Journal of Theoretical and Applied Information Technology
20th September 2014. Vol. 67 No.2
© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

FLOW BASED ANALYSIS TO IDENTIFY BOTNET INFECTED SYSTEMS

[1]R.KANNAN, [2]A.V.RAMANI

[1]Associate Professor in Computer Science, SRMV CAS, Coimbatore-20, Tamilnadu, India.

[2]Associate Professor in Computer Science, SRMV CAS, Coimbatore-20, Tamilnadu, India.

E-mail: [1]dhiyakanna@gmail.com [2]avvramani@yahoo.com

`

ABSTRACT

Botnet most widespread and occurs commonly in today's cyber-attacks, resulting in serious threats to our network assets and organization's properties hence there is a high need to detect and prevent the adverse effects of bots. Botnets are collections of compromised computers (Bots) which are remotely controlled by its originator (Bot-Master) under a common Command-and-Control (C&C) infrastructure. This paper focuses on classifying the bots and the regular hosts in the network through the classification based on their behavior. The goal is to develop a live version of the botnet detection system which identifies a botnet activity in a network, based on traffic behavior analysis and flow intervals which does not depend on packet pay load i.e., they can work on encrypted network communication protocol. The approach is to classify packets based on source IP, destination IP, number of packet, etc., using decision tree which is a classification technique in machine learning. The attribute selection is mainly based on packet attribute and does not consider the data part. The feasibility of the approach is to detect botnet activity without having seen a complete network flow by classifying behavior based on time intervals.

Keywords: *Botnet, Machine learning, Malicious, Intrusion, Network flow.*

1. INTRODUCTION

The term botnet is used to define networks of infected end-hosts, called bots or zombies (a bot is defined as a set of small scripts used to perform automated function), which are under the control of a human operator commonly known as a bot-master (a person or a group of person which controls a remote bot). A collection of bots, when controlled by a single command and control (C& C) infrastructure, form what is called a botnet. The main difference between Botnet and other kind of malwares is the existence of Command-and-Control (C&C) infrastructure. Bots interact over legitimate communication channels. Internet Relay Chat (IRC) used to be the most prevalent communication scheme among traditional botnets until the early 2000s, HTTP is also used because web traffic is generally allowed in most networks. After infection, the bot will locate and connect with an IRC server. The bot master will use established IRC command and control (C&C) channels to communicate and control the bots.

The centralized C&C mechanism of such Botnet has made them vulnerable to being detected and disabled. Therefore, new generation of Botnet which can hide their C&C communication have emerged, Peer-to-Peer (P2P) based Botnets. The P2P Botnets do not suffer from a single point of failure, because they do not have centralized C&C servers. This architecture is very difficult to locate. Among all the form of malware, botnets in particular have recent distinguished themselves as the primary platform on cyber criminals create global cooperative networks to support on-going growth of criminal attacks and activities such as DDos, spam, phishing and information theft.

According to the Command-and-Control(C&C) channel, we categorized Botnet topologies into two different models, the Centralized model and the Decentralized model.

In centralized model, Bot-Master chooses a host to be the central point (C&C) server of all the Bots. The C&C server runs certain network services such as IRC or HTTP. Since all connections happen through the C&C server,

Journal of Theoretical and Applied Information Technology
20th September 2014. Vol. 67 No.2
© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

therefore, the C&C is a critical point in this model. Fig 1 shows the basic communication architecture for a Centralized model. Botnet based on IRC: The IRC is a form of real-time Internet text messaging or synchronous conferencing. The protocol is based on the Client-Server model, which can be used on many computers in distributed networks.

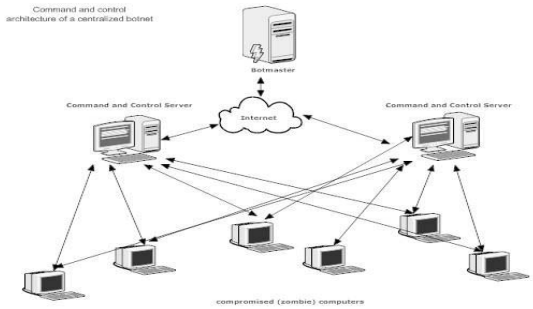


Fig. 1: Centralized Model

Botnet based on HTTP: The HTTP protocol is another popular protocol used by Botnets. Since IRC protocol within Botnets became well-known, more internet security researchers gave attention to monitoring IRC traffic to detect Botnet.

Due to main disadvantage of Centralized model attackers started to build alternative Botnet communication system that is much harder to discover and to destroy. Hence, they decided to find a model in which the communication system does not completely depending on only some selected servers and even discovering and destroying a number of Bots. As a result, attackers exploit the idea of Peer-to-Peer (P2P) communication as a Command-and-Control(C&C) pattern which is more resilient to failure in the network. In the P2P model, as shown in Fig 2, there is no Centralized point for communication. Each Bot keeps some connections to the other bots of the botnet. A new Bot must know some addresses of the Botnet to connect there. If Bots in the Botnet are taken offline, the Botnet can still continue to operate under the control of Bot-Master.

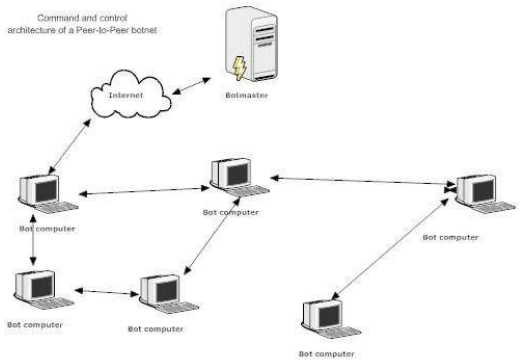


Fig. 2: Decentralized Model

Some P2P botnets operates in completely decentralized or decentralized to some extend. Those Botnets that are completely decentralized allow a Botmaster to inject a command into any Bots, and have it either be broadcasted to a specified node. Since P2P botnets usually allow commands to be injected at any node in the network, the authentication of commands become essential to prevent other nodes from injecting incorrect commands. Hence P2P have their own limitations rooted mainly in the higher latency underlying C&C transmission and the impact of such relation leads to bot synchronization.

The rest of the paper is structured as follows. Section 2 describes some related work of botnet detection, section 3 shows the system overview and how to implement using various tools and technique, in section 4 analysis of the work is done and the results are compared to various time windows and conclusion are drawn in the last session along with the future work.

2. RELATED WORK

A significant amount of literature has made on botnet detection, and the flow based analysis is emerged over the last decade.

BotHunter [2] is a system for botnet detection which correlates alarms from the Snort intrusion detection system with bot activities. Specifically, BotHunter exploits the fact that all bots share a common set of underlying actions as part of their lifecycle: scanning, infection, binary download, C&C and outbound scanning. BotHunter monitors a network and captures activity related to port scanning, outbound scanning and performs some payload analysis and malware activity detection based on Snort rules, and then uses a correlation engine to generate a score for the probability that a bot has infected the network. Like many behavior

© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

correlation techniques, BotHunter works best when a bot has gone through all phases of its lifecycle, from initial exploit to outbound scan. BotHunter is also vulnerable to encrypted command and control channels that cannot be detected using payload analysis. Payload based analysis need to phrase large amount of data and generally slow. User privacy is also disrupted when payload is inspected.

BotMiner [4] relies on the group behavior of individual bots within a botnet for its detection. It exploits the underlying uniformity of behavior of botnets and detects them by attempting to observe and cluster similar behavior being performed simultaneously on multiple machines on a network. BotMiner performs 'C-plane' clustering to first group network traffic behaviors which share similarities. Flows with known safe signatures (such as for some popular protocols) are filtered out of their list to improve performance. Once similar flows have been identified, BotMiner uses a second 'A-Plane' clustering technique which groups flows by the type of activities they represent using anomaly detection via Snort. By examining both the A-Plane and C-Plane, BotMiner correlates hosts which exhibit both similar network characteristics as well as malicious activity and in doing so identify the presence of a botnet as well as members of the network. Experimentally, BotMiner was able to achieve detection accuracies of 99% on several popular bot variants with a false positive rate around 1%.

Similar work for detecting botnet based on network behavior is given in [5]. It first eliminates the traffic that's unlikely to be the part of a botnet, and remaining into another group which are likely to be the part of a botnet. From that the activity of the botnet is identified by correlating the likely traffic which has common communication pattern.

Wurzinger et al. (2009) proposed an approach for detecting individual hosts in a network that are members of a botnet by observing and correlating commands and responses in network traces. It first identifies a responds in network flow and then analyze the traffic to find the specific command. From the response and the commands detection model are built. The detection model is evaluated using 18 bots.

3. SYSTEM OVERVIEW AND IMPLEMENTATION

Two prerequisites for the analysis are data collection (i.e. identifying and collecting data of interest) and tool acquisition and selection (i.e. identifying and deploying data mining techniques). The acquired data requires a pre-processing phase to move it into the form necessary for decision tree algorithms. Once the data is processed for various time windows, decision trees can be trained using the processed data and tools. Executing and analyzing the result of this data is an important next step to understand the resulting model and its rule sets.

Procedure to classify normal traffic from malicious traffic and to implementing decision trees can require some network data for that ISOT botnet dataset is used.

Implementing decision tree requires various tools, feature extraction tools used during the data pre-processing phase For this wire shark tool is used to perform feature extraction from pcap files. Data mining analysis tool: Weka is used for this purpose. Of all the open-source tools, Weka has been described as "perhaps the best-known open source machine learning and data mining environment".

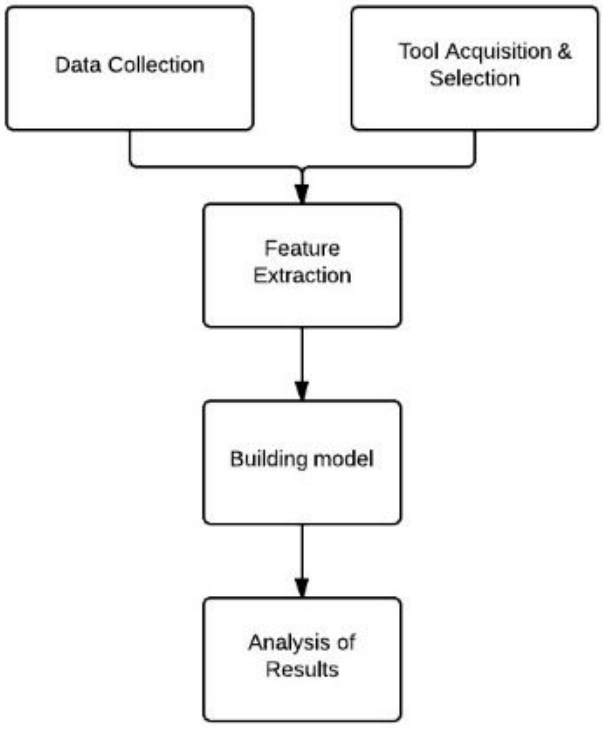


Fig. 3: Botnet Detection Framework

Journal of Theoretical and Applied Information Technology
20th September 2014. Vol. 67 No.2
© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A) Traffic Analysis:

A network flow between two hosts may run for a few seconds to several days. Obtaining real world dataset of botnet malicious activity is very difficult. Many publically available dataset may not reflect real world usages. In order to evaluate the system, a set of network traces which contains malicious and non-malicious traffic is generated.

The network traffic is collected from the ISOT Botnet dataset. The collected packet will be in unreadable format. Using Wire shark tool, which is a network protocol analyzer, separate the traffic based on user conditions like source IP, destination IP, and payload length and so on.

Real network traffic data can also be generated using sensors. Sensors will monitor the complete network exchanges. Fig 4 and 5 show the raw network packets and filtered packets respectively.

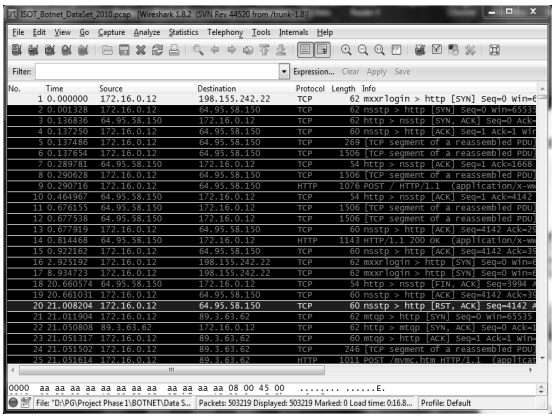
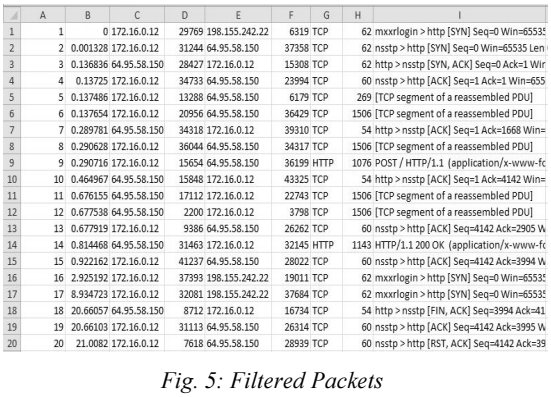


Fig. 4: Raw Network Packets



	A	B	C	D	E	F	G	H	I
1	1	0	172.16.0.12	29769	198.155.242.22	6319	TCP	62	mxxrlogin > http [SYN] Seq=0 Win=65535
2	2	0.001328	172.16.0.12	31244	64.95.58.150	37358	TCP	62	nsstp > http [SYN] Seq=0 Win=65535 Len
3	3	0.136836	64.95.58.150	28427	172.16.0.12	15308	TCP	62	http > nsstp [SYN, ACK] Seq=0 Ack=1 Wir
4	4	0.13725	172.16.0.12	34733	64.95.58.150	23994	TCP	60	nsstp > http [ACK] Seq=1 Ack=1 Win=655
5	5	0.137486	172.16.0.12	13288	64.95.58.150	6179	TCP	269	[TCP segment of a reassembled PDU]
6	6	0.137654	172.16.0.12	20956	64.95.58.150	36429	TCP	1506	[TCP segment of a reassembled PDU]
7	7	0.289781	64.95.58.150	34318	172.16.0.12	39310	TCP	54	http > nsstp [ACK] Seq=1 Ack=1668 Win=
8	8	0.290628	172.16.0.12	36044	64.95.58.150	34317	TCP	1506	[TCP segment of a reassembled PDU]
9	9	0.290716	172.16.0.12	15654	64.95.58.150	36199	HTTP	1076	POST / HTTP/1.1 (application/x-www-fo
10	10	0.464967	64.95.58.150	15848	172.16.0.12	43325	TCP	54	http > nsstp [ACK] Seq=1 Ack=4142 Win=
11	11	0.676155	64.95.58.150	17112	172.16.0.12	22743	TCP	1506	[TCP segment of a reassembled PDU]
12	12	0.677538	64.95.58.150	2200	172.16.0.12	3798	TCP	1506	[TCP segment of a reassembled PDU]
13	13	0.677919	172.16.0.12	9386	64.95.58.150	26262	TCP	60	nsstp > http [ACK] Seq=4142 Ack=2905 W
14	14	0.814468	64.95.58.150	31463	172.16.0.12	32145	HTTP	1143	HTTP/1.1 200 OK (application/x-www-fo
15	15	0.922162	172.16.0.12	41237	64.95.58.150	28022	TCP	60	nsstp > http [ACK] Seq=4142 Ack=3994 W
16	16	2.925192	172.16.0.12	37393	198.155.242.22	19011	TCP	62	mxxrlogin > http [SYN] Seq=0 Win=65535
17	17	8.934723	172.16.0.12	32081	198.155.242.22	37684	TCP	62	mxxrlogin > http [SYN] Seq=0 Win=65535
18	18	20.66057	64.95.58.150	8712	172.16.0.12	16734	TCP	54	http > nsstp [FIN, ACK] Seq=3994 Ack=41
19	19	20.66103	172.16.0.12	31113	64.95.58.150	26314	TCP	60	nsstp > http [ACK] Seq=4142 Ack=3995 W
20	20	21.0082	172.16.0.12	7618	64.95.58.150	28939	TCP	60	nsstp > http [RST, ACK] Seq=4142 Ack=39

Fig. 5: Filtered Packets

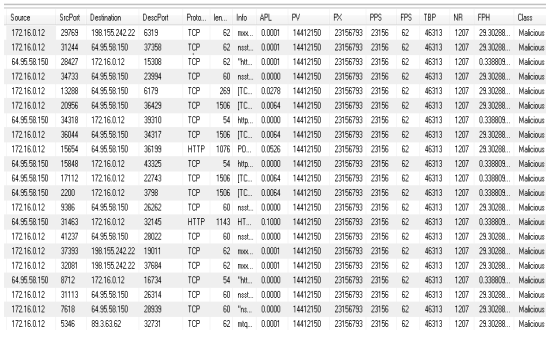
B) Feature Extraction:

The network characteristic of the flow is calculated for different time window T and also two observations are made on the time window. First, if a time window is too small, we may fail to capture unique traffic characteristics that only become apparent over a longer period of time. If a time window is too large, the decision cannot be made until the window size is met. Thus the selection of time window size will be based on a compromise between detection accuracy and speed.

C) Attribute Selection:

An attribute is some characteristic of a flow or a collection of flow in a given time window T which may be represented as a numeric or nominal value.

An attribute is some characteristic of a flow or a collection of flows in a given time window T which may be represented as a numeric or nominal value. The set of 13 attributes like source IP, source port, destination IP, destination port and protocol field can be directly derived from the network packet and Average payload length for time interval, Variance of payload packet length for time interval, Number of packet exchanged for time interval, Number of packet exchanged per second in time interval, Size of the first packet in the flow, Average time between packets in time interval, Number of reconnect for the flow, Number of flows from this address over the total number of flows generated per hour, rest of the attributes are calculated with the help of time and length fields for specific time interval. After extracting the network flow attributes from ISOT botnet dataset each packet need to labeled as malicious or non-malicious as per the description given in the document. The document describes about the system which generates the malicious or non-malicious packets.



Source	SrcPort	Destination	DescPort	Proto...	len...	Info	APL	PV	PX	PPS	FPS	TBP	NR	FPH	Class
172.16.0.12	29769	198.155.242.22	6319	TCP	62	mxx...	0.0001	14412150	23156793	23156	62	46313	1207	29.302088...	Malicious
172.16.0.12	31244	64.95.58.150	37358	TCP	62	nsst...	0.0001	14412150	23156793	23156	62	46313	1207	29.302088...	Malicious
64.95.58.150	28427	172.16.0.12	15308	TCP	62	*ht...	0.0001	14412150	23156793	23156	62	46313	1207	0.338809...	Malicious
172.16.0.12	34733	64.95.58.150	23994	TCP	60	nsst...	0.0000	14412150	23156793	23156	62	46313	1207	29.302088...	Malicious
172.16.0.12	13288	64.95.58.150	6179	TCP	269	[TC...	0.0278	14412150	23156793	23156	62	46313	1207	29.302088...	Malicious
172.16.0.12	20956	64.95.58.150	36429	TCP	1506	[TC...	0.0064	14412150	23156793	23156	62	46313	1207	29.302088...	Malicious
64.95.58.150	34318	172.16.0.12	39310	TCP	54	http...	0.0000	14412150	23156793	23156	62	46313	1207	0.338809...	Malicious
172.16.0.12	36044	64.95.58.150	34317	TCP	1506	[TC...	0.0064	14412150	23156793	23156	62	46313	1207	29.302088...	Malicious
172.16.0.12	15654	64.95.58.150	36199	HTTP	1076	PO...	0.0526	14412150	23156793	23156	62	46313	1207	29.302088...	Malicious
64.95.58.150	15848	172.16.0.12	43325	TCP	54	http...	0.0000	14412150	23156793	23156	62	46313	1207	0.338809...	Malicious
64.95.58.150	17112	172.16.0.12	22743	TCP	1506	[TC...	0.0064	14412150	23156793	23156	62	46313	1207	0.338809...	Malicious
64.95.58.150	2200	172.16.0.12	3798	TCP	1506	[TC...	0.0064	14412150	23156793	23156	62	46313	1207	0.338809...	Malicious
172.16.0.12	9386	64.95.58.150	26262	TCP	60	nsst...	0.0000	14412150	23156793	23156	62	46313	1207	29.302088...	Malicious
64.95.58.150	31463	172.16.0.12	32145	HTTP	1143	HT...	0.1000	14412150	23156793	23156	62	46313	1207	0.338809...	Malicious
172.16.0.12	41237	64.95.58.150	28022	TCP	60	nsst...	0.0000	14412150	23156793	23156	62	46313	1207	29.302088...	Malicious
172.16.0.12	37393	198.155.242.22	19011	TCP	62	mxx...	0.0001	14412150	23156793	23156	62	46313	1207	29.302088...	Malicious
172.16.0.12	32081	198.155.242.22	37684	TCP	62	mxx...	0.0001	14412150	23156793	23156	62	46313	1207	29.302088...	Malicious
64.95.58.150	8712	172.16.0.12	16734	TCP	54	*ht...	0.0000	14412150	23156793	23156	62	46313	1207	0.338809...	Malicious
172.16.0.12	31113	64.95.58.150	26314	TCP	60	nsst...	0.0000	14412150	23156793	23156	62	46313	1207	29.302088...	Malicious
172.16.0.12	7618	64.95.58.150	28939	TCP	60	*ns...	0.0000	14412150	23156793	23156	62	46313	1207	29.302088...	Malicious
172.16.0.12	5346	89.3.63.62	32731	TCP	62	ntq...	0.0001	14412150	23156793	23156	62	46313	1207	29.302088...	Malicious

Fig. 6: Network Flow Attributes

D) Classification Model:

After extracting the network flow attributes the profile is given to the classifier for building the

© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

model. In order to support real time detection goals while at the same time exhibiting high detection accuracy a popular machine learning approach decision tree classifier is used. The total numbers of flows used for classification process are

Table: 1

Class Info	Unique Flows
Malicious	3, 71,385(73.81%)
Non-Malicious	1, 31,834(26.19%)
Total	5, 03,219(100%)

E) Decision Tree Learning:

The decision tree is very powerful and popular data mining algorithm for decision-making and classification problems. Automated decision tree classifiers uses a decision tree as a predictive model, taking in a series of observations and generating a value for those given inputs. Leaf nodes in a decision tree represent a class or a node linked to two or more sub trees called a test node. At a test node, some outcome is computed based on the attribute values of an instance, where each possible outcome leads to one of the sub trees rooted at the node. To implement this framework the popular weka machine learning framework and libraries are used.

4. ANALYSIS OF RESULT

After the decision tree is created, the resulting model must be analyzed. The accuracy of the model and the insights gained from the resulting tree are important to consider. Model accuracy is usually straightforward to measure techniques such as k-fold cross validation can test model accuracy in a meaning way.

Cross validation calculates the accuracy of the model by separating the data into two different populations: a training set and a testing set. The decision tree model is created from the training set and its accuracy is measured based on how well it classifies the testing set. This testing process is continued k times to complete the k-fold cross validation procedure.

As an example of k-fold cross validation, consider that k=2, meaning that we have 2-fold cross-validation. The 2-fold cross validation starts its first iteration. The entire dataset is divided in two, with half of the elements belonging to a training set and half belonging to a testing set. A decision tree is created from the training set and it attempts to correctly identify elements from the testing set, a set of data the tree has not seen up to and this accuracy is recorded. The first iteration of the validation is complete. With the second iteration, the training set and the testing set are switched. A decision tree is built from the new training set (the testing set in iteration 1) and attempts to correctly classify elements in the new testing set (the training set in iteration 1). The accuracy of this classification is captured and the second iteration is completed. To assess the accuracy of the entire decision tree model, an average of the first accuracy score and the second accuracy score is presented.

The following table demonstrates the accuracy of the classification process being used in detecting the botnets.

Table: 2

Time Window Size(Sec)	Correctly Classified	Incorrectly Classified
10	4, 97,393 (98.84%)	5,826 (1.16%)
300	5, 03,216 (99.994%)	3 (0.006%)
1000	5, 03,219 (100%)	0 (0%)

The k-fold cross validation accuracy measure provides a meaningful estimation of the overall accuracy of the classifier. When model performance is poor, this may imply that there are no meaningful patterns in the feature data extracted for the experiment or that a different classification algorithm should be used.

The model itself often provides insight into the problem of botnet detection. For example, a decision tree presents a set of rules that differentiate between malicious and benign traffic. The rules may highlight specific ports, IP addresses, or other features extracted from data of interest. These insights can help an intrusion detection team work towards follow-on action based on the model result.

The constructed model using decision tree classifier produced very high (around 100%). These results indicate that there are indeed unique characteristics of the evaluation botnets when compared to everyday network traffic.

© 2005 - 2014 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

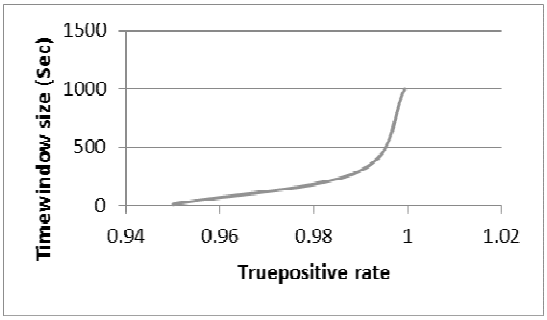


Fig. 7: Effect of time window size

The true positive rate for different time window (10, 300 1nd 1000 Sec) are also calculated which increases sharply when time window T is more than 300 Sec and saturated beyond 1000 Sec. from this the optimal time window can be calculated for monitoring the network traffic.

5. CONCLUSION

The result showed that using decision tree classifier, the system able to successfully detect botnet activity with high accuracy by simply observing small portions of a full network flow, allowing us to detect and respond to botnet activity in real time. The existing systems to detect botnets are less efficient as they tend to identify only the regular botnets. This disadvantage is overcome by forming new rules to detect the botnets in the proposed system, hence improving the efficiency and. The system is modeled in such a way that the bots can be detected even if the communication is encrypted. The Experimental evaluation under various settings shows that our detector is able to achieve a true positive rate of over 95%. While both the performance and accuracy of our classifiers are satisfactory for real time detection, it must be noted that the system may be sensitive to new behaviors from bots implementing highly varied protocols. To prepare for such risk, a technique for web preparing and nonstop refinement of the classifiers must in case de actualized.

REFERENCES

[1] David Zhao, IssaTraore, BassamSayed, Wei Lu, SherifSaad, Ali Ghorbani and Dan Garant. "Botnet detection based on traffic behavior analysis and flow intervals", Computers and Security in year 2013.

[2] GuofeiGu, Phillip Porras, VinodYegneswaran, Martin Fong, Wenke Lee "BotHunter: detecting malware infection through IDS-driven dialog correlation". In: Proceedings of the 16th USENIX security symposium, Boston, MA, USA 2007.p.167e82.

[3] Gu G, Zhang J, Lee W. "BotSniffer: detecting botnet command and control channels in network traffic". In: Proceedings of the 15th annual network and distributed system security symposium 2008.

[4] Gu G, Perdisci R, Zhang J, Lee W. "BotMiner: clustering analysis of network traffic for protocol- and structure-independent botnet detection". In: Proceedings of the 17th USENIX security symposium, San Jose, CA, USA 2008.

[5] W. Timothy Strayer, David Lapsely, Robert Walsh, Carl Livadas. "Botnet Detection Based on Network Behavior" In: Advances in *Information Security, Springer, Volume 36, 2008*, pp 1-24.

[6] Feily M, Shahrestani A and Ramadass S, "A survey of botnet and botnet detection", In: Proceedings of the third international conference on emerging security information, systems and technologies. IEEE Computer Society, 2009. p. 268e73.

[7] Julian B Grizzard, Vi kram Sharma, Chris Nunnery, Brent ByungHoon Kang, David Dagon. "Peer-to-peer botnets: overview and case study", In: Proceedings of the first workshop on hot topics in understanding botnet (HotBots'07), Cambridge, MA, April 2007. Berkeley: USENIX Association, 2007.

[8] ISOT Botnet Dataset. URL: http://www.isot.ece.uvic.ca

[9] Wurzinger P, Bilge L, Holz T, Goebel J, Kruegel C, Kirda E. Automatically generating models for botnet detection. In: Proceedings of the 14th European conference on research in computer security (ESORICS 2009). Lecture Notes in Computer Science, vol. 5789. Springer Verlag; 2009. p. 232e49.

[10] The Honeynet Project. French Chapter [Online] http://www.honeynet.org/chapters/france.