

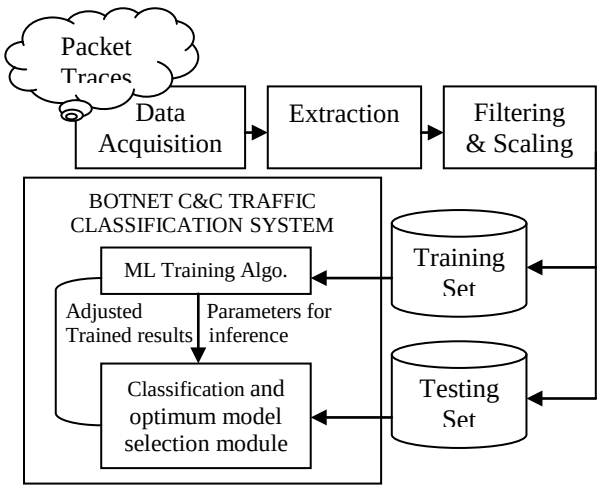

Figure 1: Pipeline diagram for botnet traffic classification

Description of the pipeline diagram:

i) *Data Acquisition:*Raw packets were collected using Wireshark[22] from different computers connected to our campus network. We acquired the botnet dataset of Nugache bot from Department of Computer Science, The University of Texas at Dallas. This is the same dataset which were used in the botnet related research works of [23].

ii) *Extraction:*Useful features for classification were extracted from packet headers. We extracted different kind of information from packet data like the size of largest packet transferred in a flow, the ratio of this packet in a given flow, the difference in time (calculated in seconds) for last packet received in either direction for responding flows, difference in number of packet being sent in either direction for responding flows and also several other features extracted from packet headers. For non-responding flows we include a unique number(we consider a very large number) to make it differentiable. We used 999 for difference in number of packets and 99999 for difference of time. However, more than 60% of normal traffic flows in our dataset has responding flows.

iii) *Filtering & Scaling:* We filtered our datasets through removal of unwanted flows so that it is optimized for classification. We removed the flows having a single packets , since single packets does not provide any statistically significant information. Similarly, flows representing NetBIOS services, broadcasting and DHCP were removed. We scaled the datasets to the range of 0 to 1. We created three separate files containing 18926, 18898 and 18826 instances. Each file contains flow captured for more than 10 hours for each bot as well as normal web traffic. Normal network flows comprised 17.83 % of total flows on an average of the three files.

iv) *Botnet C&C traffic classification system:* We feed our optimized datasets to our Botnet C&C traffic classification using 10 fold croos validation. The botnet C&C traffic classification system has two modules – one for training the system using input training sets and the other to evaluate the optimum model using testing set.

V. CLASSIFICATION AND ANALYSIS OF RESULTS

We used three machine learning classification algorithm for classification of P2P botnet control traffic. A brief description of the three algorithm is provided first and then we provide an analysis of results obtained from classification models.

a. *Decision Tree (J48):* A Decision Tree (C4.5 decision tree algorithm) [24] is one of the most popular classification algorithm that uses recursive partition of instance space based on concept of information entropy. Training is done on an already labeled set of instances having a fixed set of attributes and then splitting it by choosing an attribute giving maximum normalized information gain (difference in entropy). The algorithm then repeats this process recursively for each of the subparts. A Decision Tree classifier uses *pruning* tactics that results in reducing the size of the tree (or the number of nodes) to avoid unnecessary complexity, and to avoid over-fitting of the data set when classifying new data. The overlying principle of pruning is to compare the amount of error that a decision tree would suffer before and after each possible prune, and to then decide accordingly to maximally avoid error.

b. *BayesNet (using Genetic Search) [25]:* Given a set of variables U = {x1,x2,…,xn}, n≥ 1, a Bayesian Network B over the set of variable U is a network structure Bs, which is a directed acyclic graph(DAG) over U and a set of probability tables

$$B_p=\{p(u|pa(u))|u\in U\}$$

where pa(u) is the set of parents of u in B_s. The learning task consists of finding an appropriate Bayesian network given a dataset D over U. We use Bayes Network learning algorithm [26] that uses genetic search for finding a well scoring Bayes network structure. Genetic search works by having a population of Bayes network structures and allow them to mutate and apply cross over to get offspring. The best network structure found during the process is returned.

c. *Linear Support Vector Machine:* Linear SVMs are very powerful classification tools. The software packages that implements Linear SVM are SVMperf [27] Pegasos[28] and LIBLINEAR[29]. Given a set of instance-label pairs (xi, yi), i = 1,..,l, xi ∈ Rn, yi ∈{-1,+1 }, Linear SVM solve the following unconstrained optimization problem with loss function ξ(w;xi,yi):

$$\min_{w} \frac{1}{2} w^T w + C\sum_{i=1}^{l} \xi(w;x_i,y_i)$$

Where, C>0 is a penalty parameter. In Linear SVM, the two common loss functions are max (1- $y_i w^T x_i$, 0) and max (1- $y_i w^T x_i$, 0)^2. The former is referred to as L1-SVM and the later as L2-SVM.

Here we discuss our simulation results of classification using three machine learning algorithms namely J48 (Weka implementation of C4.5), Bayesian Network and Linear SVM. In Table 1, we provide the list of 10 flow and botnet characteristic features initially considered for classification. However, after thorough investigation we used only bottom four features in our final classification models. This is mainly because of inconsequential nature of first six features. We use WEKA [30] Data Mining environment for classification. Weka provides a collection of Machine Learning (ML) algorithms and several visualization tools for data analysis and predictive modeling.

The results show very high True Positive (TP) rate and very low False Positive (FP) rate for the best models we obtained. High true positive rate or Hits mean that the machine learning classifiers worked well in prediction of actual bot flows. Very low false positive rate or false alarm shows that very few normal web flows were confused as bot generated flows. We consider the following performance metrics to compare our classification models:

$$\text{Accuracy}=\frac{TP+TN}{TP+TN+PP+PN} \quad (1)$$

$$\text{Sensitivity}=\frac{TP}{TP+PN} \quad (2)$$

$$\text{Positive Predictive Value}\left(\text{PPV}\right)=\frac{TP}{TP+PP} \quad (3)$$

$$\text{F}-\text{Measure}=\frac{2*\text{TPrate}\,\text{Pr}ecision}{\text{TPrate}+\text{Pr}ecision} \quad (4)$$

Where TP = True Positives or Hits, TN = True Negatives or correct rejections, FP = False Positives or false alarms and FN = False Negatives or misses.

Here Sensitivity or Recall is the proportion of correctly identified bot flows. Similarly, PPV or Precision is the proportion of correctly identified bot flows out of total number of flows classified as bot by our classifier. F-Measure is a measure of a test's accuracy. The initial datasets prepared from three bots connected to the same botnet and normal web traffic samples, were passed through Randomize filter available with WEKA's unsupervised instance filter category. This was necessitated because our original datasets were imbalanced having less normal web flows. While

TABLE 1: Flow and botnet characteristics features

Flow name	Description
bytes_lrgst_pkt	Total bytes transferred with largest packets in a flow.
total_bytes	Total bytes transferred in a flow.
avg_iat	Average inter arrival time between packets in a flow.
var_iat	Variance of inter arrival time between packets in a flow.
avg_pktl	Average size of packets in a flow.
var_of_pktl	Variance of packet sizes in a flow.
lrgst_pkt	Size of the largest packet in a flow.
ratio_of_lrgst_pkt	Ratio of largest packets in a flow.
Rspt_diff	Time difference (calculated in seconds) between last packet received in either direction for responding flows.
Rsp_pkt_diff	Difference in number of packet being transferred in either direction for responding flows.

TABLE 2: Weighted average tp rate, fp rate and time taken

Algorithm	TP rate	FP rate	Time taken (seconds)
J48	0.997	0.009	0.85
BayesNet	0.996	0.008	10.53
LIBLINEAR	0.961	0.173	7.2

TABLE 3: Computed performance metrics.

Algorithm	Accuracy	Sensitivity	PPV	F-Measure
J48	0.996277	0.997	0.997	0.997
BayesNet	0.996399	0.996	0.996	0.996
LIBLINEAR	0.960953	0.961	0.962	0.959

constructing classifier, we used 10-fold cross validation so that there is no over-fitting of our training set. Table 2 shows weighted average TP rate and FP rate for best model obtained for each of the three classification algorithms along with time taken to build model on training data (or training time). Similarly, Table 3 shows results of performance metrics computed. In both Table 2 and Table 3, values are average of results obtained for the three data sets.

Next, we applied Synthetic Minority Oversampling Technique (SMOTE) [31] to increase the number of samples in the minority class. SMOTE generates synthetic samples by multiplying the differences between the feature vector (sample) under consideration and its nearest neighbor with a random number between 0 and 1. By applying SMOTE our sample count in the minority class increased from 17.83% to 30.27%. Results obtained from synthetically increased dataset are shown in Table 4 and Table 5.

Among the three algorithms used for classification of our datasets J48 and BayesNet shows very promising results in prediction of suspicious botnet flows. While Bayesian Network is slightly better in accuracy i.e. correct prediction of previously unknown data, J48 takes very little simulation time on training data. We use weka's default setting for J48 with C.4.5 pruning technique. Next we use J48graft algorithm to produce grafted C4.5 decision tree. Grafting [32] is an algorithm for adding nodes to the tree to increase the probability of rightly classifying instances that falls outside the area covered by the training data. However, we find a very marginal increase in accuracy (Accuracy: 0.996419 for the original datasets and 0.996675 for the synthetically increased datasets). But, it also leads to an enlarged tree making it more complex and also increase in training time (Time taken: 1.02 Seconds for the original datasets and 1.26 seconds for the synthetically increased datasets). We also evaluated the models generated using LIBLINEAR algorithm for exponentially growing sequences of C starting from its default value 1.0. We find the best performing model at $C = 2^8$. However, the average training time and the number of false negatives increased with each subsequent increase in C. For example, in case of C = 1.0, the average training time was 0.58 second and the average number of false negatives was only 17. For $C = 2^8$, the average training time increased to 7.2 seconds and the average number of false negatives stood at 29. Moreover, with increase in size of the datasets using SMOTE, there is a fall in performance of LIBLINEAR model.

In many real world problems where datasets may be highly imbalanced, the accuracy (the rate of correct classification) of a classifier may not be a good measure of performance because the accuracy measure does not consider the probability of the prediction: as long as the class with largest probability estimation is the same as the target, it is regarded as correct. That is, probability estimations or 'confidence' of the class prediction

produced by most classifiers is ignored in accuracy [31, 33]. In case of botnets, the number of botnet flows is bound to be large enough compared to normal web traffic flows when captured for same time duration. The area under ROC (Receiver Operating Characteristic)

TABLE 4: Weighted average tp rate, fp rate and time taken for the synthetically increased datasets.

Algorithm	TP rate	FP rate	Time taken (seconds)
J48	0.997	0.005	1.03
BayesNet	0.997	0.004	15.02
LIBLINEAR	0.942	0.129	8.66

TABLE 5: Computed performance metrics for the synthetically increased datasets.

Algorithm	Accuracy	Sensitivity	PPV	F-Measure
J48	0.9966	0.997	0.997	0.997
BayesNet	0.996855	0.997	0.997	0.997
LIBLINEAR	0.942322	0.942	0.946	0.941

curve (AUC) provides a alternative and better measure for machine learning algorithms. AUC is more sensitive to Analysis of Variance (ANOVA) tests and is independent to the decision threshold, as well as it is invariant to a *priori* class probability distributions[34].

The ROC curve compares the classifiers' performance across the entire range of class distributions and error costs. The ROC curve is given by TP rate and FP rate. ROC curve drawing algorithm use decision threshold values and construct the curve by sweeping it across from high to low. This gives rise to TP rate and FP rate at each threshold level which can intern be interpreted as points on the ROC curve. For more detail on ROC curve drawing algorithm one can refer to the work done by Hamel [35]. AUC provides a good measure of comparing the performances of ROC curves in particular to the cases where dominance of one curve is not fully established. More details can be found in Ling et. al. work [33]. In case of perfect predictions the AUC is 1 and if AUC is 0.5 the prediction is random.

The model performance through ROC curves for our classification models is shown in Fig. 2, Fig. 3 and Fig. 4. The X-axis represents False Positive Rate and Y-axis represents the True Positive Rate. For original randomized dataset, the average AUC value obtained are 0.995, 0.999 and 0.894 for J48, BayesNet and the LIBLINEAR classification models respectively. For the synthetically increased datasets, the corresponding average AUC values are 0.997, 1 and 0.907. Thus comparing the results in Table 2, Table 3, Table 4, Table 5 and AUC values, we can say BayesNet using Genetic search provides the best classifier. Nevertheless, J48 takes very less time in building the training model with a reasonably good model performance.

VI. A RULE INDUCTION ALGORITHM FOR BOTNET TRAFFIC CLASSIFICATION

From analysis of results obtained from three classifiers in Section VI, it is apparent that Decision Tree (J48) gives both high predictive accuracy and faster model building time. Therefore, we used the indirect method of building classification rules i.e. to extract rules from C4.5 classification model discussed in Section V. Rule induction from Quinlan's famous C4.5 algorithm [24] is the conjunction of antecedents to arrive at a consequence. That is, if A, B and C are the test nodes encountered in the path from root to leaf node D, then the rule generated would be in conjunctive form such as "if A and B and C then D". The approach for rule generation is as follows:

First we trained the C4.5 tree. Then from it we extracted initial set of rules by considering test conditions in each path as conjunctive rule antecedents and corresponding class labels as rule consequences. We extracted 21 such rules from the decision tree trained on our dataset. Then we remove those antecedents which can trivially be removed. For example, if there are two antecedents in the same rule, say t>x1 and t>x2 where t is the attribute and x1, x2 are the numeric attribute values such that x1>x2, then we accept the antecedent t>x1 and

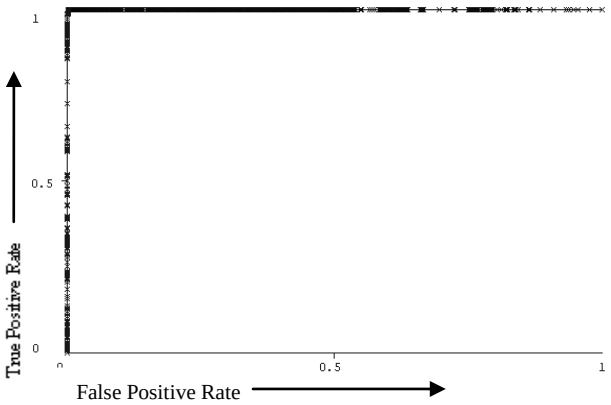

Figure 2: ROC curve for the BayesNet classification model (Class: P2Pbot)

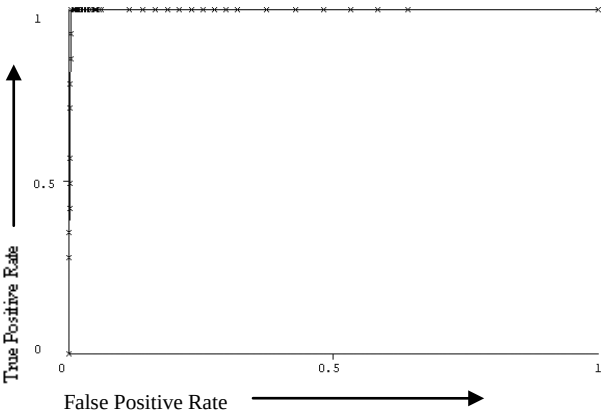

Figure 3: ROC curve for the J48 classification model. (Class: P2Pbot)

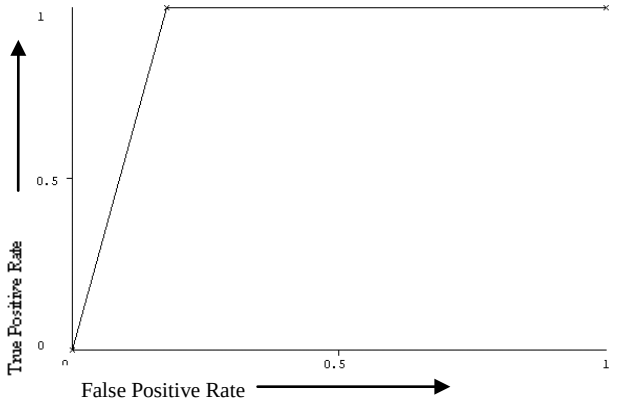

Figure 4: ROC curve for the LIBLINEAR classification model. (Class: P2Pbot)

discard the other. Similarly, if antecedents were t<x1 and t<x2, then we accept t<x2 provided x2<x1. This reduced rule lengths.

We computed Coverage and Accuracy for our initial set of C4.5 rules. Coverage is the fraction of records that satisfy antecedents of a rule, given by

$$\text{Coverage} = (|\text{LHS}|) / n \quad (5)$$

And, accuracy is fraction of records covered by the rule that belong to the class on the RHS. It is given by

$$\text{Accuracy} = (|\text{LHS} \cap \text{RHS}|) / (|\text{LHS}|) \quad (6)$$

Where n is the number of records in our dataset, |LHS| is the number of records that satisfies antecedent of a rule and |LHS ∩ RHS| is the number of records that satisfies the rule as a whole.

In our trained C4.5 tree, the attribute 'response packet difference' is the root followed by number of splitting on attributes 'ratio of largest packet' and 'largest packet'. Only in one case 'response packet difference' and 'response time difference' is used for splitting dipper in the tree. The initial set of rule is generalized by removing antecedents not contributing to accuracy and coverage of its original rule. To do this, antecedents corresponding to test nodes higher up in the tree were removed first (initially the root node). Then the Coverage and accuracy values for remaining part of the rules containing antecedents corresponding to test nodes dipper down the tree were calculated. If the freshly calculated coverage and accuracy values were not worse than the original, we replaced the original rule with its new variant. This process was repeated until further generalization of the rules was not possible. Rules are then grouped according to their predicted classes and subjected to further polishing using Minimum Description Length (MDL) principle [36] so that rules that do not contribute to the accuracy of our rule based classifier are removed.

In our newly generated rule set, we are left with ten rules that predicted normal traffic and four rules for botnet traffic, down from fourteen and seven respectively for our initial rule set. One important observation of our newly generated optimized rule set is that there are seven rules out of fourteen which have flows classified based on "proportions of large packets transferred in a flow" and "packets carrying maximum payload" only. This is shown in Table 6. This led us to believe that some more rules of our new rule set can be modified to fall within mutually exclusive ranges of these two attribute values without / insignificant degradation of their corresponding coverage and accuracy values.

We applied heuristic method to create a variant of some of the existing rules of the remaining rule set. The procedure adopted to modify remaining part of the rule set is as follows: We created a list of test conditions that belongs to remaining rules in the rule-set. Then we weighted each test condition according to summation of coverage values of their participated rules. We grouped them according to attribute name and arranged it in decreasing order of their weight-age values in each group separately. Then we considered one rule at a time from the remaining pool and used heuristic to replace one of its antecedents For example, in case of the following rule that predicts Normal flow,

"If (Response packet difference ≤ 0.003) And (Ratio of largest packet ≤ 0.504274) And (Largest packet ≤ 0.0115) And (Largest packet > 0.0063) Then Class = Normal" we replaced the antecedent (Response packet difference ≤ 0.003) with (Ratio of largest packet > 0.142857). The new rule generated with this replacement has an accuracy and coverage of 100% and 1.68 % respectively. The corresponding figures for the rule before replacement were 99% and 1.15%. Table 7 shows new variant of four such rules. However, in three rules we need to retain antecedents on other two attributes for correct prediction. Those three rules are shown in Table 8.

Finding the best decision tree is NP-hard and all current decision tree algorithms are heuristic algorithms. Therefore, the decision tree structures would be different for different training sets. However, using our approach most of the rules can be converted in to ranges of packet carrying maximum payload and its proportions in a flow.

We have generated the rules from decision tree created on one data set and tested it on all the three data sets. We found that rules in Table 6 has a coverage of approximately 24%, rules in Table 7 has a coverage of approximately 76% and rules in Table 8 has a very negligible coverage. We also found that the rule based approach has produced better accuracy (Average Accuracy = 99.63 %) than the original decision trees in all the three test cases.

VII. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a methodology for detecting P2P botnets using Machine Learning techniques. We used

TABLE 6: Rules based on "ratio of largest packet" and "largest packet" only

Rule antecedents		Rule Consequence	Accuracy (%)
Ratio of Largest Packets	Largest Packet		
> 0.504274	-	Normal	99.8
<= 0.142857	> 0.0457	Normal	99.3
<= 0.142857	<= 0.0083	Normal	100
<= 0.142857	> 0.0095 AND <= 0.0117	Normal	100
<= 0.142857	> 0.0118 AND <= 0.0365	Normal	97.7
> 0.00831 AND <= 0.142857	> 0.0365 AND <= 0.457	Normal	100
<= 0.142857	> 0.0117 AND <= 0.0118	P2Pbot	100

TABLE 7: Replaced antecedents in the new rules

Rule Antecedents			Rule Consequence	Accuracy (%) before replacement	Accuracy (%) after replacement
Replaced Antecedents	Replaced With	Unchanged Antecedents			
(Response Packet Difference ≤ 0.003)	(Ratio of Largest Packet > 0.142857), (Ratio of Largest Packet <= 0.504274)	(Largest Packet > 0.0118)	Normal	99.5	99.7
(Response Packet Difference ≤ 0.003)	(Ratio of Largest Packet > 0.142857)	(Ratio of Largest Packet <= 0.504274), (Largest Packet > 0.0063), (Largest Packet <= 0.0115)	Normal	99	100
(Response Packet Difference ≤ 0.003)	(Ratio of Largest Packet > 0.142857)	(Ratio of Largest Packet <= 0.504274), (Largest Packet <= 0.0063)	P2Pbot	99.8	99.8
(Response Packet Difference ≤ 0.003)	(Ratio of Largest Packet > 0.142857)	(Ratio of Largest Packet <= 0.504274), (Largest Packet > 0.0115), (Largest Packet <= 0.0118)	P2Pbot	100	99.9

TABLE8: Unchanged rules

Rule Antecedents	Rule Consequence	Accuracy (%)
(Response Packet Difference > 0.003), (Ratio of Largest Packet <= 0.00831), (Largest Packet > 0.0365), (Largest Packet <= 0.0457), (Response Time Difference <= 0.01261)	Normal	100
(Response Packet Difference <= 0.003), (Ratio of Largest Packet <= 0.142857), (Largest Packet > 0.0083), (Largest Packet <= 0.0095), (Response Time Difference <= 0.01261)	Normal	100
(Response Packet Difference > 0.003), (Ratio of Largest Packet <= 0.142857), (Largest Packet > 0.0083), (Largest Packet <= 0.0095)	P2Pbot	100

ML classification algorithms to classify P2P botnet Command & Control(C&C) traffic based on combined power of selected flow features and botnet behavioral characteristic features. From our experiments, we found that the BayesNet classifier created using Genetic search produces very promising result with an accuracy rate as high as 0.996399 and AUC value of 0.999. These are good indicators to justify efficacy of our classification techniques.

Other classification algorithms such as J48 and LIBLINEAR are very fast. For example J48 took only 0.85 second to build model on our training data having a reasonably good predictive accuracy. Therefore, a noteworthy contribution of this research work is that we proposed a machine learning based framework for quick

detection of P2P botnet traffic that has a high predictive accuracy. Our ROC curve analysis points to predictive accuracy of our classification models. However, we need to test our classification techniques in large-scale network set-ups.

Finally, we proposed a rule induction algorithm for P2P botnet traffic classification. We achieved better classification accuracy than decision tree classifier. We generated rules from traffic samples collected from Nugache botnet. However, same procedure can be adopted to generate rules for other P2P botnet traffic samples as well. Furthermore, our rule based approach can be a stepping stone for development of an unsupervised detection technique. Large number of botnet flows tends to exist within short intervals of proportions at which largest packets are transferred and also the size of the largest packet. Only when these two characteristic features failed to provide high predictive accuracy for a particular range of its values, the other two features were used.

ACKNOWLEDGMENT

We would like to thank Mohammad M. Masud, Department of Computer Science, University of Texas at Dallas for providing us the botnet dataset to carry out this research work.

REFERENCES

[1] E. Florio and M. Ciubotariu, *Peerbot: Catch me if you can*, Symantec Security Response,Tech. Rep., April 2007.

[2] S Stover, D Dittrich, J Hernandez, S Dietrich, "Analysis of the Storm and Nugache Trojans: P2P is here ", in USENIX December 2007, Volume 32, Number 6.

[3]G. Sinclair, C. Nunnery, B. Byung and H. Kang, "The Waledac Protocol: The How and Why" Proc. 4th International Conference on Malicious and Unwanted Software(MALWARE 09), IEEE Press, Feb. 2010.

[4] Wen-Hwa Liao, Chia-Ching Chang, "Peer to Peer Botnet Detection Using Data Mining Scheme", International Conference on Internet Technology and Applications, 2010.

[5] Guofei Gu, Vinod Yegneswaran, Phillip Porras, Jennifer Stoll, and Wenke Lee, "Active Botnet Probing to Identify Obscure Command and Control Channels" in Annual Computer Security Applications Conference,2009.

[6] Craig A. Schiller, Jim Binkley, David Harley, Gadi Evron, Tony Bradley, Carsten Willems, Michael Cross, "BOTNETS THE KILLER WEB APP", Syngress Publishing Inc.,2007.

[7] Kevin Gennuso Shedding Light on Security Incidents Using Network Flows, The SANS Institute 2012.

[8] Carl Livadas, Robert Walsh, David Lapsley, W. Timothy Strayer, "Using Machine Learning Techniques to Identify Botnet Traffic" in 2nd IEEE LCN Workshop on Network Security (WoNS'2006).

[9] David Zhao, Issa Traoré, Ali A. Ghorbani, Bassam Sayed, Sherif Saad, Wei Lu: Peer to Peer Botnet Detection Based on Flow Intervals. SEC 2012, pp. 87-102, 2012.

[10] Wernhuar Tarng, Li-Zhong Den, Kuo-Liang Ou, Mingteh Chen, "The Analysis and Identification of P2P Botnet's Traffic Flows", International Journal of Communication Network and Information Security(IJCNIS), Vo. 3, No. 2, August 2011.

[11] Pijush Barthakur, Manoj Dahal, Mrinal Kanti Ghose,"A Framework for P2P Botnet Detection using SVM", in the 4th International conference on Cyber-Enabled Distributed Computing and Knowledge Discovery(CyberC), 2012.

[12] H. Choi, H. Lee, H. Lee, and H. Kim, "Botnet Detection by Monitoring Group Activities in DNS Traffic," in Proc. 7th IEEE International Conference on Computer and Information Technology (CIT 2007), 2007, pp.715-720.

[13] Ricardo Villamarín-Salomón, José Carlos Brustoloni, "Identifying Botnets Using Anomaly Detection Techniques Applied to DNS Traffic", in IEEE CCNC proceedings,2008.

[14] Sandeep Yadav, Ashwath Kumar Krishna Reddy, A.L. Narasimha Reddy, Supranamaya Ranjan ,"Detecting Algorithmically Generated Domain-Flux Attacks with DNS Traffic Analysis.", 2012.

[15] Mohammad M. Masud, Tahseen Al-khateeb, Latifur Khan, Bhavani Thuraisingham, Kevin W. Hamlen. Flow Based Identification of Botnets Traffic by Mining Multiple Log Files. In Distributed Framework and Applications, 2008. DFmA 2008.

[16]Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection. In 17th USENIX Security Symposium,2008.

[17]Hossein Rouhani Zeidanloo, Farhoud Hosseinpour, Farhood Farid Etemad,"New Approach for Detection of IRC and P2P Botnets", International Journal of Computer and Electrical Engineering, Vol. 2, No. 6, December 2010.

[18]Huy Hang, Xuetao Wei, Michalis Faloutsos, Tina Eliassi-Rad, "Entelecheia: Detecting P2P P2P bots with Structured Graph Analysis", 19th USENIX conference Botnets in their Waiting Stage", IFIP Networking 2013.

[19] Shishir Nagaraja, Prateek Mittal, Chi-Yao Hong, Matthew Caesar, Nikita Borisov, "BotGrep: Finding on Security, 2010.

[20] Babak Rahbarinia, Roberto Perdisci, Andrea Lanzi, Kang Li, "PeerRush: Mining for Unwanted P2P Traffic", in proceedings of 10th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 2013), July, 2013.

[21] Huabo Li, Guyu Hu, Jian Yuan, Haiguang Lai. "P2P Botnet Detection based on Irregular Phased Similarity", in Second International Conference on

Instrumentation, Measurement, Computer, Communication and Control(IMCCC), 2012.
[22] http://www.wireshark.org/.
[23] M. M. Masud, J. Gao, L. Khan, J. Han and B.Thuraisingham," Mining Concept-Drifting Data Stream to Detect Peer to Peer Botnet Traffic", Univ. of Texas at Dallas, Tech. Report# UTDCS-05-08(2008).
[24] J. R. Quinlan, "C4.5: Programs for Machine Learning", San Mateo CA:Morgan Kaufman, 1993.
[25] Remco R. Bouckaert, "Bayesian Network Classifiers in Weka for Version 3-5-7 ", The University of Waikato, 2008.
[26]http://www.androidadb.com/source/weka-3-7-4/weka-src/src/main/java/weka/classifiers/bayes/net/search/local/GeneticSearch.java.html.
[27] T. Joachims, "A Support Vector Method for Multivariate Performance Measures",Proceedings of the International Conference on Machine Learning (ICML), 2005.
[28] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In ICML, pages 807–814, Corvalis, Oregon,2007.
[29]Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, Chih-Jen Lin, "LIBLINEAR: A Library for Large Linear Classification", Journal of Machine Learning Research 9(2008).
[30] http://www.cs.waikato.ac.nz/ml/weka/
[31] Nitesh V. Chawla, Kevin W.Bowyer, Lawrence O. Hall, W. Philip Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling TEchnique" in Journal of Artificial Intelligence Research, Volume 16, 321-357,2002.
[32] Emil Brissman, Kajsa Eriksson,"Classification: Grafted Decision Trees", Linkoping University, 2011.
[33] Ling, C., Huang, J., & Zhang, H. Auc: a better measure than accuracy in comparing learning algorithms. Proceedings of Canadian Artificial Intelligence Conference. (2003).
[34] Bradley, A.P, "The use of the area under the ROC curve in the evaluation of machine learning algorithms", Pattern Recognition 30(1997), 1145-1159.
[35] Lutz Hamel,"Model Assessment with ROC curves", The Encyclopedia of Data Warehousing and Mining, 2nd Edition, Idea Group Publishers, 2008.
[36] J. R. Quinlan and R. L. Rivest, "Inferring decision trees using the minimum description length principle," Information and computation, vol.80, no.3, pp.227-248, 1989.

Pijush Barthakur received the Master of Computer Application (M.C.A) degree from Dibrugarh University, India in 2001. Currently he is working as associate professor at Department of Computer Science & Engineering, Sikkim Manipal Institute of Technology, Sikkim, India. He is also pursuing his doctoral degree at Sikkim Manipal University. His research interests lie in the area of Network Security and Data Mining. He is currently a member of Technical Program Committee at 5th International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Beijing, China, 2013.

Dr. Manoj Dahal is currently working at Novell, India and his professional works mostly lie on File Access Protocol areas. He received the Ph.D degree on Networking from Tezpur University, India in 2008 for his thesis on Addressing Transport Layer Congestion Control Issues. He is also associated with research on Detection of Botnets using Machine Learning at Sikkim Manipal Institute of Technology, Sikkim, India. He has around 15 years experience in Software Industry. He was a Post-Doctoral Fellow for about a year with INRIA, France at LIP Labs, ENS de Lyon, where he has worked on Traffic Engineering for Optical Networks before working as a Professor for a short period with Sikkim Manipal Institute of Technology, Sikkim. Manoj also worked with Nokia (via Satyam) on routing devices and National Informatics Centre on e-Governance Projects in India before joining Novell.

Prof. (Dr.) M.K.Ghose is currently the Dean (R & D), SMIT and Professor and Head of the Department of Computer Science & Engineering at Sikkim Manipal Institute of Technology, Majitar, Sikkim, India since June, 2006. During June 2008 to June 2010, he had also carried out additional responsibilities of Head, SMU-IT. Prior to this, Dr. Ghose worked in the internationally reputed R & D organization ISRO – during 1981 to 1994 at Vikram Sarabhai Space Centre, ISRO, Trivandrum in the areas of Mission simulation and Quality & Reliability Analysis of ISRO Launch vehicles and Satellite systems and during 1995 to 2006 at Regional Remote Sensing Service Centre, ISRO, IIT Campus, Kharagpur in the areas of RS & GIS techniques for the natural resources management. His areas of research interest are Data Mining, Simulation & Modeling, Network, Sensor Network, Information Security, Optimization & Genetic Algorithm, Digital Image processing, Remote Sensing & GIS and Software Engineering and published 221 research papers in various national and international journals. Till date, he has produced 8 Ph.Ds and research assistance given for 2 Ph.Ds. Presently 11 scholars are pursuing Ph.D work under his guidance.

