International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE) Vol. 1, Issue 2, January 2015

Counter Attack Tool (Cat) For Tunnel Attacks

[1]S. Prasanna,[2]V.Vijayakumari
[1]Department of Computer Science & Engineering, Mailam Engineering College, Mailam, Tamil Nadu, India
[2]Computer Science and Engineering at Mailam Engineering College, Mailam, Tamil Nadu, 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 scalabil-ity of detection systems. In this paper, we propose a novel scalable botnet detection system capable of detecting stealthy P2P botnets. Our 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.**

Index Terms— **Botnet, P2P, P2P, intrusion detection, network, security.**

I. INTRODUCTION

BOTNET is a collection of compromised hosts **A**(a.k.a. bots) that are remotely controlled by an attacker (the botmaster) through a command and control (C&C) chan-nel. 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 a botnet because botmasters rely on the C&C channel to issue commands to their bots and receive information from the compromised machines. Botnets may structure their C&C channels in dif-ferent 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 architecture, using a peer-to-peer (P2P) structure [1]–[3] or hybrid P2P/centralized C&C structures [4]. Bots belonging to a P2P botnet form an overlay network in which any of the nodes (i.e., any of the bots) can be used by the botmaster to distribute commands to the other peers or collect information from them. Notable examples of P2P botnets are represented by Nugache [5], Storm [2], Waledac [4], and even Confiker, which has been shown to embed P2P capabilities [3]. Storm and Waledac are of particular interest because they use P2P C&C structures as the primary way to organize their bots. While more complex, and perhaps more costly to manage compared to centralized botnets, P2P botnets offer higher resiliency against take-down efforts (e.g., by law enforcement), since even if a significant portion of bots in a P2P botnet are disrupted the remaining bots may still be able to communicate with each other and with the

Manuscript received November 3, 2012; revised July 5, 2013 and October 24, 2013; accepted October 24, 2013. Date of publication November 11, 2013; date of current version December 16, 2013. This work was supported in part by the Research Initiation Grant at Wright State University under Grant 282040, in part by the GRF PolyU 5389/13E, NSF under Grant CNS-1149051 and Grant 0831300, in part by the Department of Homeland Security under Contract FA8750-08-2-0141, and in part by the Office of Naval Research under Grant N000140710907 and Grant N000140911042. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. C.-C. Jay Kuo.

J. Zhang is with the Department of Computer Science and Engi-neering, Wright State University, Dayton, OH 45435 USA (e-mail: junjie.zhang@wright.edu).

R. Perdisci is with the Department of Computer Science, University of Georgia, Athens, GA 30602 USA (e-mail: perdisci@cs.uga.edu).

W. Lee is with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: wenke@cc.gatech.edu).

X. Luo is with the College of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: csluo@comp.polyu.edu.hk).

Sarfraz was with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA. She is now with Cisco, San Jose, CA 95134 USA (email: usarfraz@cisco.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2013.2290197

botmaster.

Detecting botnets is of great importance. However, design-ing an effective P2P-botnet detection system is faced with sev-eral challenges. First, the P2P file-sharing and communication applications, such as Bittorrent, emule, and skype, are very popular and hence C&C traffic of P2P botnets can easily blend into the background P2P traffic. This challenge is further compounded by the fact that a bot-compromised host may exhibit mixed patterns of both legitimate and botnet P2P traffic (e.g., due to the coexistence of a file-sharing P2P application and a P2P bot on the same host). Second, modern botnets tend to use increasingly stealthy ways to perform malicious activities that are extremely hard to be observed in the network traffic. For example, some botnets send spam through large popular webmail services such as Hotmail [6], which is likely transparent to network detectors due to encryption and overlap with legitimate email use patterns. Third, as the vol-ume of network traffic grows rapidly, the deployed detection system is required to process a huge amount of information efficiently. To date, a few approaches capable of detecting P2P botnets have been proposed [7]–[9]. However, these approaches can-not address all the aforementioned challenges. For example, BotMiner [7] identifies a group of hosts as bots belonging to the same botnet if they share similar communication patterns and meanwhile perform similar malicious activities, such as scanning, spamming, exploiting, etc. Unfortunately, the malicious activities may be stealthy and non-observable, thereby making BotMiner ineffective. In addition, BotMiner's scalability is significantly constrained [10]. Yen et al. [8] proposed an algorithm that aims to distinguish between hosts that run legitimate P2P file sharing applications and P2P bots. Nevertheless, this algorithm [8] does not take into account the fact that a bot may coexist with a legitimate P2P application on the same host. As a consequence, the mixed traffic profile of the compromised host may disguise the communication patterns related to the bot, rendering the algorithm [8] inef-fective. BotGrep [9] analyzes network flows collected over multiple large networks (e.g., ISP networks), and attempts to detect P2P botnets by analyzing the

In this paper, we present 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 overlapped with traffic generated by legitimate P2P applications (e.g., Skype) running on the same compromised host and ii) achieve high scalability. To this end, our system identifies P2P bots within a monitored network by detect-ing the C&C communication patterns that characterize P2P botnets, regardless of how they perform malicious activities in response to the 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 (e.g., file-sharing networks) and P2P bots. The high scalability of our system stems from the parallelized computation with bounded computational complexity. To summarize, our work makes the following contributions:

A A new *flow-clustering*-based analysis approach to iden-tify hosts that engage in P2P communications.

B An efficient algorithm for P2P traffic *profiling*, where we build statistical fingerprints to profile various P2P applications and estimate their *active time*.

C A P2P botnet detection method that can effectively detect stealthy P2P bots even if the P2P botnet traffic is overlapped with traffic generated by legitimate P2P applications (e.g., Skype) running on the same compro-mised machine.

D A scalable design based on an efficient detection algo-rithm and parallelized computation.

A prototype system and extensive evaluation based on real-world network traffic, which has demonstrated high detection accuracy (i.e., a detection rate of 100% and 0.2% false positive rate) and great scalability (i.e., processing 80 million flows in 0.8 hour) of our design.

Compared to our preliminary version of this work [11], we have made the following substantial improvements. First, we simplified the system design by eliminating a coarse-grained analysis component for P2P client detection based on failed network connections without sacrificing its detection performance. The new design eradicates the necessity of keeping failed connections, reducing the storage cost by 60%. Second, we redesigned the clustering-based P2P client detec-tion algorithm to enhance its efficiency, which decreases the processing time by at least 68% compared to the original design. Third, we parallelize our system to boost its scalability and demonstrated its efficiency. Finally, we have empirically evaluated the extent to which configurable parameters

affect the detection performance and manifested that our system is effective over a large range of parameter values.

II. RELATED WORK

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 applica-tions 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 *flow-clustering*-based analysis approach to identify hosts that are mostly likely running P2P applications. In contrast to exist-ing approaches of identifying hosts running P2P applica-tions [15]–[19], our approach differs in the following ways:

1) unlike [16], our approach does not need any content signa-ture 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.

III. 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, we divide our systems into two phases. In the first phase, we aim at detecting all hosts within the monitored network that engage in P2P communications. As shown in Figure 1, we analyze raw traffic collected at the edge of the monitored network and apply a pre-filtering step to discard network flows that are unlikely to be generated by P2P

Fig. 1. System overview.

TABLE I

Notations and Descriptions

notation	Description
T_{p2p}	the active 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'_{p2p}	the estimated active time for P2P application

TABLE II

Measurement of Features

Trace	T_{p2p}	*No-DNS Peers*	N_{clust}	N_{bgp}	T'_{p2p}
T-Bittorrent	24hr	96.85%	17	12857	24hr
T-Emule	24hr	99.99%	8	1133	24hr
T-Limewire	24hr	99.97%	36	5661	24hr
T-Skype	24hr	99.93%	12	12806	24hr
T-Ares	24hr	99.99%	16	1596	24hr

applications. We then analyze the remaining traffic and extract a number of statistical features to identify flows generated by P2P clients. In the second phase, our system analyzes the traffic generated by the P2P clients and classifies them into either *legitimate* P2P clients or P2P *bots*. Specifically, we investigate the active time of a P2P client and identify it as a *candidate* P2P bot if it is persistently active on the underlying host. We further analyze the overlap of peers contacted by two *candidate* P2P bots to finalize detection.

To illustrate the statistical features and motivate the related thresholds used by our system, we ran five popular P2P applications, including Bittorrent, Emule, Limewire, Skype, and Ares, for 24 hours to collect their traffic traces. For the Bittorrent application, we generated two separate 24-hour traces (T-Bittorrent and T-Bittorrent-2). In this section we report a number of measurements on the obtained traffic traces to better motivate the design of statistical features, whose notations are summarized in Table I. Table II reports the feature values measured on the collected traffic traces. We now elaborate on each component of our system.

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, we leverage the following feature: 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 Table II (*No-DNS Peers*), which illustrates that the vast majority of flows gener-ated by P2P applications do not have destination IPs 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. We consider as successful those TCP connections 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.

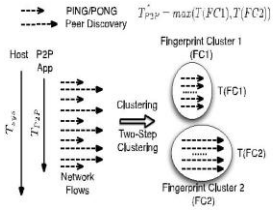
In order to detect P2P clients, we first consider the fact that each P2P client frequently exchanges control messages (e.g., ping/pong messages) with other peers. Besides, we notice that the characteristics of these messages, such as the size and frequency of the exchanged packets, are *similar* for 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. Consequently, the destination IP addresses of network flows that carry these control messages will spread across a large number of networks where each network can be represented by its BGP prefix.

To identify flows corresponding to P2P control messages, we first apply a flow clustering process intended to group together similar flows for each candidate P2P node *h*. Given sets of flows $S_{tcp}(h)$ and $S_{udp}(h)$, we characterize each flow 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. We then apply a clustering algorithm to partition the set of flows into a number of clusters. Each of the obtained clusters of flows, $C_j(h)$, represents a group of flows with similar size. For each $C_j(h)$, we consider the set of destination IP addresses related to the flows in the clusters, and for each of these IPs we consider its BGP prefix (using BGP prefix announcements).

Finally, we count 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 $bgp_j <$ $\_bgp$. We call *fingerprint clusters* the remaining cluster of flows. Therefore, each host *h* can now be described by a set of fingerprint clusters $FC(h) = \{ FC_1, \ldots, FC_k \}$. We label *h* as P2P node if $FC(h) = \emptyset$, namely if *h* generated at least one fingerprint cluster.

Figure 2 illustrates an example of the flow cluster-ing process for a P2P node. Flows corresponding to ping/pong and peer-discovery share similar sizes, and hence they are grouped into two clusters (FC_1 and FC_2), respectively. Since the number of destination BGP pre-fixes involved in each cluster is larger than $\_bgp$, we take FC_1 and FC_2 as its fingerprint clusters. A *fingerprint cluster summary*, (Pkt_s , Pkt_r , $Byte_s$, $Byte_r$, proto), repre-sents the protocol and the average number of sent/received

Th flow packets/bytes for all e s in this fingerprint clus-implemente Th analysis component ter. We d e flow (with identified fingerprint cluster for $\_bgp$ = 50) and the sampl e P2P traces including two Bittorrent traces (T-Bittorrent, T-Bittorrent-2), and one Skype trace. Figure 3 describes the distribution of flow sizes for these two Bittorrent traces and a large number of flows share similar sizes. The summaries of the fingerprint clusters are illustrated in Table III, which can successfully cluster flows with similar sizes as presented in Figure 3. Particularly, manual investigation of the flow payload has confirmed flows corresponding to two fingerprint clusters, "(1 1 145 319, UDP)" and "(1 1 109 100, UDP)", are used for node discovery and exchanging ping/pong messages, respectively.



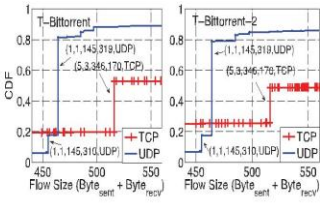


Fig. 2. Example of flow clustering *Fig.3. CDF of flow size*

to identify P2P hosts.

B. Detecting P2P Bots

Coarse-Grained Detection of P2P Bots: Since bots are malicious programs used to perform profitable malicious activ-ities, 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.

TABLE III

Summaries of Fingerprint Clusters

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 [20]. 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.

Trace	Fingerprints
T-Bittorrent	1 1 145 319, UDP 1 1 109 100, UDP 1 1 146 340, UDP 5 3 346 170, TCP 1 1 145 310, UDP
T-Bittorrent-2	1 1 145.01 317.66, UDP 1 1 109 100, UDP 1 1 146 342, UDP 5 3 346 170, TCP 2 2 466 461, UDP

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 T_{sys} 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 T_{sys} we proceed as follows. For each host $h \in \mathbf{H}$ that we identified as P2P clients according to Section IV-B, 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, we divide the time $t_{end}(h) - t_{start}(h)$ into w epochs (e.g., of one hour each), denoted as $T = [t_1, \ldots t_i, \ldots, t_w]$. We further compute a vector $A(h, T) = [a_1, \ldots a_i, \ldots, a_w]$ where a_i is equal to 1 if h generated any network traffic between t_{i-1} and t_i. We then estimate the active time of h as $T_{sys} = \sum_{i=1}^{w} a_i$.

In order to estimate the active time of a P2P application, we can leverage obtained fingerprint clusters. It is because that a P2P application periodically exchanges network con-trol (e.g., ping/pong) messages with other peers as long as the P2P application is active. For each host h (again, we consider only the hosts in **H**, which we previously identi-fied as P2P clients), we examine the set of its fingerprint clusters $FC(h) = \{ FC_1, \ldots FC_j \ldots, FC_k \}$ (see Section III). Based on the flows belonging to a fingerprint cluster FC_j, we calcu- late its active time, denoted as $T(FC_j)$. Then, we estimate the active time (T_{P2P}) of a P2P application as $\hat{T}_{P2P} = \max(T(FC_1), \ldots T(FC_j), \ldots T(FC_k))$.

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**). We leverage one feature: 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.

f 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 imple-mentation 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 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 $\_^a_i$ and $\_^b_j$, will share a large overlap, thereby generating a small value of $d_{IPs}(FC_i^{(a)}, FC_j^{(b)})$. 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 $d_{IPs}(FC_i^{(a)}, FC_j^{(b)})$.

In other words, two P2P clients from the same P2P network will share at least one pair of fingerprint clusters $(FC_i^{(a)}, FC_j^{(b)})$, 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.

IV. SYSTEM IMPLEMENTATION

The implementation objective is to integrate high scalability ability 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 $O(S*n^2)$.

For instance, given a typical ISP net-work 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 V. EVALUATION

A. Network Traces

The traffic we collected from an academic network came from a span port mirroring all traffic crossing the gateway router (around 200-300Mbps) for the college networks. We used Argus [25] to efficiently collect network flow infor-mation of the traffic between internal and external networks for one entire day. Along with various flow statistics we also recorded the first 200 bytes of each *flow payload*, which we used to identify known legitimate P2P clients within our network.

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 Fo exclusive subsets $H T_1, H T_2.. H T_M$ and assign $H T_i$ to c_i r analysis, whose processing time is denoted as $exc(c_i , H T_i)$. Our target is to design a partition algorithm so that the overall processing time, denoted as $T = max (exc(c_i , H T_i))$, 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. *C.Experimental Results on Botnet Detection*

1) DNS-Based Traffic Filter: We applied our detection system on data set D. The traffic filter drastically reduced the workload for the whole system. As indicated in Figure 4, it reduced the number of hosts subject to analysis by 67% (from 953 to 316) but retained all P2P clients.

2) Identifying and Profiling P2P Applications: Based on the traffic after the DNS-based filter, our system discovers fingerprint clusters. The discovered clusters can effectively not only identify P2P clients, but also profile P2P applications running in a host. Our system can successfully identify all 26 P2P clients in D. Examples of fingerprint cluster summaries ([Pkt_s , Pkt_r , $Byte_s$, $Byte_r$, $Proto$]) are presented in Table VII and Table VIII.

Table VII presents fingerprint cluster summaries for two Storm bots and two Waledac bots before their traffic was overlaid with legitimate P2P client traffic. In this table, we can find that bots from the same botnet share similar fingerprint clusters with respect to the flow size. We used $N ET_{Zeus}$ to evaluate whether our system can effectively profile a Zeus bot. Our system generated 6 statistical fingerprint clusters, whose summaries are illustrated in Table IX. Obtaining fingerprint clusters from this host demonstrates that our system can successfully identify a P2P application running on the host. Although a single Zeus bot is insufficient for the final component to perform detection, its fingerprint clusters offer valuable information: the identi-fied P2P application (Zeus) adopts different P2P protocols compared to known P2P applications (5 legitimate P2P appli-cations and 2 P2P botnets) since its fingerprint clusters have not been observed in our experiments.

3) Detecting P2P Bots: Among 26 P2P clients identified in the previous step, 25 out of them exhibit persistent P2P behaviors. We further evaluate the similarity of fingerprint clusters and peer IPs for each pair of persistent P2P clients and derive a dendrogram as shown in the Figure 5. As bots from the same botnet share similar fingerprint clusters and a large overlap of peer IPs, they will have small distance (*dist ()*), resulting in a dense cluster of these bots in the dendrogram. As indicated in Figure 5, all 13 Storm bots and 3 Waledac bots form two dense clusters, respectively. We conservatively cut the tree at $\_{bot} * height_{max} = 0.95$ ($\_{bot} = 0.95$ and $height_{max} = 1$). Hence, three clusters are identified and overall 18

hosts were labeled as suspicious, which include *all* 16 P2P bots and 2 false positives, resulting a high detection rate of 100% and a low false positive rate of 0.2% (2/953).

TABLE VI

Traces of Botnets

Trace	duration	size	# of bots
Waledac	24hr	1.1G	3
Storm	24hr	4.8G	13
Zeus	24hr	7.3M	1

B. Experiment Setup

We prepared a data set (D) for evaluation. Specifically, we randomly selected half (8) of the P2P bots from NET_{bots}. Then for each of the 5 P2P applications we ran, we randomly selected one out of its two traces from NET_{P2P} and overlaid its traffic to the traffic of a randomly selected host from NET_{CoC}. We further randomly chose 3 P2P hosts from

In this case, our traffic filter could eliminate bot flows. To solve this problem, we can only filter out network flows whose destination IP addresses are resolved from popular domains, i.e., domains queried by a non-negligible fraction of hosts in the monitored networks.

B. Evasion by Joining Legitimate P2P Applications

Bots may "blend" with legitimate P2P clients to evade detection by directly joining legitimate P2P networks, in which botmasters can propagate commands. In fact, the initial version of Storm adopted this "blending" strategy, while recent P2P botnets, including the most recent version of Storm, Waledac, and Zeus, build their own P2P networks. Never-theless, the "blended" P2P bots may make our detection more difficult because bots' fingerprint clusters will be identical to those of legitimate P2P applications.

C. Evasion by Limiting the Activity of P2P Bots

Since the Coarse-Grained P2P Bot Detection component leverages the feature that bots are usually persistently active on the underlying compromised hosts, botmasters may signifi-cantly reduce the active time of each P2P bot in order to evade our system. Specifically, P2P bots can repeat the following pattern: bots are active for only a short time and then silent for a long period. While this technique might be effective, it will introduce significant negative consequences to botnets. *D. Evasion by Building Topology-Aware P2P Bots*

Our system analyzes network traffic exchanged between internal hosts and external networks and it assumes that at least two bots from the same botnet. Therefore, botmasters may design topology-aware P2P botnets for evasion. To be specific, P2P bots inside the monitored network can form a P2P network whereas only one among them is responsible for communicating with external peers. In order to address this challenge, we can further improve our system. First, we can implement our system in a distributed manner, where all components except the Fine-Grained Bot Detection component can be moved "close" to end hosts to gain complete view of their network activities. Fingerprint clusters of each single P2P client can then be acquired by taking into account the traffic it exchanged with other internal hosts.

E. Evasion by Tunneling P2P Traffic Through Tor Networks

As suggested by Dennis Brown [32], botmasters could leverage Tor's hidden service to build P2P C&Cs that are extremely robust against disruption efforts. In a Tor-based P2P botnet, each peer can announce a service descriptor through Tor's hidden service with its IP address concealed.

F. Evasion by Randomizing Communication Behaviors

Bots could randomize their P2P communication patterns to prevent our system from getting accurate fingerprint clusters for P2P protocols. In this case, we can measure the number of failed connections to perform coarse-grained detection of P2P clients and also adopt more general features, such as the distribution of

flow/packet sizes and flow/packet intervals, to profile P2P applications.

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., file-sharing networks) and P2P bots.

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, 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, 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.

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.