Botnet Behavior Detection using Network Synchronism

Chapter · April 2011

DOI: 10.4018/978-1-60960-836-1.ch005

1 author:

Sebastián García
Czech Technical University in Prague
35 PUBLICATIONS **46** CITATIONS

Privacy, Intrusion Detection, and Response:

Technologies for Protecting Networks

Peyman Kabiri
Iran University of Science and Technology, Iran

Information Science
REFERENCE

Managing Director: Lindsay Johnston
Senior Editorial Director: Heather Probst
Book Production Manager: Sean Woznicki
Development Manager: Joel Gamon
Development Editor: Michael Killian
Acquisitions Editor: Erika Gallagher
Typesetters: Mackenzie Snader
Print Coordinator: Jamie Snavely
Cover Design: Nick Newcomer

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data
Privacy, intrusion detection, and response: technologies for protecting networks / Peyman Kabiri, editor.
p. cm.
Includes bibliographical references and index.
Summary: " This book discusses the latest trends and developments in network security and privacy, and serves as a vital reference for researchers, academics, and practitioners working in the field of privacy, intrusion detection, and response"--Provided by publisher.
ISBN 978-1-60960-836-1 (hardcover) -- ISBN 978-1-60960-837-8 (ebook) -- ISBN 978-1-60960-838-5 (print & perpetual access) 1. Computer networks--Security measures. I. Kabiri, Peyman, 1968-
TK5105.59.P757 2012
005.8--dc23
2011019909

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 5
Botnet Behavior Detection using Network Synchronism

Sebastián García
Universidad Nacional del Centro University, Argentina

Alejandro Zunino
Universidad Nacional del Centro University, Argentina

Marcelo Campo
Universidad Nacional del Centro University, Argentina

ABSTRACT

Botnets' diversity and dynamism challenge detection and classification algorithms depend heavily on static or protocol-dependant features. Several methods showing promising results were proposed using behavioral-based approaches. The authors conducted an analysis of botnets' and bots' most inherent characteristics such as synchronism and network load within specific time windows to detect them more efficiently. By not relying on any specific protocol, our proposed approach detects infected computers by clustering bots' network behavioral characteristics using the Expectation-Maximization algorithm. An encouraging false positive error rate of 0.7% shows that bots' traffic can be accurately separated by our approach by analyzing several bots and non-botnet network captures and applying a detailed analysis of error rates.

DOI: 10.4018/978-1-60960-836-1.ch005

INTRODUCTION

In the last decade botnets have evolved from being used as a personal activity platform to becoming a financially aimed structure controlled by malicious groups (Wilson, 2007). A botnet is a network of remotely controlled, compromised computers, used for malicious purposes. Hosts in a botnet are called 'Bots' and the owner of a botnet is called 'Botmaster'. From small DDoS (Distributed Denial of Service attacks) to world wide spam campaigns, botnets have become the technological backbone of a growing community of malicious activities (Clinton, 2008) and remain as the most significant threat to the Internet today.

Technology to control malicious programs remotely first surfaced in late 1999 and since then their primary goal has been to obtain financial gain. This situation forced the development of several botnet detection technologies trying to cope with the attacks, but botnets resisted besiege security measures resting on their home based client attacks, circumventing security methods (Stone-Gross, Cova, Cavallaro, Gilbert, Szydlowski, Kemmerer, Kruegel & Vigna, 2009), encryption and anti-reverse engineering techniques. Although the IRC (Internet Chat Relay) protocol has been the most used means of communication among bots, in the last couple of years the trend towards decentralized networks, like P2P (Peer to Peer) (Yan, Eidenbenz & Nago, 2009) (Kang, Zhang, Li & Li, 2009) and Fast-Flux (Ssac, 2008), has made more difficult to shut botnets down.

Several botnet detection methods have been proposed to cope with this problem. A general classification schema includes signature-based methods, protocol-dependant feature analysis and some more recent techniques based on network behavior, but most of these approaches only detect a subset of botnets, limiting their applicability. Signature-based approaches (like looking for certain IRC messages or certain DNS names) only detect what they were configured to. Most of these approaches do not have a correct error rate analysis because they did not propose a testing environment that includes non-botnet data.

To detect every botnet, we find out what botnets and bots have in common. A thorough analysis was performed to learn their most inherent characteristics. Our proposal works under the assumption that botnets most typical characteristics are maliciousness (attacking and infecting, sending SPAM, DDoS, etc.), being remotely managed and synchronization. We also found that bots might synchronize differently when scanning new victims, downloading binary updates, attacking sites, asking for orders or receiving orders, among others situations.

This chapter proposes a new method for bots detection based on network behavioral patterns. We aim at detecting bots in a general practical manner regardless of its connection protocol.

Our work use clustering algorithms (Baeza-Yates, 1999) to group network flows, revealing the relationships between the number of IP addresses and ports among time intervals. These relationships may characterize the most inherent botnet activities. It is known that during some botnet life cycle phases, a single bot computer generates high network flow rates within very short time periods (Gu, 2008), for example, during Spam sending, DDoS attacks, network scanning and botnet distribution. We propose to extract TCP (Transport Control Protocol) flows from the network, aggregating three TCP features within a predefined time window. These features are the amount of unique source IP address (referred hereafter as *sips*), the amount of unique destination IP address (referred hereafter as *dips*) and the amount of unique destination ports (referred hereafter as *dports*). These features had been used to detect manual network attacks before (Onut & Ghorbani, 2007). Our first hypothesis is that these features can also be used to identify bots network behaviors. The algorithm has been validated by using labeled data from real botnets in the wild and labeled data from real non-botnet computers. This data set allowed us to achieve a verified and robust algorithm.

Some advantages of our method include bots detection independent of encrypted traffic and protocol details and that bots detection within the first stages of infection. Our second hypothesis is that a bot could be detected because of the high amount of connections generated in a time window. The preliminary experimental results reported, suggest that this detection can be accomplished successfully under certain conditions. Bots and non-botnet traffic showed different network characteristics that helped to separate them in clusters.

This chapter makes two contributions. First the separation of aggregated network flows in time windows using non-protocol dependant bots features, and second the clustering of them using the Expectation-Maximization algorithm with labeled data.

The rest of the chapter is organized as follows: Section Background describes previous work in the area; Section Proposed Technique shows details about our approach; Section Validation explains corroboration procedures; Section Data Acquisition shows how network data has been obtained; in subsection Data Processing the data preparation steps are explained; Section Results discusses the experiment outcomes; Section Future Research Directions refers to future work we are planning and finally Conclusions are presented.

BACKGROUND

Some approaches have been proposed to detect botnets in recent years using network features, including the study of anomalous DNS (Domain Name System) usage (Villamarin-Salomon & Brustoloni, 2008), IRC protocol analysis (Mazzariello,

2008) and P2P network features (Kang et al., 2009) among others (Shahrestani, Ramadass & Feily, 2009) (Zhu, Lu, Chen, Fu, Roberts & Han, 2008), but the variability of botnets and their rapid mutation have forced researchers from different security domains to consider the analysis of behavioral patterns as a solid foundation for botnet detection.

Particularly, network behavior detection has been studied from different perspectives: traffic classification based on flow characteristics (Strayer, Lapsely, Walsh & Livadas, 2007), protocol analysis based on temporal-frequent features (Lu, Tavallaee, Rammidi & Ghorbani, 2009) and network analysis based on spatial-temporal correlation (Gu, 2008), where activity response crowds on IRC botnets are grouped together by destination IP address-port pairs and time windows.

Behavior analysis of bots, based on protocol-dependent features, has proved to be successful only under certain conditions. There seems to be three main problems with behavioral protocol-dependant approaches. First, only a subset of botnets can be detected analyzing protocol dependent characteristics, e.g., analyzing IRC messages behavior is not enough to detect P2P botnets. Second, new unseen botnets are unlikely to be detected if a new protocol is used. Finally, botmasters can change their current malware protocol characteristics used to detect their botnet (Stinson & Mitchell, 2008).

Addressing the aforementioned problems, botnets invariants were sought in order to detect them independently of protocol changes and implementation details. Botnet correlation was proposed (Gu, Perdisci, Zhang & Lee, 2008) among other techniques (Liu, Chan, Yan & Zhang, 2008), based on the fact that every bot in a botnet normally act at the same time. However, synchronization has not been deeply studied, and temporal correlation approaches were found to be evadable under certain circumstances (Chen, Chen & Wang, 2009), for example, proposing a time delay in bot responses before applying commands.

Our feature set was selected based on a previous work of the best network features for intrusion detection (Onut & Ghorbani, 2007). They defined a time window based detection schema, where some features related to the number of TCP connections were used. We extended these paper ideas to the botnet detection problem. Detection of botnets using groups of source IP address and groups of destination of IP address was also proposed by (Li, Goyal, & Chen, 2007). They used a honeypot to analyze this *Sips/Dip* relationship.

PROPOSED TECHNIQUE

Detecting botnets behavior and bots behavior are two quite different problems. Bot detection can be accomplished by the distinguishable behavior of one host. Botnet

behavior can be detected, for example, based on the synchronization of its individual bots, regardless of the amount of flows per second. Our technique focuses on differentiating bot behavior among non-botnet traffic.

The first step of our methodology consists in capturing network data from infected computers. This could be done by using the tcpdump tool (Lawrence Berkeley National Labs, 2010) on any machine in the wired network, taking care of not losing packets and that no other traffic is captured.

The second step entails extracting information from captures files and TCP flow separation. A tool called tcptrace (Shawn Ostermann, 2010) is used to find out every TCP flow, including its start time and network characteristics. Flow information is used because botnets tend to generate new flows almost constantly and use them for a short time period, where a non-botnet computer tends to generate few flows per second but use each one of them for long time periods.

As we aim at detecting synchronization, the third step of our approach involves dividing flows in one-second time windows, allowing us to monitor and identify bots behavior. Each time window contains aggregated information about every TCP flow within it. Then, for every TCP flow in that time window, we calculate the amount of unique source IP addresses, unique destination IP addresses and unique destination ports. This information is summarized in an attempt to reduce data processing complexity and only save useful information for further analysis. A time window can then be represented by a four dimensional vector containing the window Id, amount of unique source IP addresses in that time window, amount of unique destination IP addresses in that time windows and amount of unique destination TCP ports in that time window. An example vector should look like this: [23, 1, 10, 1].

The fourth step involves clustering these vectors using the EM (Expectation-Maximization) algorithm (Dempster, Laird & Rubin, 1977). Expectation-maximization is a method for finding maximum likelihood estimates of parameters in statistical models with incomplete data, and has been used successfully before in traffic flow analysis for characterizing communication connectivity patterns (Chen, Li & Cao, 2009), botnet detection (Masud, Gao, Khan, Han, & Thuraisingham, 2008) and spam detection (Zhang, 2009).

A basic introduction to the algorithm follows. We have a density function $p(x|\Theta)$ that is defined by a set of parameters Θ. We also have a known dataset $X=\{x1, \ldots, xn\}$, of size N, extracted from this distribution. The resulting density for the samples can be seen in Equation (1).

$$p(X \mid \Theta) = \prod_{i=1}^{N} p(x_i \mid \Theta) = L\left(\Theta \mid X\right) \tag{1}$$

This function L is called the likelihood of the parameters given the known data. The likelihood is a function of the parameters Θ where the known data X is fixed. In the maximum likelihood problem, represented by Equation (2), the goal is to find the Θ parameter that maximizes L.

$$\Theta' = \arg\max_{\Theta} L\left(\Theta \mid X\right) \tag{2}$$

The EM algorithm is a technique to solve the maximum likelihood problem. In our proposal, the windows vectors are the known data, coming from different statistical distributions (botnets). How many distributions and which vectors correspond to each one are the hidden variables to be found. Each botnet generates packets given a unique statistical distribution with unknown parameters Θ and our goal is to maximize the a-posteriori probability of these parameters given the known data (vectors) in the presence of hidden data.

The intuition of EM is to alternate between estimating the unknowns Θ and the hidden variables. The idea is to start with a guess of these parameters Θ and to determine for each vector which underlying distribution is more likely to have generated the observed data (using the current parameters estimates). Then, assume these guessed assignments to be correct and apply the regular maximum likelihood estimation procedure to get next Θ. The procedure iterates until convergence to a local maximum. The algorithm computes the models parameters and assigns the cluster number with highest probability to each window vector (McGregor, Hall, Lorier & Brunskill, 2004).

Our already captured and labeled data allow us to verify the proposed detection method error rates later. Expectation-maximization algorithm provides a simple, easy-to-implement and efficient tool for learning parameters of a model. Weka's (Hall, Frank, Holmes, Pfahringer, Reutemann & Witten, 2009) implementation of this algorithm was used because of the independence assumption of the attributes in the model, making it suitable for our purposes.

In this chapter we do not work with the concept of *normal* traffic, instead our algorithm focuses on differentiating bots behavior from non-botnet behavior. What our algorithm classifies as non-botnet could be normal traffic or manual attacks, that is why we use the term non-botnet. It can not find the difference between bots behaviors and *normal* behavior, because *normal* traffic varies greatly. This also forces us to carefully analyze non-botnet traffic, looking for manual attacks or misconfigurations.

It was important to consider the distinction between bot traffic and intrusion attempts, because non-botnet traffic could consist of both normal traffic and

Table 1. Labeled network data captures details

Name	Duration	Unique Flows
Botnet1	11h:12m:29s	37389
Botnet2	00h:20m:30s	21124
Botnet3	10h:11m:33s	34117
Non-botnet1	00h:10m:18s	45148
Non-botnet2	03h:28m:14s	1517
Non-botnet3	04h:19m:36s	976

manual or automatic attacks. Network attacks are often done manually and using automatic tools. Manual attacks have low traffic rates and do not have synchronous characteristics, while automatic attacks tend to have high traffic rates and could be misdetected by our method as bot traffic. One of the most traffic consuming attacks is the port scanning activity, which is often an essential part of the attack process. To cope with this problem we propose some experiments in the Results Section to accurately separate bots traffic and port scanning activity.

DATA ACQUISITION

Three different test beds were used to acquire the 92.630 unique botnet flows and more than 47500 non-botnet flows, against witch our algorithm was verified. We considered a TCP flow as the group of packets exchanged between two hosts and two ports in a single connection. Table 1 shows details about captures.

The first test bed, used for botnet1 and botnet3 captures, is composed of a Linux computer virtualizing a Microsoft Windows XP sp3 operating system using VirtualBox. The VM (Virtual Machine) connects to the Internet through a DHCP (Dynamic Host Configuration Protocol) enabled home DSL router. Packet sniffing is performed in the Linux box with certain filters applied; assuring that only Windows packets were caught. Current botnets usually target this type of home computers, looking for its high bandwidth and personal sensitive information (Corrons, PandaLabs, 2010), (Nazario, 2006). Botnet1 capture corresponds to a 'Virut' malware variant called 'Virut.n' and botnet3 capture corresponds to an 'Agobot' family member called 'Rbot'.

Botnet2 was captured with a second test bed composed of 18 hosts in an internal sub-network of a university campus. The network was monitored while five hosts were infected. The botnet2 malware is a spam sending 'Agobot' variant called 'eldorado'.

The third test bed used for non-botnet2 and non-botnet3 traffic captures is composed of one Linux computer connected to a University Campus WiFi Network. We ensure that the traffic was non-botnet by manually inspecting traffic and scanning it with Snort NIDS. Non-botnet traffic includes SMTP mail, web sites with AJAX updates, edonkey like protocols, torrent protocols, operating system updates, and web traffic using several Google online tools.

The non-botnet1 capture used the third test bed to perform an nmap port scanning activity, representing one of the most common network attacks. Using nmap version 5.35DC18, we performed a TCP SYN scan of the 65536 remote ports of one destination computer without changing nmaps default timing behavior.

Botnet1 malware, called 'Virut', is an IRC-based variant capable of giving total control of the computer to the remote botmaster. Botnet3 malware, called 'Rbot', is a highly configurable IRC controlled bot used to gain access to infected computers in several different ways, including password cracking, SMB (Server Message Block) shares access and exploiting software vulnerabilities. This bot can be instructed to download and execute files, perform denial of service (DoS) attacks or log keystrokes among other attacks. Botnet2 bot variant, called 'eldorado', is a web controlled bot designed to send spam, self update and perform malicious actions. These bots do not recognize the VirtualBox setup and consequently the infection was performed successfully. Newer bots with the ability to detect virtual environments were used, but its analysis was left for future work.

Data Processing

Data processing is a fundamental part of the analysis as it determines which characteristics will be taken into account. Processing steps include data verification, data labeling, data preprocessing and feature extraction among others. The first phase of data processing involves the extraction of TCP flows from the captured pcap data using the tcptrace tool (using the command *tcptrace -nl <pcap-file>*). Tcptrace generates a text file with every TCP flow characteristic.

The second phase is performed with a specially developed tool, called tcptrace-reader.py, to process the tcptrace output file. As a result of this phase, a text file is generated with the following information per flow: start timestamp in UNIX time format, source IP address in decimal format, destination IP address in decimal format and destination port. Each of these lines is called an instance. In this phase we also assign 'botnet' and 'non-botnet' labels to every instance of both types of traffic by hand.

Phase three uses the tcptrace-reader.py tool to perform time windows aggregation of instances, creating vectors in an Arff (Attribute-Relation File Format) file.

This Arff file will be used for clustering in Weka. This phase also generates a text file with the relation of instances and vectors for future back reference. When the algorithm assigns a certain vector to a cluster, we need to know which IP addresses were included on it.

The fourth phase consists in the processing of the Arff file using the Weka toolset. We apply the EM clustering algorithm to this data set. After clustering is done, a complete Arff file is also saved from Weka, adding the clusters assignments to each vector.

Once the clustering phase is finished, we compute the percentage of botnet and non-botnet instances in each cluster. This is done in the fifth phase using another specially developed tool called analysis.sh. For each cluster in the complete Arff file, this tool retrieves every vector on it and for each vector it retrieves every instance on it with its label. We can then compute how many botnets and non-botnets instances were assigned to each cluster.

VALIDATION

Validation of every phase was conducted to prove the practical soundness of our method. Following the methodology steps described in Proposed Technique section, in the first step, validation procedures included ensuring a clean experimental setting. For the first test bed, a Virtualized Windows XP SP3 was freshly installed, scanned with antivirus products and its traffic analyzed both with Snort NIDS (Network Intrusion Detection System) (Roesch, 1999) and by hand.

This configuration was used to create Virtual Machines for computers that were later infected. Confirmation of malware binaries was conducted using VirusTotal (Hispasec Sistemas, 2010) and the EUREKA! automated Malware Binary Analysis Service (Yegneswaran, Saidi, Porras, Sharif, Mark & President, 2008). Also a manual packet analysis was done to verify malware activity of botnet1 and botnet3 captures.

Validation in the third step was conducted through window counting, e.g., a one-hour capture must have almost 3600 one-second windows. Windows with a high amount of flows were manually verified.

The clustering step was verified by labeling data. Trustworthy botnet and non-botnet experiments were performed, building an accurate cluster compare base. Once available, this information allowed interpretation of the resulting clusters and experiments improvement. Labeling was performed by manually analyzing every flow using Wireshark (Wireshark, 2010).

Finally, an experimental validation was conducted on the hypothesis about the bot behavior detection methodology. Our hypothesis, based on the time window and

network load concept proved in (Gu, 2008), states that bots can be detected analyzing the correlation of the time window *timestamp*, the amount of unique source IP addresses within the time window, the amount of unique destination IP addresses within the time window and the amount of unique destination ports within the time window. For example, finding 1 *sips*, 20 *dips* and 1 *dports* within a one second time frame might mean that we have detected a bot looking for vulnerabilities. This validation step generated next section Results.

RESULTS

Several experiments were conducted to analyze botnet behavior. Every experiment included a combination of one botnet and one non-botnet capture, because in this type of statistical experiments it is paramount to have a realistic setup. This combination was done concatenating both text files at tcptrace level and modifying captures start time. This approach proved easier than modifying and mixing pcap traces.

Non-botnet computers with normal data have continuous low traffic generation rates most of the time, whereas bots tend to have bursts of high traffic rates (Yen & Lee, 2009). This difference can make bots flow generation rate roughly three times faster than a non-botnet computer. Clustering experiments, then, tend to have an uneven number of packets, resulting in natural unbalanced experiments. If we have a balanced data set (the number of samples in botnet and non-botnet classes is almost equal), classifier error rates could be correctly interpreted because each instance will belong to one class with 50% probability. If we have an unbalanced data set (the number of samples in different classes varies greatly) the error rate of a classical statistics classifier *could not* represent the true effectiveness of the classifier. Some studies had to modify the EM clustering method to deal with unbalanced data sets (Wu, 1995). Both types of experiments were conducted to analyze this situation.

Selection criteria of both captures for each experiment had two steps. In the first step we selected captures based on how much representative of its own class they were. Captures with more non-botnet applications were preferred and botnets with more traffic rates were preferred. In the second step we decided how to mix both captures based on the assumption that superimposed traffic is more difficult to differentiate. When non-botnet and botnet traffic was only concatenated but not superimposed, separation of both classes was straight forward. Mixture was done trying to insert the smaller traffic into the larger one.

Our approach proposes to differentiate bot traffic from non-botnet traffic, but the latter can include normal traffic as well as manual attacks attempts and automatic penetration testing. It is very important that our algorithm does not confuse botnets

Table 2. Detection percentages of 45148 Non-botnet1 flows mixed with 21124 Botnet2 flows

Cluster Number	Botnet Flows	Non-botnet Flows	Botnet %	Non-botnet %
0	2128	0	100%	0%
1	7129	0	100%	0%
2	5633	0	100%	0%
3	0	7971	0%	100%
4	480	0	100%	0%
5	240	0	100%	0%
6	3322	0	100%	0%
7	2192	0	100%	0%
8	0	37177	0%	100%

with these intrusion attempts. One of the most common phases in every network attack is the port scanning activity, which generates high traffic rates. Clustering of concatenated port scanning and bot traffic is analyzed in the first experiment (Table 2), where the higher traffic rate of the port scan results in a very clear traffic separation. This experiment suggests that botnet traffic and network scans have different network behaviors.

The aim of the second experiment (Table 3) was to analyze the Rbot malware within the traffic of one non-botnet computer. Assuming that a computer behaves almost identically during fourteen days of office work, the original four-hour non-botnet capture was copied twenty times, making each copy start in a different day at the same hour. We finally had 19443 non-botnet flows from September 13 to October 2 and 34118 botnets flows from September 14 to September 15. After aggregating flows into one-second time windows, we generate the Arff file for this traffic and processed it with the EM algorithm, which generated four clusters.

Table 3. Detection percentages of 19443 Non-botnet3 flows mixed with 34118 Botnet3 flows

Cluster Number	Botnet Flows	Non-botnet Flows	Botnet %	Non-botnet %
0	14433	24	99.83%	0.16%
1	1517	18721	7.49%	92.50%
2	3539	14	99.60%	0.39%
3	14629	684	95.53%	4.46%

Table 4. Detection percentages of 1517 Non-botnet2 flows mixed with 34118 Botnet3 flows

Cluster Number	Botnet Flows	Non-botnet Flows	Botnet %	Non-botnet %
0	2147	1358	61.25%	38.74%
1	3539	12	99.66%	0.33%
2	14469	81	99.44%	0.55%
3	13963	66	99.52%	0.47%

This experiment tried to differentiate between 19 days of low rate non-botnet traffic and ten hours of a high rate bot burst in the middle of it. The second experiment also analyzed how the algorithm performed with an almost even number of packets in both classes.

The aim of the third experiment (Table 4) was to analyze an unbalanced and more realistic situation, where three hours of non-botnet traffic from one computer was mixed with ten hours of bots traffic. Non-botnet capture was modified to start on September 15, in the middle of botnet traffic. These capture files were selected with the goal of analyzing an IRC bot along side with a normal computer.

The aim of experiment four (Table 5) was to analyze one infected computer among a network of non-botnet computers. The five hour non-botnet capture was modified to start on September 15, in the middle of the ten-hour botnet traffic. Non-botnet traffic was composed of more than 40 office computers. This is a very realistic setup, where a lot of computers generate very low traffic rates.

False Positives and False Negatives

We analyzed error rates from two points of view. From the first point of view, false positives are accounted each time a non-botnet flow is detected as botnet. Thus, false negatives are accounted each time a botnet flow is detected as non-botnet. This is

Table 5. Detection percentages of 977 Non-botnet3 flows mixed with 34118 Botnet3 flows

Cluster Number	Botnet Flows	Non-botnet Flows	Botnet %	Non-botnet %
0	2147	916	70.09%	29.90%
1	13918	22	99.84%	0.15%
2	3539	14	99.60%	0.39%
3	14469	25	99.82%	0.17%
4	45	0	100%	0%

the most simple and theoretical point of view. From the second point of view, false positives are accounted each time a non-botnet IP address is detected as botnet at least once in a predefined period of time and false negatives are accounted each time a botnet IP address is detected as non-botnet during the whole same predefined period of time. Both points of view are discussed next.

From the first point of view, a quick analysis of our algorithm effectiveness could be done estimating false positives and false negatives regarding each IP address. Table 6 summarizes this information, where calculation about a certain IP address is made against the total number of times its flows where detected as botnet or non-botnet. We also found that IP addresses in this table are representative of the most active computers in the experiments. First experiment was not included in this calculation because its goal was to prove that botnet traffic and intrusion attempts could be successfully separated.

The second experiment false positives error rates, where a non-botnet IP address is detected as botnet, are reasonably low. There is only one false negative, where one IP address is detected as non-botnet 6.45% of the time. A detailed manual analysis showed that non-botnet label was assigned to this botnet when a single flow per second was generated. This was an example of a botnet behaving like a non-botnet computer. From a practical point of view, if we consider that IP address 10.1.1.1 was the network web proxy and that it can be white-listed by the administrator, perhaps we could achieve a better error rate.

The third experiment has, comparatively, higher false positive error rates. An analysis of these errors shows that both botnet and non-botnet computers were

Table 6. IP addresses error evaluation

Experiment	Error Type	IP address	Actually is	#Detected as Non-botnet	#Detected as Botnet	Error %
2	FP	10.1.1.1	Non-botnet	10570	180	1.67%
2	FP	192.168.2.79	Non-botnet	18697	542	2.81%
3	FP	10.1.1.1	Non-botnet	466	1	0.21%
3	FP	192.168.2.71	Non-botnet	5	1	16.66%
3	FP	192.168.2.74	Non-botnet	783	89	10.20%
3	FP	192.168.2.76	Non-botnet	235	31	11.65%
4	FP	192.168.2.79	Non-botnet	910	60	6.18%
2	FN	192.168.3.104	Botnet	1103	15978	6.45%
3	FN	192.168.3.104	Botnet	1563	15518	9.15%
4	FN	192.168.3.104	Botnet	1563	15518	9.15%

clustered together because they generated several connections at the same time. This situation is not uncommon and we are planning to analyze it carefully in future work.

The fourth experiment shows an increment in false negatives values and its false positive error rate is notably higher because fewer instances were used. In the worse case, 6.18% of the time a non-botnet computer is detected as botnet.

From Table 6 we can conclude that the most challenging problem is to classify a non-botnet computer traffic that behaves like a botnet as *non-botnet*.

After analyzing Table 6, a confusion matrix is shown in Table 7 to better analyze error rates meanings. A confusion matrix is a visualization tool commonly used in supervised learning, where each row represents how each instance was detected and each column what it actually was.

We have 124,291 different instances that the algorithm has to classify. The False Positive Rate (FPR) is the proportion of non-botnet flows that yield a botnet outcome (904 in our results) among all the tests performed. *FPR = 0.7%*. The False Negative Rate (FNR) is the proportion of botnet flows that yield a non-botnet outcome (4229 in our results) among all the tests performed. *FNR = 3.4%*.

Some technical results can also be extracted from Table 7, for example, *0.91%* is the proportion of *non-botnet* flows that were detected as *botnet* among the presumed botnets, i.e., from the total amount of infection alerts that the network administrator receives from our algorithm, only *0.91%* were not infected.

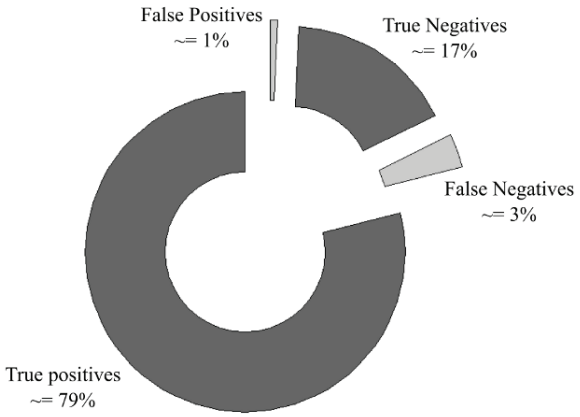
From Table 7 we can also conclude that *16.74%* is the proportion of botnet flows that were detected as *non-botnet* among the presumed non-botnet computers. This would be the proportion of infected flows that the administrator will not see. Note that we are talking about flows and not computers. An analysis from the computer point of view is done later.

Still from a technical perspective, if this algorithm was used in real botnet shut-down campaigns, 4.13% of the bots will not be detected and will remain active. Finally, we are going to bother 4.12% of our internal users trying to clean their non-infected computers.

Table 7. Confusion flow matrix

	Actual			
Detected		Botnet	Non-Botnet	
	Botnet	98125	FP=904	0.91%
	Non-Botnet	FN=4229	21033	16.74%
		4.13%	4.12%	

Figure 1. Error Analysis



A graphical analysis of the confusion matrix showing approximate percentages can be seen in Figure 1. It shows the relation between errors and true detections.

Until now we have discussed error rate analysis when trying to detect every bot network flow. We are going to analyze results now from a different point of view; the need to detect infected computers in a real environment. A network administrator wants to disinfect computers and to detect bots in the traffic. This, more practical, point of view analyze error rates taking into account the problem domain and scope.

It is infeasible to classify every botnet flow as *botnet* because their dynamic and evolving nature generates traffic with different characteristics. An infected computer behaves like a non-botnet computer most of the time and during some of its life cycle phases it does not generate traffic like a botnet at all. Following this reasoning, if we can detect the botnet at least once during the one-hour time frame, it will be enough to stop it. If we consider a time frame of, for example, one hour, during which it will be enough to detect a bot at least once, we can define the following errors.

On the one hand, a real positive is considered now when a bot IP address is classified as bot at least once within this time frame. On the other hand, a real negative is considered when a non-botnet computer IP address is classified as non-botnet *always* within this time frame. False positives are accounted only when a non-botnet IP address is considered bot at least once within this time frame and false negatives, then, are accounted only when a bot IP addresses is classified as non-botnet

every time within the same time frame. This is enough for most practical detection scenarios, and it shows the true impact of the algorithm in real scenarios clearly.

CONCLUSION

Botnet detection is still an ongoing research topic. In this chapter we presented a new approach for bot network behavior detection using the Expectation-Maximization clustering algorithm. Our work is based on the hypothesis that bots behave different than non-botnets computers. To analyze this situation, we captured several network flows with botnet and non-botnet traffic using three different test beds. This traffic included single and network captures of both current malware and not infected computers. After proper traffic pre-processing, we separated the traffic flows in time windows and studied the relationship between unique source IP addresses, unique destination IP addresses and unique destination ports within these time windows.

Error rates were analyzed closely to understand the needs of a network administrator and to know how our algorithm performed. Two points of view were proposed, one from the flow perspective and the other from the computer perspective. Results are encouraging, showing that bots can be differentiated from non-botnet traffic with a FPR of 0.7% and a FNR of 3.4%.

FUTURE RESEARCH DIRECTIONS

Several improvements are planned for future research. First of all, UDP (User Datagram Protocol) traffic will be considered in our captures, because plenty of botnet traffic is performed using this protocol, most notably DNS queries, that can be up to 40% of the total traffic. Another improvement will be to analyze the total amount of transferred bytes within a single flow, in order to capture long lived botnet connections with their C&C server. In the clustering area, we are currently working on analyzing cluster assignments to synthesize classification rules that later allows us to automatically classify clusters. In the network capture area, several one-week long LAN (Local Area Network) non-botnet capture are planned, during which different botnets will be executed from time to time. An extension in the number of captured botnets is planned too. To better analyze the difference between bot traffic and intrusion attempts, new experiments are planned, like an experiment with botnet, non-botnet and port scanning captures. Finally an important improvement will be done in the data set production and processing, creating a methodology for flow capture, publication, labeling, comparison and verification.

ACKNOWLEDGMENT

We thank the editor and anonymous reviewers for their helpful comments and suggestions to improve the quality of this paper. Also, thanks to ANPCyT for supporting this research through grants PAE-PICT 2007-02311 and PAE-PICT 2007-02312.

REFERENCES

Baeza-Yates, R. A., & Riberiro-Neto, B. A. (1999). *Modern information retrieval*. Boston, MA: ACM-Press / Addison-Wesley Longman Publishing Co., Inc.

Chen, A., Li, L., & Cao, J. (2009). Tracking cardinality distributions in network traffic. In *Proceedings of IEEE 28th Conference on Computer Comunications (INFOCOM 2009)* (pp. 819-827).

Chen, Z., Chen, C., & Wang, Q. (2009). Delay-tolerant botnets. In *Proceedings of 18th International Conference on Computer Communications and Networks (ICCCN 2009)* (pp. 1-6).

Corrons, L. (PandaLabs). (2010). *Mariposa botnet*. Retrieved August 17, 2010 from http://pandalabs.pandasecurity.com/mariposa-botnet/

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B. Methodological*, *39*(1), 1–38.

Gu, G. (2008). *Correlation-based botnet detection in enterprise networks* (Doctoral dissertation, Georgia Institute of Technology).

Gu, G., Perdisci, R., Zhang, J., & Lee, W. (2008). BotMiner: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In *Proceedings of the 17th Conference on Security symposium* (pp. 139-154). Berkeley, CA: USENIX Association.

Gu, G., Zhang, J., & Lee, W. (2008). BotSniffer: Detecting botnet command and control channels in network traffic. In *Proceedings of the 15th Network and Distributed System Security Symposium (NDSS)*, San Diego, CA.

Ha, D., Yan, G., Eidenbenz, S., & Ngo, H. (2009). On the effectiveness of structural detection and defense against p2p-based botnets. In *IEEE/IFIP International Conference on Dependable Systems & Networks (DSN '09)* (pp. 297-306).

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, H. I. (2009). The weka data mining software: An update. *ACM SIGKDD Explorations Newsletter*, *11*(1), 10–18. doi:10.1145/1656274.1656278

Hispasec Sistemas. (2010). *Virus Total*. Retrieved March 30, 2010, from http://www.virustotal.com/es/

Kang, J., Zhang, J.-Y., Li, Q., & Li, Z. (2009). Detecting new p2p botnet with multi-chart cusum. In *Proceedings of the International Conference on Networks Security, Wireless Communications and Trusted Computing (NSWCTC '09)* (vol. 1, pp. 688-691).

LBNL. (2010). *TCPDUMP & LiBPCAP*. Retrieved March 30, 2010 from http://www.tcpdump.org/

Li, Z., Goyal, A., & Chen, Y. (2007). Honeynet-based botnet scan traffic analysis. *Botnet Detection, 36*, 25–44. New York, NY: Springer.

Liu, L., Chen, S., Yan, G., & Zhang, Z. (2008). Execution-based bot-like malware detection. *Information Security* (LNCS 5222, pp. 97-113). Berlin/Heidelberg, Germany: Springer-Verlag.

Lu, W., Tavallaee, M., Rammidi, G., & Ghorbani, A. A. (2009). Botcop: An online botnet traffic classifier. In *7th Annual Communication Networks and Services Research Conference (CNSR '09)* (pp. 70-77).

Masud, M. M., Gao, J., Khan, L., Han, J., & Thuraisingham, B. (2008). A practical approach to classify evolving data streams: Training with limited amount of labeled data. In *8th IEEE International Conference on Data Mining (ICDM '08)* (pp. 929–934).

Mazzariello, C. (2008). Irc traffic analysis for botnet detection. In *4th International Conference on Information Assurance and Security (ISIAS '08)* (pp. 318-323).

McGregor, A., Hall, M., Lorier, P., & Brunskill, J. (2004). Flow clustering using machine learning techniques. In *Passive and Active Network Measurement* (LNCS, pp. 205-214). Berlin/Heidelberg, Germany: Springer-Verlag.

Mielke, C., & Chen, H. (2008). Botnets, and the cybercriminal underground. In *IEEE International Conference on Intelligence and Security Informatics (ISI 2008)* (pp. 206-211).

Nazario, J. (2006). *Botnet tracking: Tools, techniques, and lessons learned (Tech. Rep.)*. Chemsford, MA: Arbor Networks.

Onut, I. V., & Ghorbani, A. A. (2007). A feature classification scheme for network intrusion detection. *International Journal of Network Security*, *5*, 1–15.

Roesch, M. (1999). Snort-lightweight intrusion detection for networks. In *13th Systems Administration Conference (LISA '99)* (pp. 229-238).

Shahrestani, A., Ramadass, S., & Feily, M. (2009). A survey of botnet and botnet detection. In *3rd International Conference on Emerging Security Information, Systems and Technologies (SECURWARE '09)* (pp. 268-273).

Shawn, O. (2010). *Tcptrace*. Retrieved March 30, 2010 from http://www.tcptrace.org/

SSAC. (2008). *Ssac advisory on fast flux hosting and dns* (Tech. Rep.). ICANN Security and Stability Advisory Committee. Retrieved on March, 2008 from http://www.icann.org/en/committees/security/ssac-documents.htm.

Stinson, E., & Mitchell, J. (2008). Towards systematic evaluation of the evadability of bot/botnet detection methods. In *Proceedings of the 2nd Conference on USENIX Workshop on Offensive Technologies (WOOT '08)* (pp. 1-9). Berkeley, CA: USENIX Association.

Stone-Gross, B., Cova, M., Cavallaro, L., Gilbert, B., Szydlowski, M., & Kemmerer, R. …Vigna, G. (2009). Your botnet is my botnet: Analysis of a botnet takeover. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09)* (pp. 635-647). New York, NY: ACM.

Strayer, W., Lapsely, D., Walsh, R., & Livadas, C. (2007). Botnet detection based on network behavior. *Advances in Information Security, 36*, 1-24. New York, NY: Springer.

Villamarin-Salomon, R., & Brustoloni, J. (2008). Identifying botnets using anomaly detection techniques applied to dns traffic. In *Proceedings of the 5th IEEE Consumer Communications and Networking Conference (CCNC 2008)* (pp. 476–481).

Wilson, C. (2007). *Botnets, cybercrime, and cyberterrorism: Vulnerabilities and policy issues for congress* (Tech. Rep.). Washington, DC: Library of Congress Congressional Research Service.

Wireshark. (2010). *Wireshark: The world's foremost network protocol analyzer*. Retrieved March 30, 2010, from http://www.wireshark.org/

Wu, J. M. (1995). *Maximum likelihood estimation in the random coefficient regression model via the EM algorithm*. Phd Thesis, Texas Tech University, Lubbock.

Yegneswaran, V., Saidi, H., Porras, P., Sharif, M., Mark, W., & President, V. (2008). *Eureka: A framework for enabling static analysis on malware* (Tech. Rep.). Atlanta, Gerogia: Georgia Institute of Technology.

Yen, S.-J., & Lee, Y. (2009). Cluster-based under-sampling approaches for imbalanced data distributions (LNCS). *Expert Systems with Applications*, *36*, 5718–5727. doi:10.1016/j.eswa.2008.06.108

Zhang, L. (2009). A *sublexical unit based hash model approach for SPAM detection*. PhD Thesis, the University of Texas at San Antonio.

Zhu, Z., Lu, G., Chen, Y., Fu, Z., Roberts, P., & Han, K. (2008). Botnet research survey. In *32nd Annual IEEE International Computer Software and Applications (COMPSAC'08)* (pp. 967-972).

ADDITIONAL READING

Akiyama, M., Kawamoto, T., Shimamura, M., Yokoyama, T., Kadobayashi, Y., & Yamaguchi, S. (2007). A proposal of metrics for botnet detection based on its cooperative behavior. In *International Symposium on Applications and the Internet Workshops*. SAINT Workshops, (pp. 82-82). IEEE Computer Society.

Bächer, P., Holz, T., Kötter, M., & Wicherski, G. (2008). *Know your enemy: Tracking botnets*. Retrieved March 17, 2010 from HoneyNet Web site: http://www.honeynet.org/papers/bots/.

Barford, P., & Blodgett, M. (2007). Toward botnet mesocosms. In Proceedings of the first Workshop on *Hot Topics in Understanding Botnets, HotBots'07*, (pp. 6-6), Berkeley, CA, USA. USENIX Association.

Barford, P., & Yegneswaran, V. (2006). An Inside Look at Botnets. In *Special Workshop on Malware Detection*, Advances in Information Security, Springer Verlag.

Bayer, U., Comparetti, P., Hlauschek, C., Kruegel, C., & Kirda, E. (2009). Scalable, Behavior-Based Malware Clustering. In *Network and Distributed System Security Symposium* (NDSS).

Binkley, J. R., & Singh, S. (2006). An algorithm for anomaly-based botnet detection. In Proceedings of the 2nd conference on *Steps to Reducing Unwanted Traffic on the Internet,* SRUTI'06, (pp. 7-7), Berkeley, CA, USA. USENIX Association.

Caglayan, A., Toothaker, M., Drapaeau, D., Burke, D., & Eaton, G. (2009). Behavioral analysis of fast flux service networks. In Proceedings of the 5th Annual Workshop on *Cyber Security and Information Intelligence Research, CSIIRW '09,* (pp. 1-4), New York, NY: ACM.

Castle, I., & Buckley, E. (2008). The automatic discovery, identification and measurement of botnets. In Second International Conference on *Emerging Security Information, Systems and Technologies*, SECURWARE '08. (pp. 127-132).

Casullo, G. A., Fink, S. A., Jaime, J. M., & Tami, L. R. (2009). Webbotnets, la amenaza fantasma. In Proceedings of the *38 JAIIO, WSegI '09*, Buenos Aires, Argentina.

Chang, S., Zhang, L., Guan, Y., & Daniels, T. (2009). A framework for p2p botnets. In *WRI International Conference* on Communications and Mobile Computing. *CMC '09*, volume 3, (pp. 594-599).

Christodorescu, M., Jha, S., & Kruegel, C. (2008). Mining specifications of malicious behavior. In Proceedings of the *1st conference on India software engineering conference, ISEC '08,* (pp. 5-14), Hyderabad, India. ACM Press New York, NY: ACM.

Christodorescu, M., & Rubin, S. (2007). Can cooperative intrusion detectors challenge the base-rate fallacy? In *US, S., (Ed), Malware Detection, Advances in Information Security*, (pp. 193-209). Springer US.

Cooke, E., Jahanian, F., & McPherson, D. (2005). The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets. In [USENIX Association.]. *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet Workshop, SRUTI*, *05*, 6–6.

Dagon, D., Gu, G., Zou, C., Grizzard, J., Dwivedi, S., Lee, W., & Lipton, R. (2007). A Taxonomy of Botnet Structures. In *IEEE Computer Security Applications Conference. ACSAC 2007.*

Dittrich, D. & Dietrich, S. (2007). Command and control structures in malware: From Handler/Agent to P2P. *USENIX*, 32(6).

Duan, Z. (2006). Behavioral characteristics of spammers and their network reachability properties. In Proceedings of *IEEE International Conference on Communications, ICC 2007, (pp 164-171).*

Erbacher, R., Cutler, A., Banerjee, P., & Marshall, J. (2008). A Multi-Layered Approach to Botnet Detection. In Proceedings of the *2008 World congress in computer science, computer engineering and applied computing, The 2008 International Conference on Security and Management*.

Erman, J., Arlitt, M., & Mahanti, A. (2006). Traffic classification using clustering algorithms. In Proceedings of *the 2006 SIGCOMM workshop on Mining network data, MineNet '06,* (pp. 281-286), New York, NY, USA. ACM.

Gao, Y., Zhao, Y., Schweller, R., Venkataraman, S., Chen, Y., Song, D., & Kao, M.-Y. (2007). Detecting stealthy spreaders using online outdegree histograms. In Proceedings of the *Fifteenth IEEE International Workshop on Quality of Service,* (pp. 145-153).

Gu, G., Sharif, M., Qin, X., Dagon, D., Lee, W., & Riley, G. (2004). Worm Detection, Early Warning and Response Based on Local Victim Information. In *Proceedings of the AnnualComputer Security Applications Conference*, (pp. 136-145).

Husna, H., Phithakkitnukoon, S., Palla, S., & Dantu, R. (2008). Behavior analysis of spam botnets. In Proceedings of the *3rd International Conference on Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008.* (pp. 246-253).

Jacob, G., Debar, H., & Filiol, E. (2008). Behavioral detection of malware: from a survey towards an established taxonomy. *Journal in Computer Virology*, *4*(3), 251–266. doi:10.1007/s11416-008-0086-0

Kiayias, A., Neumann, J., Walluck, D., & McCusker, O. (2009). A combined fusion and data mining framework for the detection of botnets. In Proceedings of the *Conference For Homeland Security*. CATCH '09. Cybersecurity Applications & Technology, (pp. 273-284).

Kugisaki, Y., Kasahara, Y., Hori, Y., & Sakurai, K. (2007). Bot detection based on traffic analysis. In Proceeding of the *2007 International Conference on Intelligent Pervasive Computing. IPC.* (pp. 303-306).

Lim, S., & Jones, A. (2008). Network anomaly detection system: The state of art of network behaviour analysis. In Proceedings of the *International Conference on Convergence and Hybrid Information Technology. ICHIT '08* (pp. 459-465).

Lu, W., & Ghorbani, A. (2008). Detecting IRC Botnets on Network Application Communities. In *Fifth Annual Research Exposition* (pp. 57–57). Fredericton, New Brunswick, Canada: Faculty of Computer Science UNB.

Passerini, E., Paleari, R., Martignoni, L., & Bruschi, D. (2008). Fluxor: Detecting and monitoring fast-flux service networks. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, Lecture Notes in Computer Science, (pp. 186-206). Springer Berlin / Heidelberg.

Strayer, W. T., Walsh, R., Livadas, C., & Lapsley, D. (2006). Detecting botnets with tight command and control. In Proceedings of *the 31st IEEE Conference on Local Computer Networks*, LCN, (pp. 15-16).

Sung, A., & Mukkamala, S. (2003). Identifying important features for intrusion detection using support vector machines and neural networks. In Proceedings of the *Symposium on Applications and the Internet, 2003.* (pp. 209-216).

Wang, B., Li, Z., Tu, H., Hu, Z., & Hu, J. (2009). Actively measuring bots in peer-to-peer networks. In Proceedings of the *International Conference on Networks Security, Wireless Communications and Trusted Computing. NSWCTC '09*, volume 1, (pp. 603-607).

Wang, P., Wu, L., Aslam, B., & Zou, C. (2009). A systematic study on peer-to-peer botnets. In Proceedings of *18th Internatonal Conference on Computer Communications and Networks. ICCCN 2009.*(pp. 1-8).

Yu, J., Li, Z., Hu, J., Liu, F., & Zhou, L. (2009a). Structural robustness in peer to peer botnets. In Proceedings of the *International Conference on Networks Security, Wireless Communications and Trusted Computing, NSWCTC '09,* volume 2, (pp. 860-863).

Yu, J., Li, Z., Hu, J., Liu, F., & Zhou, L. (2009b). Using simulation to characterize topology of peer to peer botnets. In Proceedings of the *International Conference on Computer Modeling and Simulation, ICCMS '09.*, (pp. 78-83).

KEY TERMS AND DEFINITIONS

Bot: Infected computer, member of a botnet.

Botnet: Coordinated group of infected computers externally managed by an attacker.

Command and Control (C&C) Server: These are the servers used by the botmaster to communicate and remotely control bots, who report back to these servers periodically.

Dips: Amount of unique destination IP address seen in a time frame.

Dports: Amount of unique destination ports seen in a time frame.

Network Behavior: Group of periodic patterns extracted from different network characteristics.

Sips: Amount of unique source IP address seen in a time frame.

Time Window: Period of time in which TCP flows characteristics are aggregated. This information is stored in a four dimensional vector.