A Scalable System for Sneaking P2P Botnet Detection

Shruthi S. H
M.Tech, CNE student
T.John Institute Of Technology
Banglore, India

Anitha B
Assistant Professor, Dept. of CSE
T. John Institute of Technology
Bangalore, India

***Abstract* - Peer-to-peer (P2P) botnets have recently been adopted by botmasters for their resiliency against take-down efforts. Besides being harder to take down, modern botnets tend to be stealthier in the way they perform malicious activities, making current detection approaches ineffective. In addition, the rapidly growing volume of network traffic calls for high scalability of detection systems. In this paper, a novel scalable botnet detection system is proposed which is capable of detecting sneaking P2P botnets. The system first identifies all hosts that are likely engaged in P2P communications. It then derives statistical fingerprints to profile P2P traffic and further distinguish between P2P botnet traffic and legitimate P2P traffic. The parallelized computation with bounded complexity makes scalability a built-in feature of our system. Extensive evaluation has demonstrated both high detection accuracy and great scalability of the proposed system**

Keywords : Botnet, P2P, intrusion detection, network security

1. INTRODUCTION

A BOTNET is a collection of compromised hosts (a.k.a bots) that are remotely controlled by an attacker (the botmaster) through a command and control (C&C) channels. Botnets serve as the infrastructures responsible for a variety of cyber crimes , such as spamming, distributed denial-of-service (DDoS) attacks, identity theft, click fraud, etc. The C&C channel is an essential component of botnet because botmasters rely on C&C channel to issue commands to their bots and receive information from the compromised machines. Botnets may structure their C&C channels in different ways. In a centralized architecture, all bots in a botnet contact one (or a few) C&C server(s) owned by the botmaster. However, a fundamental disadvantage of centralized C&C servers is that they represent a single point of failure. In order to overcome this problem, botmasters have recently started to build botnets with a more resilient C&C architectures, using a peer-to-peer (P2P) structure or hybrid P2P/ centralized C&C structures. Bots belonging to a P2P botnet form an overlay network in which any of the nodes can be used by botmaster to distribute commands to the other peers or collect information from them. P2P botnets offer higher resiliency against take down efforts (Ex By law enforcement) , since even if a significant portion of bots in P2P botnet are disrupted , the remaining bots may still be able to communicate with each other and with the botmaster. A novel scalable botnet detection system capable of detecting stealthy P2P botnets. We refer to a stealthy P2P botnet as a P2P botnet whose malicious activities may not be observable in the network traffic. Particularly , our system aims to detect stealthy P2P botnet even if P2P botnet traffic is over lapped with traffic generated by legitimate P2P applications (skype) running on the same compromised host and achieve high scablility Our system identifies P2P bots within a monitored network by detecting the C&C communication patterns that characterize P2P botnets, regardless of how they perform malicious activities in response to botmaster's commands. Specifically, it derives statistical fingerprints of the P2P communications generated by P2P hosts and leverages them to distinguish between hosts that are part of legitimate P2P networks and P2P bots.

2. PROBLEM DEFINITION

A few approaches capable of detecting P2P botnets have been proposed [7]–[9], [12]–[14]. Compared with the existing methods [7]–[9], the design goals of our approach are different
in that: 1) our approach does not assume that malicious activities are observable, unlike [7]; 2) our approach does not require any botnet-specific information to make the detection, unlike [9]; 3) our approach needs to detect the compromised hosts that run both P2P bot and other legitimate P2P applications at the same time, unlike [8]; and 4) different from [7]–[9], our approach has high scalability as a built-in feature. Other methods [12]–[14] use machine learning for detection, which require labeled P2P botnet data to train a statistical classifier. Unfortunately, acquiring such information is a challenging task, thereby drastically limiting the practical use of these methods. To achieve the aforementioned design goals, our system includes multiple components. The first one is a *flowclustering*- based analysis approach to identify hosts that are mostly likely running P2P applications. In contrast to existing approaches of identifying hosts running P2P applications [15]–[19], our approach differs in the following ways:
1)unlike [16], our approach does not need any content signature because encryption will make content signature useless;
2) our approach does not rely on any transport layer heuristics (e.g., fixed source port) used by [15], [17], which can be easily violated by P2P applications;
3) we do not need training data set to build a machine learning based model as used in [18], because it is very challenging to get traffic of P2P botnets before they are detected; 4) in contrast to [19], our approach can detect and profile various P2P applications rather than identifying a specific P2P application (e.g., Bittorrent); and
5) our analysis approach can estimate the active time of a P2P application, which is critical for botnet detection.

3. SYSTEM DESIGN

System Overview : A P2P botnet relies on a P2P protocol to establish a C&C channel and communicate with the botmaster. Therefore P2P bots exhibit some network traffic patterns that are common to other P2P client applications (either legitimate or malicious). Thus, the system into two phases. In the first phase, the aim is to detect all hosts within the monitored network that engage in P2P communications. As shown in Figure 1, raw traffic collected is analyzed at the edge of the monitored network and apply a pre-filtering step to discard

Table .1
Notations and Descriptions

Notation	Description
T_{p2p}	The acive time of P2P application
No-DNS Peers	The percentage of flows associated with no domain names
N_{clust}	The number of clusters left by enforcing Θ_{bgp} and Θ_{p2p}
N_{bgp}	The largest number of unique bgp prefixes in one cluster
$T^{\wedge}_{p2p}$	The estimated active time of p2p application

discard network flows that are unlikely to be generated by P2P applications. Then the remaining traffic is analyzed to extract a number of statistical features to identify flows generated by P2P clients. In the second phase, the system analyzes the traffic generated by the P2P clients and classifies them into either legitimate P2P clients or P2P bots. Specifically, the active time of a P2P client is invesigated and it is identified as a candidate P2P bot if it is persistently active on the underlying host. Further the overlap of peers contacted by two candidate P2P bots is analyzed to finalize detection.

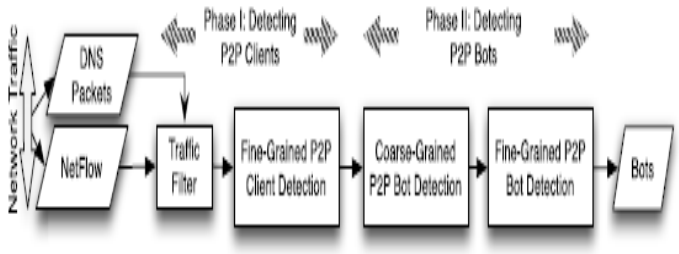

Fig 1: System Overview

A.Identifying P2P Clients

Traffic Filter: The Traffic Filter component aims at filtering out network traffic that is unlikely to be related to P2P communications. This is accomplished by passively analyzing DNS traffic, and identifying network flows whose destination IP addresses were previously resolved in DNS responses. Specifically, the following feature is leveraged: P2P clients usually contact their peers directly by looking up IPs from a routing table for the overlay network, rather than resolving a domain name. This feature is supported by Table2 (No-DNS Peers), which illustrates that the vast majority of flows generated by P2P applications do not have destination IP resolved from domain names. The remaining small fraction of flows are corresponding to a possible exception that a peer bootstraps into a P2P network by looking up domain names that resolve to stable super-nodes. Since most non-P2P applications (e.g., browsers, email clients, etc.) often connect to a destination address resulting from domain name resolution, this simple filter can eliminate a very large percentage of non- P2P traffic, while retaining the vast majority of P2P communications.

Fine-Grained Detection of P2P Clients: This component is responsible for detecting P2P clients by analyzing the remaining network flows after the Traffic Filter component. For each host h within the monitored network we identify two. flow sets, denoted as $S_{tcp}(h)$ and $S_{udp}(h)$, which contain the flows related to successful outgoing TCP and UDP connection, respectively

Table 2
Measurement of features

Trace	T_{p2p}	No-DNS Peers	N_{clust}	N_{bgp}	$T^{\wedge}_{p2p}$
T-Bittorent	24hr	96.85%	7	12857	24hr
T-Emule	24hr	99.99%	8	1133	24hr
T-Limwire	24hr	99.97%	36	5661	24hr
T-Skype	24hr	99.93%	12	12806	24hr
T-Ares	24hr	99.99%	16	1596	24hr

The successful TCP connections are with a completed SYN, SYN/ACK, ACK handshake, and those UDP (virtual) connections for which there was at least one “request” packet and a consequent response packet. In order to detect P2P clients, the fact that each P2P client frequently exchanges control messages (e.g., ping/pong messages) with other peers is considered. Besides, characteristics of these messages, such as the size and frequency of the exchanged packets, are similar for nodes in the same P2P network, and vary depending on the P2P protocol and network in use. As a consequence, if two network flows are generated by the same P2P application and they carry the same type of P2P control messages, they tend to share similar flow size. In addition, a P2P client will exchange control messages with a large number of peers distributed in many different networks. To identify flows corresponding to P2P control messages, first a flow clustering process is applied intended to group together similar flows for each candidate P2P node h. Given sets of flows $S_{tcp}(h)$ and $S_{udp}(h)$, each flow s characterised using a vector of statistical features $v(h) = [Pkt_s , Pkt_r , Byte_s , Byte_r]$, in which Pkt_s and Pkt_r represent the number of packets sent and received, and $Byte_s$ and $Byte_r$ represent the number of bytes sent and received, respectively. The distance between two flows is subsequently defined as the euclidean distance of their two corresponding vectors. Then a clustering algorithm is applied to partition the set of flows into a number of clusters. Each of the obtainedclusters of flows, C_j

(h), represents a group of flows with similar size. For each C_j (h), the set of destination IP addresses related to the flows in the clustersis considered, and for each of these IP's its BGP prefix is considered (using BGP prefix announcements). Finally, the number of distinct BGP prefixes related to destination IPs in a cluster bgp_j = BGP (C_j(h)), and discard those clusters of flows for which $bgp_j < \Theta_{bgp}$. The remaining cluster of flows are called as fingerprint clusters. Therefore, each host h can now be described by a set of fingerprint clusters FC(h) = {FC_1, ...FC_k }. 'h' is labeled as P2P node if FC(h) ≠ ∅, namely if h generated at least one fingerprint cluster

B. Detecting P2P Bots

Coarse-Grained Detection of P2P Bots**:** Since bots are malicious programs used to perform profitable malicious activities, they represent valuable assets for the botmaster, who will intuitively try to maximize utilization of bots. This is particularly true for P2P bots because in order to have a functional overlay network (the botnet), a sufficient number of peers needs to be always online. In other words, the active time of a bot should be comparable with the active time of the underlying compromised system. If this was not the case, the botnet overlay network would risk degenerating into a number of disconnected subnetworks due to the short life time of each single node. In contrast, the active time of legitimate P2P applications is determined by users, which is likely to be transient. For example, some users tend to use their file-sharing P2P clients only to download a limited number of files before shutting down the P2P application. In this case, the active time of the legitimate P2P application may be much shorter compared to the active time of the underlying system. It is worth noting that some users may run certain legitimate P2P applications for as long as their machine is on. For example, Skype is a popular P2P application for instant messaging and voice-over-IP (VoIP) that is often setup to start after system boot, and that keeps running until the system is turned off. Therefore, such Skype clients (or other "persistent" P2P clients) will not be filtered out at this stage. Hence, the first component in the "Phase II" of our system ("Coarse-Grained Detection of P2P Bots") aims at identifying P2P clients that are active for a time T_{P2P} close to the active time Tsys of the underlying system they are running on. While this behavior is not unique to P2P bots and may be representative of other P2P applications(e.g., Skype clients that run for as long as a machine is on), identifying persistent P2P clients takes us one step closer to identifying P2P bots. To estimate Tsys proceed as follows. For each host h ∈ H that we identified as P2P clients according to we consider the timestamp t_{start} (h) of the first network flow we observed from h and the timestamp t_{end} (h) related to the last flow we have seen from h. Afterwards, divide the time t_{end} (h)− t_{start} (h) into w epochs (e.g., of one hr each), denoted as T = [t_1, ... t_i . , t_w]. We further compute a vector A(h, T) = [a_1, ... a_i, . ., a_w] where a_i is equal to 1 if h generated any network traffic between t_{i-1} and t_i . Then the active time of h is estimated as Tsys =$\sum_{i=1}^{w} a_i$. In order to estimate the active time of a P2P application, obtained fingerprint clusters can be leveraged. It is because that a P2P application periodically exchanges network control (e.g., ping/pong) messages with other peers as long as the P2P application is active. For each host h (again, only the hosts in H is considered, which were previously identified as P2P clients), the set of its fingerprint clusters FC(h)={FC1,..., FCj..., FCk } is examined. Based on the flows belonging to a fingerprint cluster FC_j , we use the same approach of computing Tsys to calculate its active time, denoted as T (FC_j). Then, the active time (T_{P2P}) of a P2P application is estimated as $T\hat{}_{P2P}$ = max(T (FC_1),..T (FC_j), ... T (FC_k)). If the ratio r (h) = $T\hat{}_{P2P}$ / Tsys > Θ_{P2P}, we say that h is running a persistent P2P application, and add it to a set P of candidate P2P bots. Host h will then be input to next step, where h will be represented by a set of persistent fingerprint clusters for h,denoted as FC_p(h) ={FC_1,. . FC_i , .FC_j. . . FC_k } where T (FC_i) /T_{sys} > Θ_{P2P} for any FC_i ∈ FC_p(h).

Fine-Grained Detection of P2P Bots**:** The objective of this component is to identify P2P bots from all persistent P2P clients (i.e., set P). A feature is leveraged: the overlap of peers contacted by two P2P bots belonging to the same P2P botnet is much larger than that contacted by two clients in the same legitimate P2P network. Assume that two hosts in the monitored network, say h_A and h_B, are running the same legitimate P2P file-sharing application (e.g., Emule). Users of these two P2P clients will most likely have uncorrelated usage patterns. It is reasonable to assume that in the general case the two users will search for and download different content (e.g., different media files or documents) from the P2P network. This translates into a divergence between the set of IP addresses contacted by hosts h_A and h_B. The reason is that the two P2P clients will tend to exchange P2P control messages (e.g., ping/pong and search requests) with different sets of peers which "own" the content requested by their users, or peers that are along the path towards the content. On the contrary, if h_A and h_B are compromised with P2P bots, one common characteristic of bots is that they need to periodically search for commands published by the botmaster. This typically translates into a convergence between the set of IPs contacted by h_A and h_B. In order to leverage this feature, each host h ∈ P is represented using its persistent fingerprint clusters) is the average number of bytes sent (received) per flow in FC_i . $\prod_i$ is a set that contains the destination IP addresses (peers) of the flows in FC_i .Further two distance functions are defined below, where $FC_i^{(a)}$ and $FC_j^{(b)}$ represent fingerprint clusters from two persistent P2P clients, h_a and h_b , respectively.

- $d_{I\,Ps}\,(FC_i^{(a)}, FC_j^{(b)}) = 1 - \frac{|\prod_i^a \cap \prod_j^b|}{|\prod_i^a \cup \prod_j^b|}$

If two P2P clients (say h_a and h_b) belong to the same P2P network, regardless of a legitimate P2P network or a P2P botnet network, these two clients will follow the same implementation of the identical P2P protocol. Hence, the network flows corresponding to the same type of P2P control messages (e.g., ping/pong messages) will exhibit similar flow sizes across P2P clients running the same P2P application. Since a fingerprint cluster summarizes network flows for the same type of control messages in one client, two fingerprint clusters corresponding to the same P2P control messages belonging to the same P2P application will have similar flow

size. In other words, two P2P clients from the same P2P network will share at least one pair of fingerprint clusters $FC_i^{(a)}$ and $FC_j^{(b)}$ which have a small value of $d_{bytes}(FC_i^{(a)}, FC_j^{(b)})$ since they are corresponding to the same P2P control message. Otherwise, if two P2P clients belong to different P2P networks, d_{bytes} tends to be large. Given two P2P bots (say h_a and h_b) belonging to the same botnet, the sets of peers contacted by these two bots, denoted as , will share a large overlap, thereby generating a small value of d_{IPs} $FC_i^{(a)}$ and $FC_j^{(b)}$ Otherwise, if two P2P clients belong to i) the same legitimate P2P network or ii) different P2P networks, they will share a small overlap and produce a large value of d_{IPs} $FC_i^{(a)}$ and $FC_j^{(b)}$ A distance function dist (h_a, h_b) is defined to quantify the similarity of two P2P clients by integrating d_{bytes} and dI Ps. dist (h_a, h_b) tends to yield a small value if h_a and h_b are infected with bots from the same P2P botnet. Especially, even if h_a and h_b are infected with P2P bots from the same botnet and they run legitimate P2P applications simultaneously, the distance quantified by dist (h_a, h_b) will be small. It is because that at least one pair of fingerprint clusters that are generated by P2P bots will yield small values for both d_{bytes} and d_{IPs}.

- $dist(h_a, h_b) = \min_{i,j} \left(\lambda \times \frac{d_{bytes}(FC_i^{(a)}, FC_j^{(b)}) - \min_B}{\max_B - \min_B} + (1-\lambda) \times d_{IPs}(FC_i^{(a)}, FC_j^{(b)})\right)$

Where,

$FC_k^{(X)}$ is the k-th fingerprint cluster of host h_x
$\min_B = \min_{i,j} d_{bytes}(FC_i^{(a)}$ and $FC_j^{(b)})$
$\max_B = \max_{i,j} d_{bytes}(FC_i^{(a)}$ and $FC_j^{(b)})$
λ is a predefined constants, which we set to $\lambda = 0.5$.

After computing the distance between each pair of hosts (i.e., hosts in set **P**), a hierarchical clustering is applied, and hosts are grouped together according to the distance defined above. In practice the hierarchical clustering algorithm will produce a dendrogram (a tree-like data structure). The dendrogram expresses the "relationship" between hosts. The closer two hosts are, the lower they are connected at in the dendrogram. Two P2P bots in the same botnet should have small distance and thus are connected at lower level (forming a dense cluster). In contrast, legitimate P2P applications tend to have large distances and consequently are connected at the upper level. Then hosts in dense clusters are classified as P2P bots, and discard all other clusters and the related hosts, which we classify as legitimate P2P clients. In practice, we cut the dendrogram at Θ_{bot} ($\Theta_{bot} \in [0, 1]$) of the maximum dendrogram height ($\Theta_{bot} \in$ heightmax). To set Θ_{bot} , its assumed that: a) there is no labeled data set of botnet traffic; b) the distance between two legitimate P2P applications is much larger than that between two bots belonging to the same botnet Therefore, we conservatively set $\Theta_{bot} = 0.95$.

4. SYSTEM IMPLEMENTATION

The implementation objective is to integrate high scalability as a built-in feature into our system. To this end, we first identify the performance bottleneck of our system and then mitigate it using complexity reduction and parallelization.

A. Performance Bottleneck

Out of four components in our system, "Traffic Filter" and "Coarse-Grained Detection of P2P Bots" have linear complexity since they need to scan flows only once to identify flows with destination addresses resolved from DNS queries or calculate the active time. Other two components, "Fine-Grained Detection of P2P Clients" and "Fine-Grained P2P Detection of P2P Bots", require pairwise comparison for distance calculation. Specifically, if we denote the number of flows generated by a host as *n* and the number of hosts as *S*, the time complexity of Fine-Grained Detection of P2P Clients approximates $O(S^{n2})$. Comparably, if we denote the number of persistent P2P clients as l, the time complexity of *Fine*-Grained P2P Bot Detection approximates $O(l^2)$. Since the number of flows generated by network applications (i.e., *n*) could be enormous (e.g., more than hundreds of thousands of flows are generated by a single P2P client in our experiments), the computation overhead of Fine-Grained Detection of P2P Clients may become prohibitive. On contrary, the percentage of P2P clients in the ISP network is relatively small (e.g., 3%-13% as reported in). Consequently, Fine- Grained P2P Bot Detection is unlikely to introduce huge performance overhead. For instance, given a typical ISP network or a large enterprise network that has 65,536 hosts (/16 subnet), if we assume that 8% hosts run P2P applications and conservatively assume that half of them are persistent, the number of persistent P2P clients (i.e., l) subject to analysis by Fine-Grained P2P Bot Detection is 2,221, incurring negligible overhead. To summarize, "Fine-Grained P2P Client Detection" is the performance bottleneck.

B. Two-Step Flow Clustering

We use a two-step clustering approach to reduce the time complexity of "Fine-Grained P2P Client Detection". For the first-step clustering, we use an efficient clustering algorithm to aggregate network flows into K sub-clusters, and each subcluster contains flows that are very similar to each other. For the second-step clustering, we investigate the global distribution of sub-clusters and further group similar sub-clusters into clusters. The distance of two flows is defined as the Euclidean distance of their corresponding vectors, where each vector [Pkt_s , Pkt_r , $Byte_s$, $Byte_r$] represents the number of packets/bytes that are sent/received in a flow. In our original design , we have adopted , a streaming clustering algorithm. The number of clusters generated by BIRCH is mainly decided by a predefined parameter R, which quantifies the radius of a cluster. A greater value of R implies less clusters. Although BIRCH can perform approximate clustering of an arbitrarily large dataset given constrained memory space by scanning the dataset only once, estimating K from R remains a challenging task. To partially address this challenge in our original design, we adopted an empirical way: we start from a small R value (e.g., R = 0) and gradually

increase it by δ until K clusters are generated. Since the number of clusters generated by BIRCH is sensitive to R, δ has to be very small to assure that R is not overlarge. As a result, a huge number of iterations have to be explored until we find appropriate R
that yields K sub-clusters. This procedure results in a large amount of computation time. In the current design, we employ K-means as the first step clustering. The main reason is that K-Means can achieve bounded time complexity O(Nk I), where K explicitly indicates the number of expected clusters, n is the number of flows for each host, and I is the maximum number of iterations. For the second-step clustering, we use hierarchical clustering to group sub-clusters into clusters. Each sub-cluster is represented using a vector ([Pkt_s , Pkt_r , $Byte_s$, $Byte_r$]), which is essentially the average for all flow vectors in this sub-cluster.

C. System Parallelization

Since the two-step clustering analyzes network flows for each single host, we can parallelize the computation for all hosts. We formulate the problem as follows: given S hosts denoted as H = {h_1, h_2, . . . h_S} and M computation nodes denoted as C = {c_1, c_2, . . . c_M}, we partition H into M exclusive subsets HT_1, HT_2..HT_M and assign HT_i to c_i for analysis, whose processing time is denoted as exc(c_i ,HT_i). Our target is to design a partition algorithm so that the overall processing time, denoted as T = max(exc(c_i, HT_i)), is minimized. If we assume each computation node has the same capacity, T will be minimized when the analysis workload is evenly distributed across all computation nodes.

5. CONCLUSION

In this paper, we presented a novel botnet detection system that is able to identify *stealthy* P2P botnets, whose malicious activities may not be observable. To accomplish this task, we derive statistical fingerprints of the P2P communications to first detect P2P clients and further distinguish between those that are part of legitimate P2P networks (e.g., filesharing networks) and P2P bots. We also identify the performance bottleneck of our system and optimize its scalability. The evaluation results demonstrated that the proposed system accomplishes high accuracy on detecting stealthy P2P bots and great scalability.

REFERENCES

[1] S. Stover, D. Dittrich, J. Hernandez, and S. Dietrich, "Analysis of the storm and nugache trojans: P2P is here," in *Proc. USENIX*, vol. 32. 2007, pp. 18–27.

[2] P. Porras, H. Saidi, and V. Yegneswaran, "A multi perspective analysis of the storm (peacomm) worm," Comput. Sci. Lab., SRI Int., Menlo Park, CA, USA, Tech. Rep., 2007. [3] P. Porras, H. Saidi, and V. Yegneswaran. (2009). *Conficker C Analysis* [Online].
Available: http://mtc.sri.com/Conficker/addendumC/index.html

[4] G. Sinclair, C. Nunnery, and B. B. Kang, "The waledac protocol: The how and why," in *Proc. 4th Int. Conf. Malicious Unwanted Softw.*, Oct. 2009, pp. 69–77.

[5] R. Lemos. (2006). *Bot Software Looks to Improve Peerage* [Online].
Available: http://www.securityfocus.com/news/11390 [6] Y. Zhao, Y. Xie, F. Yu, Q. Ke, and Y. Yu, "Botgraph: Large scale spamming botnet detection," in *Proc. 6th USENIX NSDI*, 2009, pp. 1–14.

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

[8] T.-F. Yen and M. K. Reiter, "Are your hosts trading or plotting? Telling P2P file-sharing and bots apart," in *Proc. ICDCS*, Jun. 2010, pp. 241–252.

[9] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, and N. Borisov, "BotGrep: Finding P2P bots with structured graph analysis," in *Proc. USENIX Security*, 2010, pp. 1–16.

[10] J. Zhang, X. Luo, R. Perdisci, G. Gu, W. Lee, and N. Feamster, "Boosting the scalability of botnet detection using adaptive traffic sampling," in *Proc. 6th ACM Symp. Inf., Comput. Commun. Security*, 2011, pp. 124–134.

[11] J. Zhang, R. Perdisci, W. Lee, U. Sarfraz, and X. Luo, "Detecting stealthy P2P botnets using statistical traffic fingerprints," in *Proc. IEEE/IFIP 41st Int. Conf. DSN*, Jun. 2011, pp. 121–132.

[12] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, *et al.*, "Detecting P2P botnets through network behavior analysis and machine learning," in *Proc. 9th Annu. Int. Conf. PST*, Jul. 2011, pp. 174–180.

[13] D. Liu, Y. Li, Y. Hu, and Z. Liang, "A P2P-botnet detection model and algorithms based on network streams analysis," in *Proc. IEEE FITME*, Oct. 2010, pp. 55–58.

[14] W. Liao and C. Chang, "Peer to peer botnet detection using data mining scheme," in *Proc. IEEE Int. Conf. ITA*, Aug. 2010, pp. 1–4.

[15] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: Multilevel traffic classification in the dark," in *Proc. ACM SIGCOMM*, 2005, pp. 229–240.

[16] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in-network identification of P2P traffic using application signatures," in *Proc. 13th ACM Int. Conf. WWW*, 2004, pp. 512–521.

[17] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy, "Transport layer identification of P2P traffic," in *Proc. 4th ACM SIGCOMM Conf. IMC*, 2004, pp. 121–134.

[18] A. W. Moore and D. Zuev, "Internet traffic classification using Bayesian analysis techniques," in *Proc. ACM SIGMETRICS*, 2005, pp. 50–60.

[19] M. P. Collins and M. K. Reiter, "Finding peer-to-peer file sharing using coarse network behaviors," in *Proc. 11th ESORICS*, 2006, pp. 1–17.

[20] D. Stutzbach and R. Rejaie, "Understanding churn in peer-topeer networks," in *Proc. 6th ACM SIGCOMM Conf. IMC*, 2006, pp. 189–202.

[21] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. Freiling, "Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm," in *Proc. USENIX LEET*, 2008, pp. 1–9.

[22] G. Bartlett, J. Heidemann, C. Papadopoulos, and J. Pepin, "Estimating P2P traffic volume at USC," USC/Information Sciences Institute, Los Angeles, CA, USA, Tech. Rep. ISI-TR-2007-645, 2007.

[23] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," in *Proc. ACM SIGMOD*, 1996, pp. 103–114.

[24] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *J. Intell. Inf. Syst.*, vol. 17, nos. 2–3, pp. 107–145, 2001.

[25] (2011). *Argus: Auditing Network Activity* [Online]. Available: http://www.qosient.com/argus/

[26] Z. Li, A. Goyal, Y. Chen, and A. Kuzmanovic, "Measurement and diagnosis of address misconfigured P2P traffic," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[27] (2011). *Autoit Script* [Online]. Available: http://www.autoitscript.com/autoit3/index.shtml

[28] (2011). *Zeus Gets More Sophisticated Using P2P Techniques* [Online]. Available: http://www.abuse.ch/?p=3499

[29] A. Binzenhofer, D. Staehle, and R. Henjes, "On the stability of chordbased P2P systems," in *Proc. IEEE Global Telecommun. Conf.*, vol. 2. Nov./Dec. 2005, pp. 884–888.

[30] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a DHT," in *Proc. Annu. Conf. USENIX Annu. Tech. Conf.*, 2004, pp. 127–140.

[31] D. Dagon, G. Gu, C. Lee, and W. Lee, "A taxonomy of botnet structures," in *Proc. 33rd Annu. Comput. Security Appl. Conf.*, 2007, pp. 325–339.

[32] (2010). *Resilient Botnet Command and Control with Tor* [Online]. Available: http://www.defcon.org/images/defcon-18/dc-18-presentations/D.Brown/DEFCON-1%8-Brown-TorCnC.pdf