2011 Ninth Annual International Conference on Privacy, Security and Trust

Detecting P2P Botnets through Network Behavior Analysis and Machine Learning

Sherif Saad
Electrical and Computer Engineering
University Of Victoria
Email: shsaad@ece.uvic.ca

Issa Traore
Electrical and Computer Engineering
University Of Victoria
Email: itraore@ece.uvic.ca

Ali Ghorbani
University of New Brunswick
Email: ghorbani@unb.ca

Bassam Sayed
Electrical and Computer Engineering
University Of Victoria
Email: bassam@ece.uvic.ca

David Zhao
Electrical and Computer Engineering
University Of Victoria
Email: davidzhao@ieee.org

Wei Lu
Keene State College
Email: wlu@keene.edu

John Felix
University of New Brunswick
Email: johnfelixc@gmail.com

Payman Hakimian
RCMP Atlantic Region
Integrated Technological Crime Unit Fredericton
Email: Payman.Hakimian@rcmp-grc.gc.ca

***Abstract*—Botnets have become one of the major threats on the Internet for serving as a vector for carrying attacks against organizations and committing cybercrimes. They are used to generate spam, carry out DDOS attacks and click-fraud, and steal sensitive information. In this paper, we propose a new approach for characterizing and detecting botnets using network traffic behaviors. Our approach focuses on detecting the bots before they launch their attack. We focus in this paper on detecting P2P bots, which represent the newest and most challenging types of botnets currently available. We study the ability of five different commonly used machine learning techniques to meet online botnet detection requirements, namely adaptability, novelty detection, and early detection. The results of our experimental evaluation based on existing datasets show that it is possible to detect effectively botnets during the botnet Command-and-Control (C&C) phase and before they launch their attacks using traffic behaviors only. However, none of the studied techniques can address all the above requirements at once.**

I. Introduction

A botnet is a collection of computers connected to the Internet which have been compromised and are being controlled remotely by an intruder via malicious software called bots. Botnets are created for many different reasons including the need to conduct a distributed denial of service (DDOS), spread spam, conduct click- fraud scams, steal personal user information (e.g. credit card numbers, social security numbers), or take advantage of the powerful computational resources offered by the bots to carry some distributed computing tasks.

A key aspect of any botnet is the communication architecture. Bots interact over legitimate communication channels. Internet Relay Chat (IRC) has been for a while the most prevalent communication scheme among traditional botnets. After infection, the bot will locate and connect with an IRC server. The bot master (the criminal controlling the botnet) will use established IRC command and control (C&C) channels to communicate and control the bots. The bot master will try to keep the bots under control as long as possible. From time to time the bots will connect to the bot master to get new instructions and update their behavior.

Besides IRC, HTTP-based schemes have also gained in popularity in the last several years. However, both IRC and HTTP-based botnets are vulnerable because they are based on highly centralized architectures; one can disrupt the entire botnet by simply shutting down the IRC or HTTP server. Currently the new trend in botnet communication is toward Peer-to-Peer architectures. Unlike IRC and HTTP-based botnets, in P2P-based botnets, there is no central point to shutdown the botnet. Furthermore the bot master can inject commands into any part of the P2P botnet. Because of its elusive nature, it has so far been very difficult to estimate the size of a typical P2P-based botnet. To make things worse, encrypted botnets such as Nugache have been able to evade most of the available botnet detection techniques.

Leonard et al. divided the botnet lifecycle into four phases, namely, formation, C&C, attack, and post-attack phases [1]. In the formation phase the bot master spreads bots by infecting other machines on the Internet so that they become members of the botnet. Then the bots (i.e. infected machines) which are enslaved will receive on a regular basis instructions from the bot master during C&C phase (which follows the formation phase). During the attack phase, the bots will carry malicious activities based on the received instructions. After the attack phase some bots might be detected and removed, while the botmaster will try (from time to time) to probe the botnet to get infromation about active bots and plan for new formation.

To this date, most of the works on botnet detection have aimed at detecting bots during the formation or the attack phase [2]. Detecting bots during the formation or the attack

978-1-4577-0584-7/11/$26.00©2011 IEEE

phase is very similar to how intrusion detection systems (IDSs) operate. The detection of bots in this case is more likely to focus on the detection of underlying malicious activities such as the methods used to compromise machines. Other approaches detect bots by setting up a honeynet and trying to study bots behaviors [3].

In this work, we focus on detecting bots during the C&C phase. We study in particular the detection of P2P botnets, although we believe that our proposed approach can be extended to centralized botnets. We investigate network traffic behaviors and, extract and analyze a set of traffic behavior characteristics using Machine Learning (ML) techniques. We apply and compare the performance of five different ML techniques, namely, Support vector Machine (SVM), Artificial Neural Network (ANN), Nearest Neighbors Classifier (NNC), Gaussian-Based Classifier (GBC), and Naives Bayes Classifier (NBC).

The detection of botnets using network behaviors has several advantages. First, the detection will not be limited to formation or attack phases only. In general, it is possible to detect bots during any phase of their lifecycle. The second advantage is that detecting bots will be less expensive compared to other approaches that implement deep-payload-analysis or attempt to capture and study live bots using honeynets. Finally, detecting bots during C&C phase allows detecting bots that were missed during the formation phase and before they launch their attack and cause some damages.

The remaining of this paper is organized as follows. Section II provides an overview of previous approaches on detecting botnets. In Section III, we present our approach for botnet detection based on network traffic behaviors and discuss the requirements for online botnet detection. In Section IV, we evaluate our approach using existing experimental datasets and by comparing the performance obtained with the different ML techniques under consideration. Finally, we make some concluding remarks and discuss future work in Section V.

II. Related Work

Although a significant amount of literature has been produced on centralized botnet detection, the interest of the research community is only slowly shifting toward P2P botnet, since this is still an emerging technology. Most of the literature in this field focuses on analyzing P2P botnet, and characterizing their structure, organization and operation [4].

Gu et al. presents a general botnet detection framework referred to as BotMiner that is independent of the botnet C&C protocol, structure, and infection model [5]. Likewise the proposed framework targets both centralized IRC and P2P botnets. The working assumption of BotMiner is that bots are coordinated malware that exhibit similar communication patterns and similar malicious behaviors. Hosts with similar communication patterns and those performing similar malicious activities are clustered from captured network flows in the so-called A-plane (activity traffic) and C-plane (C&C communication traffic), respectively. And then, by performing cross-plane correlation across A-plane and C-place clusters, hosts that exhibit both kinds of behaviors are flagged as bots. The proposed approach was evaluated using real network traces. The results show that BotMiner has a high detection rate and generates a low number of false positives and false negatives. Furthermore, the analysis of network traffic employs a reasonable amount of resources and time, making detection relatively efficient.

A key limitation of BotMiner is that by design it targets essentially groups of compromised machines within a monitored network. However, it is common that there is only a single compromised host belonging to the monitored network, although such host may belong to a larger botnet. Under such scenario, BotMiner may not be effective at detecting the compromised host. Another important limitation of BotMiner is the systematic classification of any host that behaves maliciously and exchanges C&C messages as a bot. Although, in general, this may be the case for IRC botnets, it is not always the case for P2P botnets. A peer may behave maliciously and still exchange normal C&C messages as simply being part of a P2P network. In this case, it may only be qualified as a bot if it exchanges covert botnet messages. In the same vein BotMiner will fail to detect bots that exchange covert C&C messages, without undertaking any malicious activity.

Giroire et al. focus their work on detecting botnet C&C communications on an end host [6]. Their working assumption is that a recruited host needs to keep in touch with the bot master to remain relevant. This is characterized by frequent communications between the bot master and the bot. Such communications exhibit some temporal regularity even though they may be spread over irregular large time periods. Such regularity is captured by designing a detector that monitors user A outgoing traffic and identifies malicious destinations visited with some temporal regularity. Malicious destinations are distinguished from normal ones by building destination IP white lists referred to as atoms by the authors. To capture the temporal regularity of visited destinations, a feature referred to as *persistence* is introduced. Detection of botnet C&C channels is carried out by identifying non-white listed destinations with high persistence using thresholding methods. Evaluation of the proposed approach using real network traces yields low false positive rate.

Strayer and colleagues introduce a botnet detection system based on network behaviors [7]. The authors focus on detecting IRC bots by studying the characteristics of IRC flows. The proposed approach can be divided into four stages. In the first stage the collected network flows are filtered to eliminate flows that are most likely not carrying bot C&C data. The filters use some prior knowledge about IRC bots commands, black/white list of Internet sites and some flows characteristics such as flow duration and packet size. In the second stage, the remaining flows are clustered into predefined network applications clusters using machine learning techniques. However, it is not clear which specific machine learning techniques are used in this stage. Flows that are clustered as chat-like applications are then passed to a correlator stage. In the correlator stage flows are clustered again into group of

flows showing similar characteristics. Then, the correlated flows are passed on to topological analysis to determine flows with common controller. Finally flows that share a common controller are investigated by human analyst to determine if they belong to a botnet or not. Hence the final botnet detection is made offline since this relies on human expertise, which can be considered as a limitation of the proposed approach.

Liu and colleagues propose a P2P botnet detection model using data-mining techniques [8]. The authors discuss the possibility of detecting hosts infected with bots by analyzing bots traffic behaviors. A set of metrics that can distinguish between Bots traffic, normal P2P traffic, Gaming traffic and general Internet traffic are defined. The evaluation of the proposed approach involved studying one P2P botnet virus known as Trojan.Peacomm, and achieved a detection rate for infected hosts between 87% and 98%. However, it is unclear how the proposed approach will perform when dealing other categories of botnets.

Noh and colleagues propose a P2P botnet detection technique using a multi-phased flow model [9]. In the proposed model, P2P botnets are identified by observing similar flows occurring between a group of hosts in the network on a regular basis. Flows with similar behaviors are grouped and a transition model of the grouped flows is constructed using a probability matrix. Finally a likelihood ratio is computed and used for bots detection. Experimental evaluation is carried by studying different P2P bots such as Storm, SpamThru and Nugache. The detection rate was between 95% and 100%.

III. Network Behaviors Analysis

A. Approach

In general, there are three categories of network traffic identification methods, including port-based analysis, protocol-based analysis and behaviors-based analysis. The first method and most trivial one is based on port analysis. The problem with port-based identification is the high false identification rates involved and the fact that today there are thousands of network applications that do not use registered TCP/UDP ports. The second method is based on packets payload analysis. Payload analysis has minimum false identification rate compared to other approaches. However, two major issues limit the use of this approach. First, it is computationally intensive and has a negative effect on the network performance. Second, it poses potentially significant threat to privacy. The third method for traffic identification is based on detecting distinctive network traffic behaviors. Traffic behavior analysis methods do not depend on the packets payload, which means that they can work with encrypted traffic. Network traffic information can usually be easily retrieved from various network devices without affecting significantly network performance or service availability.

Several studies have shown that network traffic identification can effectively distinguish between different classes of network applications, although it is important to mention that the focus of these studies was not security related [10], [11], [12]. In fact, most of these studies use sophisticated machine learning (ML) techniques and have reported accuracies over 90%. In the same vein, our proposed approach consists of looking at the process of detecting botnet C&C traffic as a special case of network traffic identification. In practice, each of the existing major botnet (for instance Storm-botnet and Zeus-Botnet [13], [14]) implements their own specific C&C architecture. Such architectures tend to exhibit distinguishing behaviors that can be captured by analyzing network traffic characteristics. Moreover, it is most likely that partially matching behaviors will occur regularly in the life of the botnet. For instance, the bot master usually uses scripts that execute automatically when specific events happen such as a new bot joining the botnet. Under such consideration, our approach consists of identifying specific traffic characteristics that can be used to distinguish between botnets traffic and other network application traffic.

B. Traffic Characteristics and Features Selection

Behaviors-based network traffic identification usually aims at studying different characteristics that occur in network traffic. For example, the packet size, the flow duration, the number of ACK and SYN packets per flow, etc. Behaviors-based identification methods use a combination of these characteristics to distinguish between different types of network traffic. In our model we use information about payload size, number of packets, duplicated packets length, and concurrent active ports to build our features set.

Our proposed features can be grouped into two categories, namely, flow-based features and host-based features. The first category involves a set of features that can be extracted from network flows. These features are used to link flows to specific class of network traffic such as P2P traffic or non-P2P traffic. The host-based features are features that occur in the communications between a host $\boldsymbol{x}$ and other hosts. The host-based features are useful to identify hosts with shared communications patterns.

In our model we define 17 features that can be extracted from network flows and host communication patterns as shown in Table I.

C. Network Behavior Classification: Challenges and Techniques

Identifying network behavior characteristics represents only half of the tasks in our proposed framework. We also need to analyze and classify effectively the network behavioral characteristics in order to isolate botnet behavior from normal network behavior.

The network traffic is a data stream, which means the signature of the traffic will most likely change over time. Moreover, bots tend to change their behaviors over time as results of commands from the bot master or new botnet version replacing the old one. This will create a concept-drift problem [15] for the detection framework. Unlike legitimate network applications, new botnets can spread over the Internet at anytime. This means that the detection model may not capture the characteristics of these new bots during the training phase.

Feature	Description	Type
SrcIP	Flow source IP address	flow
SrcPort	Flow source port	flow
DstIP	Flow destination IP address	flow
DstPort	Flow destination port	flow
Protocol	Transport layer protocol	flow
Pack length	Payload size in bytes	flow
APL	Average packet length per flow	flow
FPL	The length of the first packet in the flow	flow
TPC	The total number of packets per flow	flow
TBT	Total number of bytes per flow	flow
IOP	The ratio between the number of incoming packets over the number of outgoing packets	flow
DPL	The total number of subsets of packets of the same length over the total number of packets in the same flow	flow
PL	The total number of bytes of all the packets over the total number of packets in the same flow	flow
SPDP	The ratio between the number of source ports to the number of destination ports	host
CDA	The number of connections over the number of destination IP addresses	host
TPDA	The sum of the numbers of different transmission protocols (usually 1 or 2) used per destination IP over the total number of destination IPs	host
DASP	The number of destination IPs connected to the same open port in the monitored host over the total number of open ports in the monitored host	host

TABLE I
SELECTED NETWORK TRAFFIC FEATURES

This will create a novelty detection problem for the detection framework. In addition, processing the huge size of traffic occurring in a typical network environment is a computational intensive task.

In our opinion the design of an online botnet detection framework requires a computational model with specific characteristics, including adaptability, novelty detection, and early detection. We investigate in this work the ability of different machine learning techniques in addressing the above requirements. We selected the most common machine learning classification techniques that were used in the literature to detect botnet. We used these machine learning classification techniques to investigate the possibility of detecting botnets traffic by analyzing network traffic behaviors only. The ML techniques studied in our work are the following:

- Nearest Neighbors Classifier
- Linear Support Vector Machine
- Artificial Neural Network
- Gaussian Based Classifier
- Naive Bayes classifier

IV. EXPERIMENTAL EVALUATION

A. Selecting the Datasets

Finding the right dataset that contains both labeled malicious and non-malicious traffic is always a challenge. The challenge lies in the fact that most if not all datasets of malware network traffic come from honeynet projects or similar types of projects, which do not reflect real-world usages. In a typical honeynet configuration, a honeypot is a dedicated machine that is setup to be the victim machine and from which all the generated traffic before and after malware infection phase is considered malicious. In other words, such machine does not generate regular non-malicious traffic that typically would occur in a real world scenario. In a real world scenario, the infected machine would be a regular machine used for normal day-to-day computing tasks such as, browsing the web, checking email, chatting, etc. When such machine becomes infected, in addition to the above user-generated normal traffic, the malware might generate malicious traffic such as scanning local network for further infection or participating in some DDoS attacks. Accordingly our ultimate goal is to come up with a dataset that contains both regular non-malicious and malicious traffic. The traffic in the dataset should be intermixed as if both kinds of traffic were happening at same time from the same machines. In addition both kinds of traffic should be labeled so that we can effectively evaluate our detection model.

B. Individual Datasets

We obtained and used, in this work, two datasets containing malicious traffic from the French chapter of the honeynet project, involving the Storm botnet and the Walowdac botnet, respectively. To represent the non-malicious traffic we used a labeled dataset from the Traffic Lab at Ericsson Research in Hungary. This dataset contains over a million packets of general traffic that ranges from web browsing (HTTP) to Peer-to-Peer traffic and gaming such as Quake and World of Warcraft.

The first dataset, as illustrated in Figure 1, contains 214,799 packets coming from or going to an infected machine with internal IP address of 10.0.0.1. The trace file corresponds to the C&C and attack phase of the storm botnet as the bot master used this machine to spread spam.

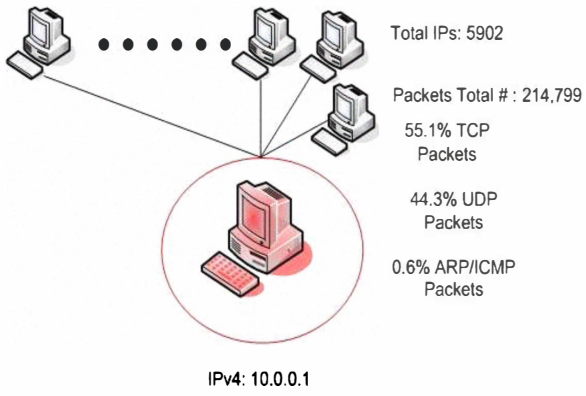

Fig. 1. Storm Botnet Trace File Summary

The second dataset, as illustrated in Figure 2, contains 157,100 packets coming from or going to an infected machine with internal IP address of 10.0.0.10. The trace file

corresponds to the attack phase of the walowdac botnet. The Walowdac botnet is well known for spreading spam: the infected machine communicated with 2,615 external IP addresses to spread spam.

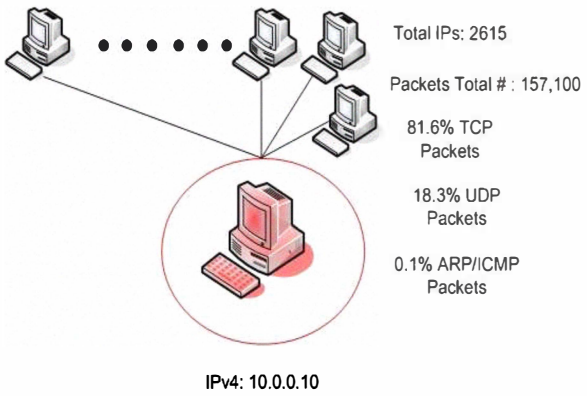

Fig. 2. Walowdac Bot Trace File Summary

As we mentioned before, the non-malicious traffic is represented by a third dataset from Ericsson Research in Hungary. The main advantage of such dataset is that it is labeled and the capturing process was conducted in a controlled environment, which means that every packet was labeled with the originating or the target process running on the test machines. We refer to this dataset as background traffic as it does provide the necessary non-malicious background traffic in our dataset. As illustrated in Figure 3, there are five internal machines with IPs 172.16.2.[2, 11, 12, 13, 14]. These five machines generated many different types of traffics which are typical for day-to-day use, by communicating with 8,525 external IPs.

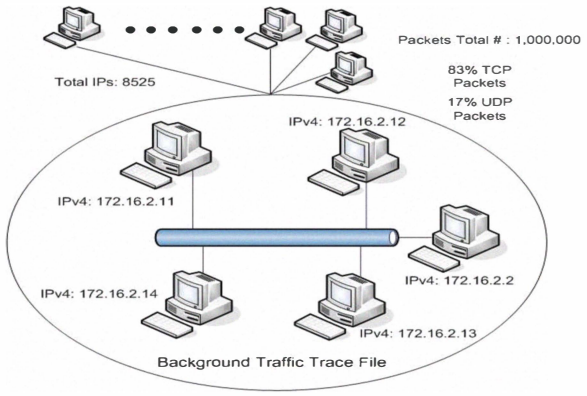

Fig. 3. Background Traffic Trace File Summary

C. Datasets Merging

In order to obtain a dataset that combines both malicious and non-malicious types of traffic we had to merge the above three individual datasets. In order to do so, we performed the following steps. Firstly, we mapped the IP addresses of the infected machines to two of the machines providing the background traffic. Secondly, we replayed all of the three trace files using TcpReplay tool on the same network interface card (eth0). At the same time we used wireshark, the ubiquitous capturing tool, to listen on eth0, and capture and save the output to a file. Figure 4 depicts the merging process.

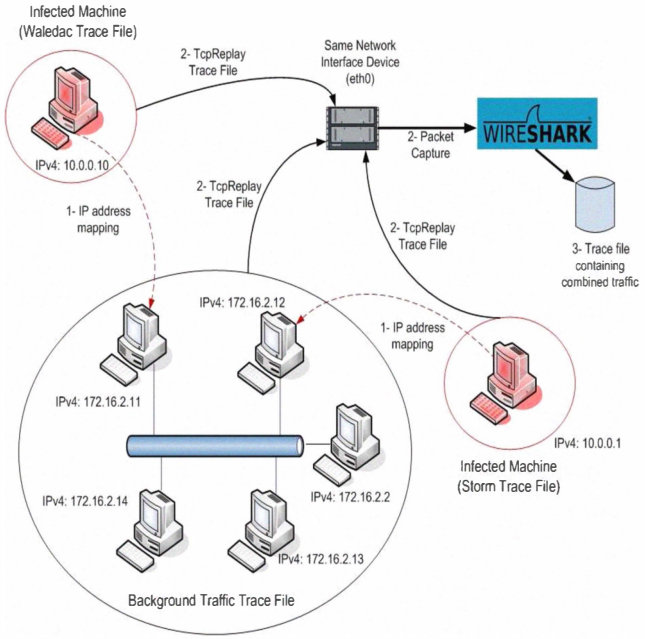

Fig. 4. Datasets Merging Process

The net result of the merging process is a trace file containing both general non-malicious background traffic and malicious traffic originating from the same machines. This would be as close as possible that we could get to the real world scenario.

D. Evaluation Results

We used Java to implement a feature extraction component that parses the network traffic dataset and extracts the feature vectors. Each feature vector represents the traffic behavior corresponding to a single flow. We extracted 129,453 feature vectors from the (merged) dataset. The feature vectors are labeled into three classes, namely, *Botnet C&C, non-P2P traffic* and *normal P2P traffic*. While obviously the Botnet C&C feature vectors represent the flows that contain P2P botnet C&C traffic, the non-P2P feature vectors describe the flows that contain general network traffic such as HTTP, FTP, SMTP and gaming traffic. The normal P2P feature vectors represent the flows that contain legitimate P2P traffic such as Bittorrent, Skype and e-Donkey.

We used a 10-fold cross-validation technique to evaluate our approach. In addition, to the features extraction component we used two machine learning packages, Weka and Java-ML, to build the five machine learning techniques being considered in our study [16], [17].

We used four metrics to evaluate each machine learning technique. These include *training speed, classification-speed, true detection rate* and *total error rate*. The training speed measures the time each machine learning technique will take

to learn the network flow traffic behaviors. The classification speed measures the time required by a machine learning technique to classify the network traffic flows into one of the three classes mentioned above. Figures 5 and 6 show the training speed and the classification speed, respectively, for each of the five machine learning techniques.

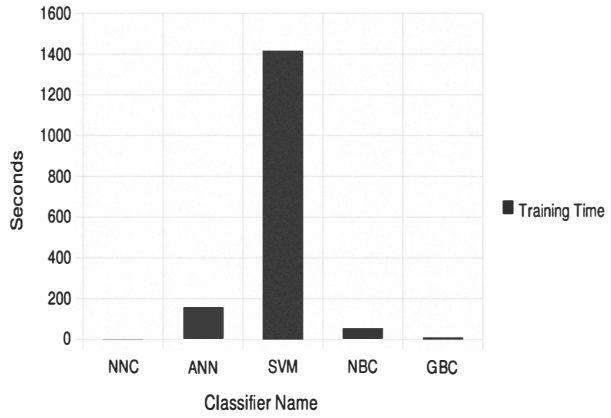

Fig. 5. Classifiers Training Time

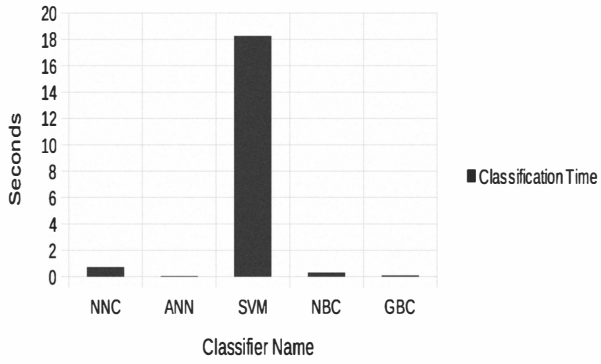

Fig. 6. Classifiers Classification Time

The true detection rate measures the accuracy (i.e. True Positives) of each machine learning technique in detecting P2P botnet C&C traffic flows. The total error rate measures the total false detection rate for each machine learning technique when it is trying to distinguish between the three different classes of network traffic in the dataset. Figure 7 and Figure 8 show the true detection rate and the total error rate, respectively, for each of the machine learning techniques.

As we can see from the figures the true detection rate of the P2P Botnet C&C is above 90% for the Support Vector Machine, Artificial Neural Network and the Nearest Neighbors Classifier and the total error rate is less than 7%. The true detection rate is also above 88% for the Gaussian Based Classifiers and the Naive Bayes classifier, but with total error rate greater than 10%. These results indicate that it is

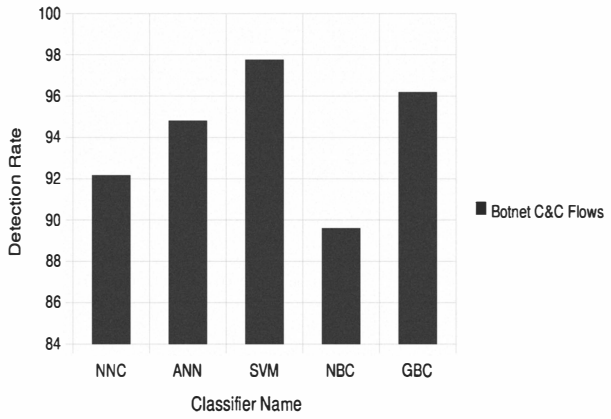

Fig. 7. Detection Rate of Botnet C&C Flows

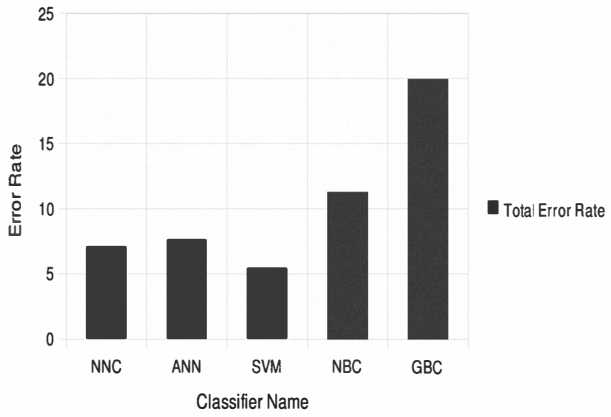

Fig. 8. Total Classification Error Rate

possible to identify P2P botnet C&C traffic by analyzing traffic behaviors only.

By comparing the five machine learning techniques based on the four metrics considered above, we can conclude that the Support Vector Machine, Artificial Neural Network and Nearest Neighbors Classifier are the top three machine learning techniques (among the five) that can be used to build a botnet detection framework.

However, none of the five classifiers can satisfy the novelty detection and the adaptability requirements of the online detection framework that were mentioned earlier. In fact, considering both the training time and classification time metrics, the SVM and the ANN are not suitable for online detection.

V. Conclusion

In this paper we proposed the characterization of network traffic behaviors to detect P2P botnet command and control (C&C) phase. The detection of botnet C&C phase is very important since it allows the detection of bots that spread under the radar through malicious email, websites, file sharing networks, ad hoc wireless network and before these bots

attack their victims or achieve their goals. We discussed the main requirements of an online botnet detection framework and investigated the power of different machine learning (ML) techniques that are commonly used in the literature in addressing these requirements. Although the performance of these techniques was promising, none of these techniques can satisfy all the requirements of an online botnet detection framework. This underscores the need to investigate new ML technique or an hybrid of existing ones that can satisfy all the requirements of online botnet detection as outlined above. This will be the main purpose of our future work. We also intend in our future work to study the use of our proposed approach to detect different classes of botnets (beyond P2P botnets).

REFERENCES

[1] J. Leonard, S. Xu, and R. Sandhu, "A Framework for Understanding Botnets," in *Proceedings of the International Workshop on Advances in Information Security (WAIS at ARES)*, (Fukuoka, Japan), Fukuoka Institute of Technology, March 2009.

[2] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "Bothunter: detecting malware infection through ids-driven dialog correlation," in *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, (Berkeley, CA, USA), pp. 12:1–12:16, USENIX Association, 2007.

[3] P. Baecher, M. Koetter, M. Dornseif, and F. Freiling, "The nepenthes platform: An efficient approach to collect malware," in *In Proceedings of the 9 th International Symposium on Recent Advances in Intrusion Detection (RAID*, pp. 165–184, Springer, 2006.

[4] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon, "Peer-to-peer botnets: overview and case study," in *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, (Berkeley, CA, USA), pp. 1–1, USENIX Association, 2007.

[5] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *Proceedings of the 17th conference on Security symposium*, (Berkeley, CA, USA), pp. 139–154, USENIX Association, 2008.

[6] F. Giroire, J. Chandrashekar, N. Taft, E. Schooler, and D. Papagiannaki, "Exploiting temporal persistence to detect covert botnet channels," in *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*, RAID '09, (Berlin, Heidelberg), pp. 326–345, Springer-Verlag, 2009.

[7] W. T. Strayer, D. Lapsley, R. Walsh, and C. Livadas, "Botnet Detection Based on Network Behavior," in *Botnet Detection: Countering the Largest Security Threat* (W. Lee, C. Wang, and D. Dagon, eds.), Springer-Verlag, 2007.

[8] D. Liu, Y. Li, Y. Hu, and Z. Liang, "A p2p-botnet detection model and algorithms based on network streams analysis," in *Future Information Technology and Management Engineering (FITME), 2010 International Conference on*, vol. 1, pp. 55 –58, 2010.

[9] S.-K. Noh, J.-H. Oh, J.-S. Lee, B.-N. Noh, and H.-C. Jeong, "Detecting p2p botnets using a multi-phased flow model," in *Digital Society, 2009. ICDS '09. Third International Conference on*, pp. 247 –253, 2009.

[10] M. Tavallaee, W. Lu, and A. A. Ghorbani, "Online classification of network flows," in *Proceedings of the 2009 Seventh Annual Communication Networks and Services Research Conference*, (Washington, DC, USA), pp. 78–85, IEEE Computer Society, 2009.

[11] F. Liu, Z. Li, and Q. Nie, "A new method of p2p traffic identification based on support vector machine at the host level," in *Proceedings of the 2009 International Conference on Information Technology and Computer Science - Volume 02*, (Washington, DC, USA), pp. 579–582, IEEE Computer Society, 2009.

[12] A. Callado, J. Kelner, D. Sadok, C. Alberto Kamienski, and S. Fernandes, "Better network traffic identification through the independent combination of techniques," *J. Netw. Comput. Appl.*, vol. 33, pp. 433–446, July 2010.

[13] C. Wei, A. Sprague, and G. Warner, "Detection of networks blocks used by the storm worm botnet," in *Proceedings of the 46th Annual Southeast Regional Conference on XX*, ACM-SE 46, (New York, NY, USA), pp. 356–360, ACM, 2008.

[14] H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. Youssef, M. Debbabi, and L. Wang, "On the Analysis of the Zeus Botnet Crimeware Toolkit," *Proceedings of the 8th Annual Conference on Privacy, Security and Trust.*, pp. 31–38, 2010.

[15] A. Dries and U. Rückert, "Adaptive concept drift detection," *Statistical Analy Data Mining*, vol. 2, no. 5-6, pp. 311–327, 2009.

[16] I. H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. J. Cunningham, "Weka: Practical machine learning tools and techniques with java implementations," 1999.

[17] T. Abeel, Y. Van de Peer, and Y. Saeys, "Java-ML: A Machine Learning Library," *Journal of Machine Learning Research*, vol. 10, pp. 931–934, Apr. 2009.