Adoption of a Fuzzy Based Classification Model for P2P Botnet Detection

Pijush Barthakur[1], Manoj Dahal[2], and Mrinal Kanti Ghose[3]
(Corresponding author:Pijush Barthakur)
Department of Computer Applications, Sikkim Manipal Institute of Technology, Sikkim, India[1]
(Email: pijush.barthakur@gmail.com)
Novell IDC, Bagmane Tech Park, C V Ramannagar, Bangalore, India[2]
Department of Computer Science and Engineering, Sikkim Manipal Institute of Technology, Sikkim, India[3]
(Received Feb. 20, 2014; revised and accepted Nov. 15, 2014 & April 26, 2015)

Abstract

Botnet threat has increased enormously with adoption of newer technologies like root kit, anti-antivirus modules etc. by the hackers. Emergence of botnets having distributed C & C structure that mimic P2P technologically, has made its detection and dismantling extremely difficult. However, numeric flow feature values of P2P botnet C & C traffic can be used to generate fuzzy rule-set which can then be used to develop an efficient fuzzy based classification model. We generated fuzzy rule based models using Fuzzy Unordered Rule Induction Algorithm (FURIA) from C & C traffic collected from Nugache, Zeus and Waledac botnets. We also provide a comparative analysis of fuzzy based classification models with that of classification models obtained from C4.5 Decision Tree algorithm of Quinlan. Experimental results shows that using fuzzy based classification models, it is possible to achieve very promising result in predicting suspicious P2P botnet flows in the network and hence can be used for proactive detection of P2P botnets.

Keywords: Botnet, classification model, fuzzy unordered rule induction algorithm, P2P

1 Introduction

A botnet is a coordinated group of compromised machines controlled via Command & Control (C&C) communication channels that are connected to some C & C servers/peers and managed by botmasters/botherders. It can be used in performing various malicious activities like sending spam mails, distributed denial-of-service (DDoS) attacks, phishing attacks and click frauds [16]. Detection of such framework is never easy because of wide distribution of its bots and C & C servers that spread across thousands of individual networks around the globe. Thus, number of bots in a given network might be very less. Moreover, botnet C & C traffic is usually low in volume and is usually hidden in existing application traffic [18]. Complexities have further increased with shifting from its traditional Internet Relay Chat (IRC) protocol for C & C operations to more general and commonly used protocols like HTTP, POP3, Peer-to-Peer(P2P) etc. The most popular method of managing botnet C&Cs in recent past is the use of IRC, either in standard form or through use of customized implementations of IRC servers and clients intended to thwart mitigation efforts [26]. IRC based botnets uses centralized topological structures for C & C operations. Botnets with centralized C & C suffer from single-point-of-failure problem i.e. if C & C is detected and taken down the botnet cripples. However, IRC botnets with their source codes widely available and their setup and maintenance simple and relatively easy, are still the most popular among bot-herders. The newer and more resilient variants of botnet have emulated Peer-to-Peer technologically for C & C operations. A P2P botnet uses distributed C & C architecture that avoids single-point-of-failure problem. Also, newer P2P botnets are using advanced techniques like Rootkits, Fast-flux etc. to avoid detection. However, there also exists a newer and stealthy variant of centralized C & C architecture that uses HTTP for C & C operations. C & C traffic of HTTP botnets hide behind normal web traffic to pass through firewalls and other detection tools used by security personnel.

A pure peer-to-peer botnet is a decentralized architecture allowing bot-master to use any peer at random to distribute commands to other peer-bots in the P2P network. Some of the well-known P2P botnets are Nugache [26], Trojan.Peacomm or Storm [26] and Waledac [25]. Nugache is the pure-P2P bot artifact that does not depend on any central server including DNS. It handles C & C through encrypted P2P Channel using a variable bit length RSA key exchange, which is used to seed symmetric Rijndael-256 session keys for each peer connection. A new Nugache peer joins the network through an already known active servant peer in the network and each Nu-

gache peer may maintain a list of up to 100 servant peers for future use in rejoining the network. Nugache peers maintain an in-degree of connections that totals no more than ten clients at any time. The out-degree varies, but it is typically less than half of the ten-client limit. The result is a typical peer with at most about 13-15 connection active at any given time. Storm uses Overnet protocol initially to join the P2P network and then to keep track of the state of the network with other overnet peers. For initial peer seeding, the storm binary carries a text file containing IP addresses of approximately 300 static peers. However, the core of Storm C & C is handled via TCP using pull C & C technology directed at servers, i.e. server do not push commands down to the clients; rather, clients pull data from server. Storm uses a Hash mechanism for encrypting data requests to peers and servers. Instead of overnet, Waledac uses HTTP communication and a fast-flux based DNS network exclusively. In order to make initial contact with the botnet, each Waledac binary carries a list of IP addresses to use as a bootstrap list. Additional resiliency is provided in Waledac binaries through a hardcoded URL to access the botnet in the event a bot is unable to find an active node in the bootstrap list. The domain used for the URL is part of the fast flux network created by the botnet. Waledac botnet consists of three hierarchical layers of servers TSL servers, Upper tier servers (UTS) and Head End C&C. At the bottom are Repeater nodes and Spammer nodes. Those infected hosts having private IP addresses are spammer nodes. Repeaters are used primarily to move requests and replies between spammers and the head-end C & C server. Repeater nodes form the Repeater layer which is connected using P2P network. Each peer bot in the repeaters layer contains a node table having a maximum capacity of 500 to 1000 entries (depending on the version of binary). Waledac assigns each IP in the node table a timestamp, in order to keep this list as fresh as possible, by replacing the older entries with newer ones. P2P Zeus [2] is the decentralized version of the popular credential- stealing trojan Zeus. Earlier centralized version is mainly known for stealing banking credentials, where as P2P Zeus is also used for stealing Skype and MSN database files, Bitcoin wallets etc. With the adoption of P2P for communication, Zeus network has become more resilient against take down efforts. P2P Zeus uses RSA-2048 to sign sensitive messages originating from the bot-masters such as updates and proxy announcements. Peerlist poisoning is made difficult due to per-bot IP filter which only allows a single IP per / 20 subnet. P2P Zeus also includes an automatic blacklisting mechanism, which blacklists IPs that contact a bot too frequently in a specified time window. This mechanism further complicates efficient crawling and poisoning of the network. The C & C communications in a P2P Zeus network can be categorized into two parts:

1) Bots exchange binary and configuration updates with each other. P2P Zeus bots check the responsiveness of their neighbors every 30 minutes. Each neighbor is contacted in turn, and given 5 opportunities to reply. If a neighbor does not reply within 5 retries, it is deemed unresponsive, and is discarded from the peer list. During this verification round, every neighbor is asked for its current binary and configuration file version numbers. If a neighbor has an update available, the probing bot spawns a new thread to download the update.

2) Bots exchange list of proxy bots, which are designated bots where stolen data can be dropped and command can be retrieved. Additionally, bots exchange neighbor lists (Peer lists) with each other to maintain a coherent network.

Apart from P2P botnets, a new trend in the evolution of botnets is the rise of botnets that spread through social networking sites. One of the largest social networking botnet is KOOBFACE [27], and its infection starts with a spam sent through Facebook, Twitter, MySpace, or other social networking sites containing a catchy message with a link to a video. Each bot in the Koobface botnet connects to any one of roughly a hundred compromised hosts acting as C & C master servers that disseminate spam instructions. The Koobface C&C is a fully-connected graph where each master server is aware of every other master server.

In this paper, we propose a fuzzy rule-set through application of Fuzzy Unordered Rule Induction Algorithm (FURIA) [12] on flow attribute values of P2P botnet command & control traffic. A flow is defined by <source IP, destination IP, protocol, source port, destination port>. Fuzzy logic often leads to creation of small rule, where each rule is an embodiment of meaningful information. Moreover, we believe that there is an inherent fuzziness in security issues and an approximate fuzzy rule set can be generated for detection of security threats. In our earlier work, we proposed a rule induction algorithm [5] using indirect method of rule generation from C4.5 algorithm [21] of Quinlan. Inference using conventional rules proposed in paper [5] depends on crisp boundaries that lead to abrupt transition between the two classes. However, a more general rule where its support for a class decreases from full (inside the core of the rule) to zero (near the boundary) in a gradual rather than an abrupt way is more appropriate. Therefore, a set of fuzzy rules that have soft boundaries definitely has merit.

Our approach is a proactive novel approach for detecting likely P2P botnet C & C traffic flows through identification of significant flow-level features of P2P botnet C & C traffic. Flow level features are basically aggregation of packet-level features in that flow. Thus, our approach can handle encrypted traffic and is also free from privacy issues. The core of our detection approach relies on identification of likely botnet C & C traffic flows through development of efficient machine learning based models and then correlating the marked botnet flows to identify the group of flows that belong to the same bot-

net. Hypotheses that forms our detection approach are stated as follows:

1) A bot is a program and therefore has a limited set of commands and 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 and the manner in which a bot responds to a specific set of commands are also more-or-less uniform.

2) A P2P bot needs to keep itself updated about other bots that are still active in its network and therefore needs to keep communicating with them.

3) In normal C & C operations, a 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.

4) We also observe that most of the packets in bot generated flows are small in size i.e. the size of the largest packet in most of the bot generated flows is less than the MTU. This is to keep privacy and to avoid detection by not influencing normal internet services.

5) Among the few packets transferred in a bot generated flow,the largest sized packets are transferred at a specific proportion (usually < 1), whereas, the normal P2P traffic carries most of the packets to the size of MTU.

6) Finally, we observed that each bot generates mutually similar communicating flows to its peers in the same P2P botnet.

Rest of the paper is organized as follows: Section 2 provides a brief overview of related works. In Section 3, we provide a brief overview of botnet detection problem using network flows and the proposed architectural overview. In Section 4 we discuss our approach for dataset preparation and description of features selected for classification. In Section 5, we briefly describe the fuzzy rule generation algorithms used for botnet C & C traffic classification. In Section 6, we provide a detail analysis of results obtained from our classification models. In Section 7, we elaborate on future works and also the conclusion.

2 Related Works

Botnet threats have been continuously growing with adoption of newer technologies and propagation techniques by the bot-masters. Much resiliency has been achieved by recent botnets through migration from purely centralized C & C architecture to a partly or wholly decentralized architecture. New botnets have emerged on other digital devices like mobile phones or Smartphones. Mobile devices could send SMS and MMS to connect to their C & C proxy servers. Emergence of Online Social Network (OSNs) botnets is another recent development. A recently published survey paper [18] has vividly covered mobile botnets (e.g. iKee.B an iPhone bot) and OSNs botnets (e.g. KOOBFACE). Compared to enormity of threats posed by recent botnets, the detection and mitigation approaches developed till date is simply inadequate. Most of the botnet detection approaches are based on the anomalies being observed in network traffic, unusual system behavior etc. Botnet detection based on anomalies may not be useful always for several reasons. First, anomalies may not be always prominent to indicate a botnet attack. Second, it requires continuous monitoring of the network. Third, traffic belonging to botnets using HTTP protocol hides under the cover of normal web traffic and thus gets passed through everywhere.

There is a long time gap between initial infection with bot codes and its final deployment for active participation in botnets activity. This pre-attack period involves many stages such as, 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 the process of securing the newly created bot client. Measures taken to make a bot client secure normally involves deployment of anti-antivirus tools and Rootkit [20] 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 [24]. It is imperative to study botnet behavior during these early phases of exploitation in order to neutralize a bot possibly before its active participation in malicious activities. We may term such detection approaches as proactive. Many good reactive techniques [10, 11] have been suggested so far for botnet detection. Reactive techniques are about anomaly being observed in compromised machines mainly due to its use in cyber attacks or its exploitations over a long period of time. However, the aforementioned time gap is a good pointer for annihilation of bots before it causes any damage.

Lin et al. [14] proposed an automatic classification of obfuscated bot binaries by using system call sequences. The framework tested on 2256 binaries, achieves a 94% true positive rate and 93% true negative rate. A bot detection mechanism on a single host, proposed by Soniya et al. [3] initially identifies suspicious traffic by filtering out normal traffic from traffic generated on a host. For filtering out normal traffic, normal profiles of users are created. Suspicious traffic is then subjected to detail analysis based on observations made from characterization of bot traffic. A game bot detection framework through analysis of temporal characteristics of online game traffic has been proposed by Lu et al. [17]. The proposed approach is based on modelling of gaming behaviors of game bots

using Hidden Markov Models (HMM). The proposed approach can detect game bots accurately with only a small number of training traces.

Basheer et al. [1] proposed BotDigger, which utilizes fuzzy logic to derive logical rules based on defined botnet characteristics. The system uses fuzzy rule base to identify suspicious hosts within the set of monitored hosts after filtering out unlikely flows. The conserved flows are correlated with each other, looking for group of flows that may be part of same botnet. The fuzzy rule base is generated using few attributes that characterize IRC and HTTP based botnet activities. Kuochen et al. [28] uses fuzzy pattern recognition techniques to propose a novel behavior-based botnet detection system based on frequently observed bots TCP and DNS behaviors. The proposed system attempts to identify malicious domain names and IP addresses using maximum membership principle. The system achieved false positive rate of 0 - 3.08%, leaving room for further improvement. However, the proposed algorithm can detect inactive bots, which can be used to identify vulnerable hosts. Roshna et al. [23] proposed a botnet detection technique using Adaptive Neuro Fuzzy Inference System (ANFIS), which is a kind of neural network that incorporates the techniques of fuzzy inference system. However, there is scope for improvement in results claimed in this research work. Another fuzzy based intrusion detection system [9] has been proposed to detect anomalous network traffic by comparing with a behavioral model of normal network activity. A fuzzy rule based detection model is better suited for adaptive anomaly based intrusion detection when compared with static models due to changes in network traffic patterns over time.

3 Problem Description and Architectural Overview

Data captured from various applications in the internet involving different data types, such as files, e-mails, web contents, real-time audio/video data streams etc., are heterogeneous in terms of volume, time etc. Flows involving such data streams are in many cases unidirectional. Types of data transfer that needs to be reliable such as transfer of files, e-mails, Web contents etc. use Transmission Control Protocol (TCP) as transport layer protocol, whereas for transfer of real-time audio/video data streams, which is time-sensitive, the User Datagram Protocol (UDP) is typically used. Most P2P applications use UDP protocol for communication. Unlike normal web traffic, packets captured from botnets are largely uniform in terms of volume, time etc.

Network flows are extracted from packets captured from network traffic for both normal as well as botnet C & C traffic. Network flow level features used for classification are actually aggregate of packet-level features extracted from packet header. We derive twin advantages from our approach. First, we completely avoid packets payload analysis that involves a high amount of privacy issues. Second, our approach can handle encrypted traffic. Figure 1 shows architectural overview of our proposed Fuzzy rule based detection framework. It has two major components, first one is a module for extraction of flows from raw packets, and the second one generate fuzzy rules for classification.

4 Dataset Preparation and Feature Selection

We describe below the dataset preparation and feature selection procedure of our experiment.

4.1 Dataset Preparation

Botnet datasets used in this work were collected from the following sources: The Nugache botnet C & C traffic was obtained from Department of Computer Science, The University of Texas at Dallas. This is the same botnet traffic sample used in the botnet related research works of [19]. Similarly, Zeus and Waledac traffic traces were obtained from Department of Computer Science, University of Georgia. These are the botnet traces used in the botnet related research works of [22]. We acquired the benign data randomly from different machines in our campus network using Wireshark [13]. Our campus network is protected using network level firewall. Though this device has its own limitations, it is believed that the device can prevent malicious attacks entering the protected unit by setting different network zone and the rules that control the access in and out flow [16]. Moreover, we collected data from known benign applications only. Therefore, we assume that the data collected is benign.

We prepared three datasets having 20,000 flows each, one each for flow extracted for Nugache, Zeus and Waledac traces. Datasets are prepared in such a way, that each has 15000 flows of botnet C & C traffic and 5000 flows of benign traffic. Our benign traffic samples include varied traffic such as HTTP, FTP, SMTP etc. We also include traffic captured from legitimate P2P application in our benign dataset. Datasets are then labelled accordingly. While preparing the datasets, we discarded flows that are unlikely to contribute significantly in the process of classification viz.

1) Flows having single packet;

2) Flows that involves local area network broadcast activities.

Reasons for discarding these flows are as follows:

1) Flows carrying single packet does not carry any meaningful statistical information, and the proportion of largest sized packet attribute values in our dataset would become 1.

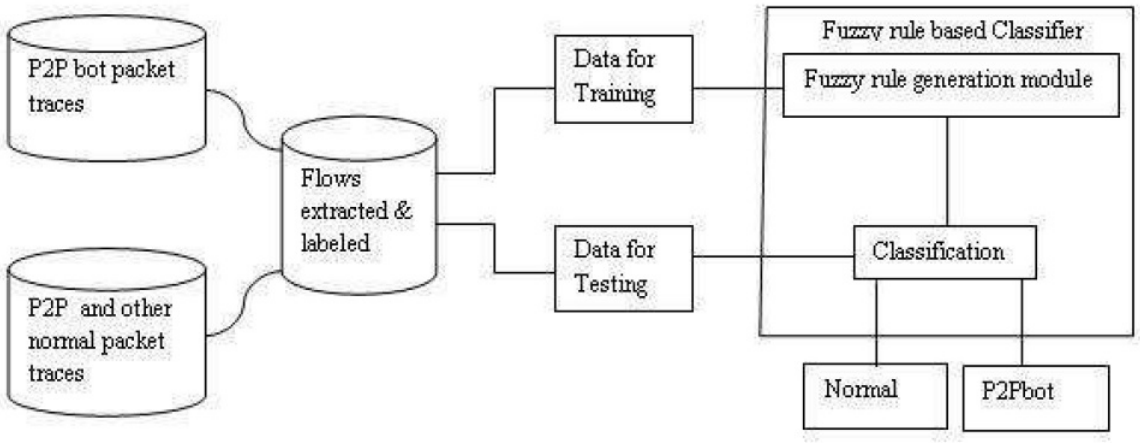

Figure 1: Architecture of the proposed Fuzzy rule based detection framework

2) The bot infected hosts may involve in local broadcasts activities. However, our objective is to consider host-to-host directed interaction in the network and broadcast traffic is never part of bot C & C interaction. Therefore, we tag such traffic as unwanted for our classification model. After removing unwanted flows, we scaled the datasets to the range of 0 to 1.

In our architectural framework shown in Figure 1, the first component has a module to extract flows from raw data. Then the attributes in the flows are scaled and useless flows are deleted. The final task of this component is to label the retained flows. The second component takes as input the dataset containing refined flows prepared by the first component and generates fuzzy rules for classification. We perform botnet C & C traffic classification using 10-fold cross validation. In general, in n-fold cross validation, the training set is first divide into n subsets of equal size. Sequentially one subset is tested using classifier trained on remaining n-1 subsets. Finally, when all subsets are tested, n results from folds are averaged to produce a single estimation.

4.2 Feature Selection

Detail analysis of behavioral characteristic of botnet C & C traffic flow was conducted after which, useful features for classification were extracted from packet headers. Following are the botnet flow and behavior characteristic features used in this work:

1) Total packets transferred (TPT): Number of packets transferred (or packet count) in a flow. It is a flow direction dependent attribute i.e. the numeric value of the attribute may be different for command and response flows within the same pair of peer bots.

2) Largest sized packet (LSP): Size of the packet carrying maximum bytes in a flow. It is also flow direction dependent attribute.

3) Total bytes transferred with largest sized packets (TBLSP): It is the multiplication of total number of largest sized packets (LSP) and the size of the largest packet.

4) Total bytes transferred (TBT): It is the summation of bytes transferred with all the packets in a flow. It is also flow direction dependent attribute.

5) Proportion of largest sized packet (PLSP): It is the ratio of largest sized packet transferred in a flow. It is also flow direction dependent attribute.

6) Variance of inter-arrival time (VIT): Variance calculated for inter-arrival time of packets within a flow. It is also flow direction dependent attribute.

7) Average packet length (APL): Average calculated for packet sizes of packets within a flow. It is also flow direction dependent attribute.

8) Variance of packet length (VPL): Variance calculated for sizes of packets within a flow. It is also flow direction dependent attribute.

9) Response packet difference (RPD): Difference in number of packets between two responding flows. The numeric value of this attribute is common for responding flows between a pair of hosts. For unidirectional flow (i.e. flow without a responding flow) we put a high numeric value for this attribute. For example, in our experiment we put 999, because the maximum difference is of three digits in our dataset.

10) Response time difference (RTD): Difference in time of last packet received for two responding flows between a pair of hosts. The numeric value of this attribute is also common for responding flows. For unidirectional flow (i.e. flow without a responding flow) we put a high numeric value for this attribute. For example, in our experiment we put 99999, because the maximum difference calculated in second is of five digits in our dataset.

5 Overview of FURIA

FURIA [12] is similar to well-known conventional rule learner RIPPER [7] with its distinctive features of generating fuzzy rules and of generating unordered rule sets instead of rule lists. By unordered, it means a set of rules for each class in a one-vs-rest scheme. In FURIA, fuzzy rules are obtained using fuzzy intervals derived with trapezoidal membership function. It uses four parameters stated as $I^F = (\Phi^{s,L}, \Phi^{c,L}, \Phi^{c,U}, \Phi^{s,U})$ to represent fuzzy intervals. The trapezoidal membership function for fuzzy sets (or fuzzy intervals) is given by:

$$I^F(v) \quad \stackrel{\text{df}}{=} \left\{ \begin{array}{ll} 1 & \Phi^{c,L} \leq v \leq \Phi^{c,U} \\ \frac{v-\Phi^{s,L}}{\Phi^{c,L}-\Phi^{s,L}} & \Phi^{s,L} < v < \Phi^{c,L} \\ \frac{\Phi^{s,U}-v}{\Phi^{s,U}-\Phi^{c,U}} & \Phi^{c,U} < v < \Phi^{s,U} \\ 0 & \text{else} \end{array} \right\} \quad (1)$$

$\Phi^{c,L}$ and $\Phi^{c,U}$ are, respectively, lower and upper bound of the core (elements with membership 1) of the fuzzy set; likewise, $\Phi^{s,L}$ and $\Phi^{s,U}$ are, respectively, the lower and upper bound of the support(elements with membership > 0). Thus, $\Phi^{s,L}$ and $\Phi^{s,U}$ are the fuzzy extensions of original RIPPER intervals [$\Phi^{c,L}$, $\Phi^{c,U}$] that are considered as core. Rules are fuzzified in a greedy way through fuzzification of every antecedent in a rule or in other word, through replacement of sharp boundaries of a rule with soft boundaries. Fuzzification of each antecedent is done by testing all relevant values $\{x_i | x = (x_1 \cdots x_k) \in D_i^T, x_i < \Phi_i^{c,L}\}$ as candidates for $\Phi_i^{s,L}$ and for all values $\{x_i | x = (x_1 \ldots x_k) \in D_i^T, x_i > \Phi_i^{c,U}\}$ as candidates for $\Phi_i^{s,U}$. Here, relevant data for each antecedent ($A_i \in I_i$) is the one considered by ignoring all those instances that are excluded by any other antecedent ($A_j \in I_j^F$), $j \neq i$: $D_T^i = \{x = (x_1 \ldots x_k) \in D_T | I_j^F(x_j) > 0$ for all $j \neq i\} \subseteq D_T$.

FURIA being a fuzzy rule generating algorithm is characterized by its core and its support. It is valid inside the core and invalid outside the support; in-between, the validity drops in a gradual way. Apart from having this definite advantage of fuzzy rule generation over other conventional rule generation algorithms such as RIPPER and C4.5, FURIA generates unordered rule set instead of rule lists and provides an efficient rule stretching method to deal with uncovered instances. All these features of the algorithm make it most suitable for rule generation for network security threat detection.

6 Results and Analysis

We use WEKA [8] Data Mining environment for fuzzy rule generation and subsequent classification of botnet C & C traffic flows of Nugache, Zeus and Waledac botnets. Weka provides a collection of Machine Learning (ML) algorithms and several visualization tools for data analysis and predictive modelling. We present results of our experiments in two parts: First, a brief analysis of structure of fuzzy rules generated for all bot flows is presented. Next we provide analysis of results using various performance metrics.

6.1 Analysis of Rule Sets

Unlike sharp boundaries generated by RIPPER, C4.5 etc. a fuzzy rule is characterized by soft boundaries. Each fuzzy rule consists of two parts: its "core"and its "support". For example, one of the rules generated from C & C traffic of Nugache botnet is:

(TBLSP in [-inf, -inf, 0.000006, 0.000006]) and (RPD in [-inf, -inf, 0.001, 0.002]) and (LSP in [0.0055, 0.0062, inf, inf]) ⇒ Class=bot (CF = 1.0).

The antecedents of the rule can be interpreted as: (1)TBLSP in [-inf, -inf, 0.000006, 0.000006]: it is valid for TBLSP $\leq$ 0.000006 and invalid for TBLSP $>$ 0.000006, (2) RPD in [-inf, -inf, 0.001, 0.002]: it is completely valid for RPD $\leq$ 0.001, invalid for RPD $>$ 0.002 and partially valid in-between,(3) LSP in [0.0055, 0.0062, inf, inf]: it is completely valid for LSP $\geq$ 0.0062, invalid for LSP $<$ 0.0055 and partially valid in-between. Now, performing logical AND operation for completely valid and partially valid cases of Part (1), (2) and (3) on the LHS of our above rule, we get the 'Coverage'of the rule in our dataset. Completely valid parts associated with antecedents of the rule are its 'core', whereas partially valid part forms the 'support'. The Certainty factor of the rule is 1. List of fuzzy rules generated that predict bot flows for Nugache, Zeus and Waledac are shown in Tables 1, 2 and 3 respectively.

In Table 4, we provide structural attribute values for comparative analysis of the structure of fuzzy rule sets generated for the three botnets. The attributes considered for comparison are: number of fuzzy rules generated (NFR), average number of antecedents in the rules generated for each botnet (ANAR), number of rules that predicts a bot flow (NRB), percentage of coverage of cases (PCC) and the number of rules with certainty factor 1.0 (NRCF).

From the structural attribute values, we find that least complex rules are generated for Waledac C & C traffic flows and the most complex rule set is generated for highly stealthy Zeus botnet. This can be observed from the number of fuzzy rules generated (NFR) and the average number of antecedents in the rules generated for each botnet (ANAR) attributes of the three botnet C & C traffic samples. Among the other attributes, Nugache rule set has 52% rules predicting bot flows followed by 42.5% for Zeus and 42.1% for Waledac. Similarly, percentage of rules with certainty factor 1.0 is 72% for Nugache, 47% for Waledac and 38.75% for Zeus. All these statistics along with the percentage of coverage of cases by the rule sets indicates that all the three botnet traces produced very efficient rule set.

Table 1: Fuzzy rules for detection of Nugache bot C & C traffic

Serial No	Rule
1	(TBLSP in [-inf, -inf, 0.000006, 0.000006]) and (RPD in [-inf, -inf, 0.001, 0.002]) and (LSP in [0.0055, 0.0062, inf, inf]) ⇒ Class=bot (CF = 1.0)
2	(TBLSP in [-inf, -inf, 0.000012, 0.000013]) and (TBT in [0.000034, 0.000036, inf, inf]) and (TBLSP in [0.000012, 0.000012, inf, inf]) and (LSP in [-inf, -inf, 0.0118, 0.0119]) and (APL in [-inf, -inf, 0.0072, 0.00828]) ⇒ Class=bot (CF = 1.0)
3	(LSP in [-inf, -inf, 0.0062, 0.0064]) and (VPL in [0, 0, inf, inf]) ⇒ Class=bot (CF = 1.0)
4	(APL in [-inf, -inf, 0.00725, 0.00828]) and (VPL in [0.000001, 0.000032, inf, inf]) and (LSP in [-inf, -inf, 0.0118, 0.0119]) and (LSP in [0.0115, 0.0118, inf, inf]) ⇒ Class=bot (CF = 1.0)
5	(TBLSP in [-inf, -inf, 0.000006, 0.000006]) and (TPT in [0.00003, 0.00004, inf, inf]) ⇒ Class=bot (CF = 1.0)
6	(APL in [-inf, -inf, 0.006133, 0.0062]) and (APL in [0.006, 0.006009, inf, inf]) and (VPL in [-inf, -inf, 0, 0.000001]) and (RTD in [-inf, -inf, 0.03273, 0.03388]) ⇒ Class=bot (CF = 1.0)
7	(APL in [-inf, -inf, 0.006189, 0.0062]) and (APL in [0.006, 0.006006, inf, inf]) and (VPL in [-inf, -inf, 0, 0]) and (VIT in [0, 0.0343, inf, inf]) and (RTD in [-inf, -inf, 0.06001, 0.06455]) ⇒ Class=bot (CF = 1.0)
8	(PLSP in [-inf, -inf, 0.04, 0.333333]) and (LSP in [0.0275, 0.0354, inf, inf]) and (TBLSP in [-inf, -inf, 0.000047, 0.000049]) and (VPL in [-inf, -inf, 0.000033, 0.00004]) ⇒ Class=bot (CF = 0.95)
9	(APL in [-inf, -inf, 0.0075, 0.00828]) and (LSP in [0.0115, 0.0118, inf, inf]) and (LSP in [-inf, -inf, 0.0118, 0.0119]) ⇒ Class=bot (CF = 1.0)
10	(APL in [-inf, -inf, 0.006093, 0.006133]) and (APL in [0.006, 0.006006, inf, inf]) and (VPL in [-inf, -inf, 0.000002, 0.000003]) and (TPT in [0.00008, 0.00009, inf, inf]) ⇒ Class=bot (CF = 0.99)
11	(PLSP in [-inf, -inf, 0.007937, 0.009524]) and (LSP in [0.0276, 0.0289, inf, inf]) and (VPL in [-inf, -inf, 0.00008, 0.000083]) and (RPD in [-inf, -inf, 0.154, 0.19]) ⇒ Class=bot (CF = 0.95)
12	(TPT in [0.00009, 0.00025, inf, inf]) and (LSP in [-inf, -inf, 0.006, 0.0065]) and (LSP in [0.0054, 0.0058, inf, inf]) ⇒ Class=bot (CF = 0.92)
13	(PLSP in [-inf, -inf, 0.04, 0.111111]) and (TBT in [-inf, -inf, 0.00034, 0.000344]) and (LSP in [0.0309, 0.0354, inf, inf]) and (VPL in [-inf, -inf, 0.000352, 0.000363]) ⇒ Class=bot (CF = 0.92)

6.2 Analysis of Classification Results

Final datasets prepared from botnet C & C traffic of the three bots under consideration are being used to build classification models using WEKA machine learning tools. We randomized flow instances in our datasets by passing it through Randomize filter available with WEKA's unsupervised instance filter category. This was necessitated because our original datasets are imbalanced having less normal web flows. While constructing classifier, we used 10-fold cross validation so that there is no over-fitting of our training set.

Results of Classification task by any classification algorithm during testing are usually displayed in a confusion matrix. A confusion matrix holds the count of the correct and incorrect classification from each class or the differences between the true and predicted classes for a set of labelled instances. Table 5 shows the format of a confusion matrix with TP, TN, FP, FN representing True Positive, True Negative, False Positive and False Negative counts respectively.

The row total, CN and CP are the number of truly negative and positive instances. Similarly, RN and RP are the number of predicted negative and positive instances, with N being the total number of instances ($N = CN+CP = RN+RP$). Although confusion matrix incorporates all the performance measures of a classification algorithm, more meaningful results can be extracted from it to represent certain performance criteria. Accuracy is the first performance criteria we are using to compare the three classification models on botnet datasets:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2)$$

Figure 2 shows comparison of accuracy achieved with our fuzzy rule based classification models with that of decision tree based classification models. Decision trees can be used to generate crisp rule sets for classification. In this work we generate the decision tree from Quinlans famous C4.5 algorithm.

The percentage accuracy value achieved using FURIA are 99.745%, 99.715%, and 99.105% for Nugache, Waledac and Zeus flows respectively. Corresponding figures using C4.5 algorithm are 99.655%, 99.695%, and 98.615%. We find a distinct increase in correctly classified instances using fuzzy rule based classification models. We also observe that fuzzy based classifier is largely successful in classifying C & C traffic generated by stealthy botnets like Zeus, though accuracy achieved is lower than that of Nugache and Waledac. The increase in number of correctly classified flow instances by FURIA when compared with C4.5 algorithm is 18, 4 and 98 respectively for Nugache, Waledac and Zeus sample botnet datasets. This increase is because of fuzzification of classification rules by FURIA.

We also consider the following additional performance criteria to compare our fuzzy based classification models:

$$Sensitivity = \frac{TP}{TP + FN} \quad (3)$$

$$PositivePredictiveValue(PPV) = \frac{TP}{TP + FP} \quad (4)$$

$$FalsePositiveRate = \frac{FP}{FP + TN} \quad (5)$$

Sensitivity (or True Positive Rate) is the proportion of correctly identified bot flows out of total flows labelled as bot. Similarly, PPV or Precision is the proportion of correctly identified bot flows out of total predicted bot flows. Figure 3 shows graphical comparison of Sensitivity, PPV and FP rate of our three fuzzy based classification models. The graph shows a three dimensional view of changes in aforesaid three performance metrics values with respect to the three botnet sample data set in consideration. The results shown in the graph are in the range

Table 2: Fuzzy rules for detection of Zeus bot C & C traffic

Serial No	Rule
1	(VPL in [0.00017, 0.000171, inf, inf]) and (TPT in [-inf, -inf, 0.0006, 0.0007]) and (APL in [-inf, -inf, 0.013633, 0.015883]) and (LSP in [0.0308, 0.0324, inf, inf]) and (RTD in [-inf, -inf, 0.00252, 0.00253]) and (RTD in [0.00099, 0.0011, inf, inf]) ⇒ Class=bot (CF = 1.0)
2	(VPL in [0.043302, 0.043385, inf, inf]) and (TPT in [-inf, -inf, 0.0006, 0.0007]) and (VPL in [-inf, -inf, 0.049932, 0.052853]) and (RPD in [-inf, -inf, 0.001, 0.002]) ⇒ Class=bot (CF = 1.0)
3	(VPL in [0.000032, 0.000171, inf, inf]) and (LSP in [-inf, -inf, 0.02, 0.0202]) and (LSP in [0.0199, 0.02, inf, inf]) and (APL in [-inf, -inf, 0.00925, 0.01595]) ⇒ Class=bot (CF = 1.0)
4	(VPL in [0.000149, 0.00015, inf, inf]) and (LSP in [-inf, -inf, 0.0545, 0.0547]) and (LSP in [0.0523, 0.0529, inf, inf]) and (RTD in [-inf, -inf, 0.01448, 0.01463]) and (RTD in [0.00082, 0.001, inf, inf]) ⇒ Class=bot (CF = 1.0)
5	(VPL in [0.00006, 0.00015, inf, inf]) and (RPD in [-inf, -inf, 0.006, 0.999]) and (RPD in [0.001, 0.002, inf, inf]) and (RTD in [0.00497, 0.00805, inf, inf]) and (TBLSP in [-inf, -inf, 0.000055, 0.000062]) and (APL in [0.012533, 0.0126, inf, inf]) and (LSP in [-inf, -inf, 0.0335, 0.0365]) and (TBT in [0.00004, 0.000044, inf, inf]) ⇒ Class=bot (CF = 1.0)
6	(VPL in [0.00009, 0.000965, inf, inf]) and (RPD in [-inf, -inf, 0.001, 0.002]) and (RTD in [0.00105, 0.00122, inf, inf]) and (RTD in [-inf, -inf, 0.00218, 0.00219]) ⇒ Class=bot (CF = 1.0)
7	(VPL in [0.000656, 0.000942, inf, inf]) and (RPD in [-inf, -inf, 0.006, 0.043]) and (RTD in [0.00069, 0.00122, inf, inf]) and (LSP in [0.0523, 0.0529, inf, inf]) and (LSP in [-inf, -inf, 0.0546, 0.0547]) and (VPL in [-inf, -inf, 0.003206, 0.003209]) and (RTD in [-inf, -inf, 0.07234, 0.0724]) ⇒ Class=bot (CF = 1.0)
8	(RPD in [-inf, -inf, 0.012, 0.017]) and (TBLSP in [0.000012, 0.000018, inf, inf]) and (LSP in [-inf, -inf, 0.0062, 0.0066]) and (RTD in [-inf, -inf, 0.01018, 0.01052]) and (RTD in [0.00068, 0.00122, inf, inf]) ⇒ Class=bot (CF = 1.0)
9	(VPL in [0.000091, 0.00015, inf, inf]) and (RPD in [-inf, -inf, 0.012, 0.043]) and (RTD in [0.00001, 0.00122, inf, inf]) and (APL in [0.018127, 0.018409, inf, inf]) and (PLSP in [-inf, -inf, 0.142857, 0.166667]) and (VPL in [-inf, -inf, 0.002537, 0.002894]) and (RTD in [-inf, -inf, 0.06149, 0.06156]) ⇒ Class=bot (CF = 1.0)
10	(VPL in [0.000976, 0.001038, inf, inf]) and (RPD in [-inf, -inf, 0.005, 0.006]) and (RTD in [0.00078, 0.00091, inf, inf]) and (RTD in [-inf, -inf, 0.00332, 0.00818]) and (TBLSP in [-inf, -inf, 0.000034, 0.000036]) ⇒ Class=bot (CF = 1.0)
11	(VPL in [0.039564, 0.041469, inf, inf]) and (RTD in [0.00061, 0.00097, inf, inf]) and (RTD in [-inf, -inf, 0.03579, 0.04022]) and (TBLSP in [-inf, -inf, 0.000442, 0.000454]) and (TPT in [0.0002, 0.0003, inf, inf]) ⇒ Class=bot (CF = 1.0)
12	(VPL in [0.001119, 0.001122, inf, inf]) and (RTD in [0.00045, 0.001, inf, inf]) and (RPD in [-inf, -inf, 0.006, 0.999]) and (LSP in [0.0768, 0.0772, inf, inf]) and (LSP in [-inf, -inf, 0.0811, 0.0814]) ⇒ Class=bot (CF = 1.0)
13	(VPL in [0.000079, 0.000091, inf, inf]) and (RTD in [0.0493, 0.04944, inf, inf]) and (LSP in [0.0527, 0.0529, inf, inf]) and (LSP in [-inf, -inf, 0.055, 0.0552]) and (APL in [0.017975, 0.0182, inf, inf]) ⇒ Class=bot (CF = 0.99)
14	(VPL in [0.025873, 0.03012, inf, inf]) and (APL in [0.091857, 0.093214, inf, inf]) and (TBLSP in [-inf, -inf, 0.000757, 0.000908]) and (RTD in [0.00045, 0.00255, inf, inf]) and (RTD in [-inf, -inf, 0.0676, 0.06776]) ⇒ Class=bot (CF = 1.0)
15	(VPL in [0.002699, 0.002756, inf, inf]) and (LSP in [-inf, -inf, 0.0531, 0.0532]) and (LSP in [0.0527, 0.0529, inf, inf]) ⇒ Class=bot (CF = 1.0)
16	(VPL in [0.000065, 0.000091, inf, inf]) and (LSP in [-inf, -inf, 0.0335, 0.0343]) and (TBT in [0.000116, 0.000117, inf, inf]) and (APL in [0.016038, 0.01607, inf, inf]) and (TBLSP in [-inf, -inf, 0.000086, 0.000105]) ⇒ Class=bot (CF = 1.0)
17	(VPL in [0.001417, 0.001493, inf, inf]) and (TBLSP in [-inf, -inf, 0.000055, 0.000055]) and (LSP in [0.0522, 0.053, inf, inf]) and (RPD in [0.002, 0.003, inf, inf]) ⇒ Class=bot (CF = 1.0)
18	(VPL in [0.001059, 0.001108, inf, inf]) and (TBLSP in [-inf, -inf, 0.000034, 0.000034]) and (TBT in [0.00005, 0.000051, inf, inf]) and (TPT in [0.0003, 0.0004, inf, inf]) ⇒ Class=bot (CF = 1.0)
19	(VPL in [0.029541, 0.03012, inf, inf]) and (APL in [0.111343, 0.112722, inf, inf]) and (VIT in [-inf, -inf, 0.0123, 0.0149]) and (TBLSP in [-inf, -inf, 0.001211, 0.001363]) ⇒ Class=bot (CF = 0.99)
20	(RTD in [-inf, -inf, 0.00686, 0.02146]) and (RTD in [0.00066, 0.00073, inf, inf]) and (APL in [-inf, -inf, 0.00644, 0.006467]) and (LSP in [0.0074, 0.0098, inf, inf]) and (LSP in [-inf, -inf, 0.0098, 0.0122]) ⇒ Class=bot (CF = 0.99)
21	(VPL in [0.001957, 0.003153, inf, inf]) and (APL in [0.142629, 0.143664, inf, inf]) and (PLSP in [-inf, -inf, 0.536036, 0.676471]) ⇒ Class=bot (CF = 0.98)
22	(VPL in [0.040972, 0.041469, inf, inf]) and (RPD in [-inf, -inf, 0.005, 0.006]) and (RTD in [0.00047, 0.00058, inf, inf]) and (TBT in [0.000649, 0.000653, inf, inf]) ⇒ Class=bot (CF = 0.98)
23	(VPL in [0.01728, 0.022948, inf, inf]) and (RPD in [-inf, -inf, 0.006, 0.007]) and (RTD in [0.00795, 0.01713, inf, inf]) and (APL in [0.111506, 0.112722, inf, inf]) and (VIT in [-inf, -inf, 0.0123, 0.0139]) ⇒ Class=bot (CF = 0.98)
24	(LSP in [-inf, -inf, 0.0062, 0.0065]) and (TBLSP in [0.000013, 0.000018, inf, inf]) and (RTD in [-inf, -inf, 0.06825, 0.07552]) and (LSP in [0.006, 0.0062, inf, inf]) ⇒ Class=bot (CF = 0.98)
25	(VPL in [0.000088, 0.000116, inf, inf]) and (RPD in [-inf, -inf, 0.006, 0.008]) and (RPD in [0.003, 0.004, inf, inf]) and (VPL in [-inf, -inf, 0.000162, 0.000162]) ⇒ Class=bot (CF = 0.99)
26	(VPL in [0.00071, 0.000712, inf, inf]) and (LSP in [-inf, -inf, 0.0268, 0.0845]) and (TBT in [0.000036, 0.000037, inf, inf]) and (PLSP in [-inf, -inf, 0.4, 0.428571]) ⇒ Class=bot (CF = 0.93)
27	(VPL in [0.001417, 0.001905, inf, inf]) and (LSP in [-inf, -inf, 0.0548, 0.0549]) and (LSP in [0.0531, 0.0532, inf, inf]) and (APL in [-inf, -inf, 0.017625, 0.02075]) and (RPD in [0.001, 0.002, inf, inf]) ⇒ Class=bot (CF = 1.0)
28	(LSP in [-inf, -inf, 0.006, 0.0062]) and (TBLSP in [0.000016, 0.000018, inf, inf]) and (RTD in [-inf, -inf, 0.02393, 0.02514]) and (VIT in [0.0157, 0.0199, inf, inf]) ⇒ Class=bot (CF = 0.99)
29	(VPL in [0.026365, 0.028376, inf, inf]) and (RPD in [-inf, -inf, 0.004, 0.012]) and (LSP in [-inf, -inf, 0.1494, 0.1496]) and (LSP in [0.1468, 0.1472, inf, inf]) ⇒ Class=bot (CF = 1.0)
30	(LSP in [-inf, -inf, 0.0062, 0.0065]) and (TBLSP in [0.000018, 0.000019, inf, inf]) and (RPD in [-inf, -inf, 0.001, 0.006]) and (RTD in [-inf, -inf, 0.06889, 0.07552]) ⇒ Class=bot (CF = 0.99)
31	(RTD in [-inf, -inf, 0.05404, 0.06343]) and (APL in [0.009754, 0.012425, inf, inf]) and (TBLSP in [-inf, -inf, 0.000033, 0.000037]) and (TBT in [0.000112, 0.000116, inf, inf]) and (VIT in [0.0161, 0.0163, inf, inf]) ⇒ Class=bot (CF = 1.0)
32	(VPL in [0.000093, 0.000539, inf, inf]) and (TBLSP in [-inf, -inf, 0.000027, 0.000027]) and (LSP in [0.0254, 0.0255, inf, inf]) and (TBT in [-inf, -inf, 0.000056, 0.000057]) ⇒ Class=bot (CF = 0.89)
33	(VPL in [0.000744, 0.000913, inf, inf]) and (PLSP in [-inf, -inf, 0.03125, 0.035714]) and (LSP in [-inf, -inf, 0.055, 0.0686]) ⇒ Class=bot (CF = 1.0)
34	(RPD in [-inf, -inf, 0.001, 0.002]) and (VPL in [0.023281, 0.02549, inf, inf]) and (LSP in [-inf, -inf, 0.121, 0.1221]) ⇒ Class=bot (CF = 0.96)

Table 3: Fuzzy rules for detection of Waledac bot C & C traffic

Serial No	Rule
1	(APL in [-inf, -inf, 0.007491, 0.0075]) and (VPL in [0, 0.000001, inf, inf]) and (LSP in [-inf, -inf, 0.0062, 0.0064]) ⇒ Class=bot (CF = 1.0)
2	(LSP in [-inf, -inf, 0.0096, 0.0098]) and (LSP in [0.0094, 0.0096, inf, inf]) and (TPT in [0.0002, 0.0003, inf, inf]) ⇒ Class=bot (CF = 1.0)
3	(LSP in [-inf, -inf, 0.0062, 0.0065]) and (TBLSP in [0.000013, 0.000016, inf, inf]) and (RPD in [-inf, -inf, 0.002, 0.999]) and (LSP in [0.006, 0.0062, inf, inf]) ⇒ Class=bot (CF = 0.99)
4	(APL in [-inf, -inf, 0.005533, 0.006]) and (RTD in [0.01125, 0.01838, inf, inf]) and (TPT in [0.0002, 0.0003, inf, inf]) and (RPD in [-inf, -inf, 0, 0.001]) ⇒ Class=bot (CF = 1.0)
5	(APL in [-inf, -inf, 0.005533, 0.0059]) and (RTD in [0.00001, 0.00795, inf, inf]) and (RPD in [0, 0.001, inf, inf]) and (TBLSP in [-inf, -inf, 0.000011, 0.000012]) ⇒ Class=bot (CF = 0.98)
6	(APL in [-inf, -inf, 0.005533, 0.0058]) and (TPT in [0.0002, 0.0003, inf, inf]) and (RPD in [-inf, -inf, 0, 0.002]) ⇒ Class=bot (CF = 1.0)
7	(APL in [-inf, -inf, 0.005933, 0.00605]) and (TBLSP in [-inf, -inf, 0.000006, 0.000006]) and (LSP in [0.0055, 0.0058, inf, inf]) ⇒ Class=bot (CF = 1.0)
8	(APL in [-inf, -inf, 0.005933, 0.005967]) and (RPD in [-inf, -inf, 0.001, 0.002]) and (VPL in [0.000003, 0.000003, inf, inf]) and (VIT in [-inf, -inf, 0.0156, 0.018262]) ⇒ Class=bot (CF = 0.97)

Table 4: Structural attribute values of fuzzy rule sets

	NFR	ANAR	NRB	PCC	NRCF
Nugache	25	3.04	13	99.845%	18
Waledac	19	2.89	08	99.8%	09
Zeus	80	4.1	34	99.57%	31

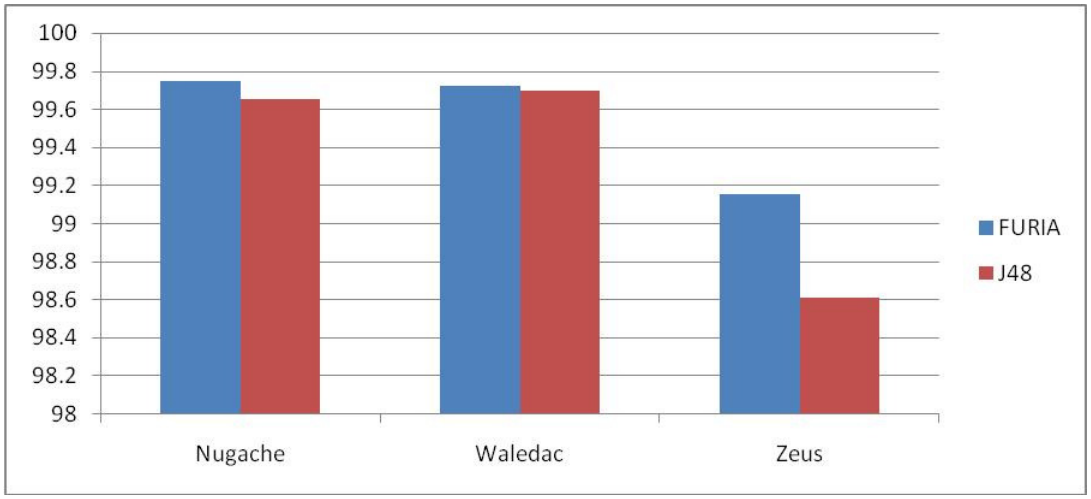

Figure 2: Percentage of accuracy

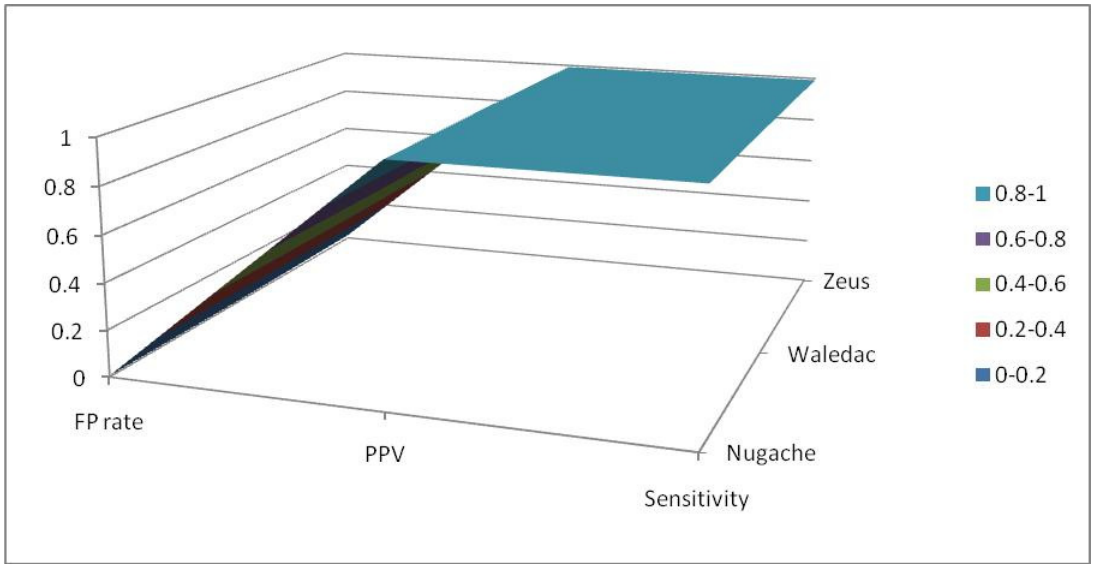

Figure 3: FP rate, PPV and sensitivity

of 0 to 1, as the performance measures are within this range only. Our fuzzy classifier produces the following results: (1) Sensitivity and PPV are 0.997 for both Nugache, Waledac traces, and 0.991 for Zeus. (2) FP rate is 0.005 for Nugache, 0.006 for Waledac and 0.017 for Zeus.

Table 5: A confusion matrix

	Predicted Class		
True class	-VE	+VE	
-VE	TN	FP	CN
+VE	FN	TP	CP
	RN	RP	N

Sensitivity, PPV and FP rate are inferior for Zeus sample dataset compared to that of Nugache and Waledac. From our analysis of sample datasets we find that Nugache and Waledac C & C flow samples are more distinguishable from normal traffic samples than the Zeus C & C flow sample. Following are the steps performed to do the analysis:

1) From the list of ten features in our feature set we proceed with two most influential pair of features i.e. Largest sized packet (LSP) and Proportion of largest sized packet (PLSP). Influence of a particular feature on classifiers performance has been judged through a simple performance-based input ranking methodology as has been described in our previous work [4].

2) We then removed the repeated values for this pair of features in all the datasets. After removal of duplicates we are left with only distinct values for each instance. The percentage of distinct combination obtained for Nugache is 0.313%, for Waledac it is 0.307%, for Zeus it is 5.887% and for Normal flow instances it is 28.3%.

3) We then calculated the percentage of distinct combinations having more than 1000 bytes in LSP for each dataset. We found that none of the packets in Nugache and Waledac datasets carry a payload of greater than or equal to 1000 bytes. For Zeus, the percentage of distinct combinations having more than 1000 bytes in LSP is 0.733% and for Normal the value is 7.64%.

From Step 2 we find that Zeus has a significantly higher percentage of distinct combinations compared to Nugache and Waledac. Similarly, from Step 3 we find that Zeus also has a good number of flows with LSP having more than 1000 bytes. Therefore, it is not difficult to ascertain that the classification error rate of Zeus is bound to be more compared to Nugache or Waledac.

The accuracy (the rate of correct classification) measure of a classifier is often used for comparison of predictive ability of learning algorithms. However, the accuracy measure completely ignores the probability estimations of the classification systems. Probability estimations generated by most classifiers can be used for ranking instances which gives likelihood estimations of instances and is therefore more desirable than just a classification. The AUC (area under the curve) of the ROC (Receiver Operating Characteristic) curve provides an alternative and better measure for machine learning algorithms by being invariant to the decision criterion selected, prior probabilities and is easily extendable to include cost/benefit analysis [6, 15]. ROC curve represents plotting of True Positive Rate against False Positive Rate as the decision threshold is varied, that can be used to compare the classifiers performance across the entire range of class distributions and error costs.

With a varied decision threshold and already obtained number of points on the ROC curve [FP rate = α, TP rate = 1 - β], the area under the ROC curve can be calculated by using the trapezoidal integration as follows:

$$AUC = \sum_i \{(1 - \beta_i . \triangle\alpha + \frac{1}{2}[\triangle(1-\beta).\triangle\alpha]\} \quad (6)$$

where, $\triangle(1-\beta) = (1-\beta_i) - (1-\beta_{i-1})$, $\triangle\alpha = \alpha_i - \alpha_{i-1}$.

In case of perfect predictions the AUC is 1 and if AUC is 0.5 the prediction is random. We provide a comparative analysis of our classification models using AUC values. Table 6 provides the AUC measures of our fuzzy based classification models and its corresponding values for decision tree based classification models. We find AUC measure for Zeus is significantly better in case of the fuzzy based classifier compared to the decision tree model, whereas for Nugache the fuzzy based classifier has a marginal edge over the one based on decision tree. The only exception is Waledac, where we have the AUC measure of decision tree classifier edge past the fuzzy classifier, though very marginally. Our explanation to this is that both these classifiers generates almost perfect classification models with equally good results for Waledac botnet C & C traffic sample, which is apparent from Figure 2 and Figure 3 having accuracy, sensitivity and PPV measures. AUC measure of a particular classification model is calculated through generation of a rank list based on probability estimations of instances. Thus it is not necessary that AUC measure of a classifier has to be higher compared to another classifier just because its other measures like accuracy, sensitivity, PPV etc. are on higher side. In fact, it implies that the error rate of decision tree based classifier generated from Waledac C & C traffic sample is slightly higher compared to fuzzy based classifier even though the decision tree classifier performs marginally better in terms of AUC measure. Nevertheless, from analysis of results we find that AUC measures of FURIA are much more consistent providing excellent predictions.

Table 6: AUC measures of fuzzy and decision tree based classification models

	FURIA	**C4.5**
Nugache	0.997	0.995
Waledac	0.997	0.998
Zeus	0.994	0.984

7 Conclusions

A fuzzy rule based detection framework for P2P botnets is presented here. The proposed approach leverages on flow level features and packet level features of network traffic to build excellent classification model for P2P botnet C & C traffic. The accuracy achieved by our system is as good as 99.745%, 99.715%, and 99.105% for Nugache, Waledac and Zeus botnet samples respectively. The fuzzy rule based approach is a supervised one and hence can detect known botnet traces only. P2P botnets have distributed C & C architecture and therefore complete annihilation of existing botnets is not easy. However, using our fuzzy rule based classification model, we can track botnet C & C traffic pro-actively as well as with high accuracy. In future, our effort will be to build similar detection model for botnets using other protocols and communication technologies such as social network based botnets, mobile botnets etc.

Acknowledgments

The authors would like to thank Mohammad M. Masud, Department of Computer Science, University of Texas at Dallas and Babak Rahbarinia, Department of Computer Science, University of Georgia, USA, for providing botnet traffic sample to carry out this research work. The authors gratefully acknowledge the anonymous reviewers for their valuable comments.

References

[1] B. Al-Duwairi and L. Al-Ebbini, "Botdigger: A fuzzy inference system for botnet detection," in *The Fifth International Conference on Internet and Web Applications and Services*, pp. 16–21, May 2010.

[2] D. Andriesse, C. Rossow, B. Stone-Gross, D. Plohmann, and H. Bos, "Highly resilient peer-to-peer botnets are here: An analysis of gameover zeus," in *Proceedings of the 8th IEEE International Conference on Malicious and Unwanted Software (MALWARE'13)*, pp. 116–123, Fajardo, Puerto Rico, USA, 2013.

[3] S. Balram and M. Wilscy, "User traffic profile for traffic reduction and effective bot C & C detection," *International Journal of Network Security*, vol. 16, no. 1, pp. 46–52, 2014.

[4] P. Barthakur, M. Dahal, and M. K. Ghose, "A framework for P2P botnet detection using SVM," in *4th International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pp. 195–200, Oct. 2012.

[5] P. Barthakur, M. Dahal, and M. K. Ghose, "An efficient machine learning based classification scheme for detecting distributed command & control traffic of P2P botnets," *International Journal of Modern Education and Computer Science*, vol. 5, no. 10, pp. 9–18, 2013.

[6] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[7] W. Cohen, "Fast effective rule induction," in *12th International Conference on Machine Learning*, pp. 115–123, 1995.

[8] R. Dimov, "Weka: Practical machine learning tools and techniques with Java implementations," in *AI Tools Seminar University of Saarland*, June 2007. (http://researchcommons.waikato.ac.nz/bitstream/handle/10289/1040/uow-cs-wp-1999-11.pdf?sequence=1&isAllowed=y)

[9] F. Geramiraz, A. S. Memaripour, and M. Abbaspour, "Adaptive anomaly-based intrusion detection system using fuzzy controller," *International Journal of Network Security*, vol. 14, no. 6, pp. 352–361, 2012.

[10] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *17th USENIX Security Symposium*, pp. 139–154, Dec. 2008.

[11] G. Gu, V. Yegneswaran, P. Porras, J. Stoll, and W. Lee, "Active botnet probing to identify obscure command and control channels," in *Annual Computer Security Applications Conference*, pp. 241–253, 2009.

[12] J. Huhn and E. Hullermeier, "FURIA: An algorithm for unordered fuzzy rule induction," *Data Mining and Knowledge Discovery*, vol. 19, no. 3, pp. 293–319, 2009.

[13] U. Lamping and E. Warnicke, *Wireshark Users Guide*, Wireshark Foundation, 2008.

[14] Y. D. Lin, Y. T. Chiang, Y. S. Wu, and Y. C. Lai, "Automatic analysis and classification of obfuscated bot binaries," *International Journal of Network Security*, vol. 16, no. 6, pp. 477–486, 2014.

[15] C. X. Ling, J. Huang, and H. Zhang, "AUC: A better measure than accuracy in comparing learning algorithms," in *16th Canadian Conference on Artificial Intelligence*, pp. 329–341, June 2003.

[16] C. Y. Liu, C. H. Peng, and I. C. Lin, "A survey of botnet architecture and botnet detection techniques," *International Journal of Network Security*, vol. 16, no. 2, pp. 81–89, 2014.

[17] Y. Lu, Ye Zhu, M. Itomlenskis, S. Vyaghri, and H. Fu, "MMOPRG bot detection based on traffic analysis," *International Journal of Electronics and Information Engineering*, vol. 2, no. 1, pp. 18–26, 2015.

[18] M. Mahmoud, M. Nir, and A. Matrawy, "A survey on botnet architectures, detection and defences," *International Journal of Network Security*, vol. 17, no. 3, pp. 272–289, 2015.

[19] M. M. Masud, J. Gao, L. Khan, and B. Thuraisingham, "A multi-partition multi-chunk ensemble technique to classify concept-drifting data streams," in *13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pp. 363–375, Apr. 2009.

[20] M. Overton, "Rootkits: Risks, issues and prevention," in *The 2006 Virus Bulletin Conference*, 2006.

[21] J. R. Quinlan, *C4.5: Programs for Machine Learning*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[22] B. Rahbarinia, R. Perdisci, A. Lanzi, and K. Li, "Peerrush: Mining for unwanted P2P traffic," *Journal of Information Security and Applications*, vol. 19, no. 3, pp. 194–208, July 2014.

[23] R. S. Roshna and V. Ewards, "Botnet detection using adaptive neuro fuzzy inference system," *International Journal of Engineering Research and Applications*, vol. 3, no. 2, pp. 1440–1445, 2013.

[24] C. A. Schiller, J. Binkley, D. Harley, G. Evron, T. Bradley, C. Willems, and M. Cross, *Botnets the Killer Web APP*, United States: Syngress Publishing Inc., 2007.

[25] G. Sinclair, C. Nunnery, B. Byung, and H. Kang, "The waledac protocol: The how and why," in *Proceedings of 4th International Conference on Malicious and Unwanted Software (MALWARE'09)*, pp. 69–77, Feb. 2010.

[26] S. Stover, D. Dittrich, J. Hernandez, and S. Dietrich, "Analysis of the storm and nugache trojans: P2P is here," *USENIX*, vol. 32, no. 6, pp. 18–27, 2007.

[27] K. Thomas and D. Nicol, "The koobface botnet and the rise of social malware," in *5th International Conference On Malicious and Unwanted Software (Malware'10)*, 2010. (https://www.ideals.illinois.edu/bitstream/handle/2142/16598/malware2010.pdf?sequence=2)

[28] K. Wang, C. Y. Huang, S. J. Lin, and Y. D. Lin, "A fuzzy pattern-based filtering algorithm for botnet detection," *Computer Networks*, vol. 55, no. 15, pp. 3275–3286, 2011.

Pijush Barthakur received the Master of Computer Application (MCA) degree from Dibrugarh University, India in 2001. Currently, he is working as Associate Professor at Department of Computer Applications, Sikkim Manipal Institute of Technology, Sikkim, India. He is also pursuing his doctoral degree in Sikkim Manipal University. His research interests are in the area of Network Security. He served as a member of Technical Program Committee at 5th International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Beijing, China, 2013.

Manoj Dahal received his Ph.D degree on Networking from Tezpur University, India in 2008 for the thesis on Addressing Transport Layer Congestion Control Issues. Currently, he is working at Novell, India and his professional work mostly lie on File Access protocol areas. He is also associated with research on Detection of Botnets using Machine Learning Techniques at Sikkim Manipal Institute of Technology, Sikkim, India. He has around 15 years of 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. He also worked as a Professor for a short period in the Department of Computer Science and Engineering, at

Sikkim Manipal Institute of Technology, Sikkim. He also worked with Nokia (via Satyam) on routing devices and National Informatics Centre on e-Governance Projects in India.

M. K. Ghose is currently the Dean (Academics), SMIT and Professor, Department of Computer Science & Engineering at Sikkim Manipal Institute of Technology, Majitar, Sikkim, India. Formerly he was Dean (R&D) since June, 2006. During June 2008 to June 2010, he had also carried out additional responsibilities of Head, SMU-IT. Prior to this, he 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. He was also associated with Regional Engg. College (NIT), Silchar (1979-1981) as Teaching Asst. and Assam Central University, Silchar as COE and HOD of Computer Science Department (1997-2000). His areas of research interest are Data Mining, Simulation & Modelling, Network, Sensor Network, Information Security, Optimization & Genetic Algorithm, Digital Image processing, Remote Sensing & GIS and Software Engineering. He chaired a number of national/international conference sessions. He has conducted quite a number of Seminars, Workshop and Training programmes in the above areas and published 126 technical papers in various national and international journals in addition to presentation/publication in several international/national conferences. 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. Dr. Ghose is having 8 sponsored projects worth of 1 crore (INR). Dr Ghose also served as technical consultant to various reputed organizations like IIT Chennai, IIT Kharagpur, WRI, Tricy, SCIMST, KELTRON, HLL, Trivandrum. Dr. Ghose can also be reached at mkghosesmu.edu,in; headcse.smit@gmail.com.