

Are Your Hosts Trading or Plotting? Telling P2P File-Sharing and Bots Apart

Ting-Fang Yen
Carnegie Mellon University, Pittsburgh, PA
tyen@andrew.cmu.edu

Michael K. Reiter
University of North Carolina, Chapel Hill, NC
reiter@cs.unc.edu

Abstract—Peer-to-peer (P2P) substrates are now widely used for both file-sharing and botnet command-and-control. Despite the commonality of their substrates, we show that the different goals and circumstances of these applications give rise to behaviors that can be distinguished in network flow records. Using features related to traffic volume, persistence of network connections, amount of “churn” among peers, and differences between human-driven and machine-driven traffic, we develop a technique for identifying P2P bots (the Plotters) and, in particular, separating them from file-sharing hosts (the Traders). Evaluations performed on traffic recorded at the edge of a university network show that we can achieve, e.g., 87.50% detection of Storm bots with a 0.81% false positive rate. We also demonstrate the significant extent to which Plotter behaviors would need to change to evade our techniques.

I. INTRODUCTION

Peer-to-peer (P2P) networks have been used as botnet communication channels as early as 2003 [1]. The decentralized nature of these networks overcame the single-point-of-attack limitation in centralized control, making the botnet resilient to individual peer failures and also harder to detect and take down. These motivations for using P2P substrates are similar to those underlying the use of P2P protocols for file-sharing; the takedown of Napster, for example, highlighted the limitations of a centralized “command-and-control” infrastructure in that domain. It is thus not surprising that P2P substrates now commonly support both activities.

A consequence of this common use of P2P technologies is that botnet command-and-control traffic will tend to “blend into” a background of P2P file-sharing, making it difficult to separate these two types of traffic. In both cases, status information about available peers needs to be maintained constantly to ensure the connectivity of the network; peers experience a high connection failure rate due to the dynamics of nodes joining and leaving (i.e., “churn”); and peers participate in both client and server activities, simultaneously receiving and initiating connections. This commonality is punctuated by the fact that one highly publicized and well-studied

P2P botnet, Storm, built its communication protocol based on the Overnet network, whose distributed hash table implementation [2] is incorporated in both eDonkey¹ and BitTorrent² file-sharing applications.

In light of this, the primary problem facing the detection of such bots is differentiating them from other P2P hosts. In this work, we focus specifically on the problem of P2P botnet detection given this challenge. We assume the viewpoint of a network administrator who collects flow records at the border of an enterprise network, and who seeks to identify internal nodes that are P2P bots. We construct a series of tests on network traffic to separate P2P bots from P2P file-sharing hosts, to which we will refer as Plotters and Traders, respectively, throughout this paper. Our tests work exclusively on traffic summaries (e.g., flow records) with no access to individual packets (much less payloads), and so can scale to very busy networks where per-packet logging may not be cost-effective. Our techniques are thus also unaffected by encryption of bot payload contents.

Given the varied nature of malware behaviors, we focus on characteristics of the traffic that do not depend on particular attack activities performed by the infected hosts (e.g., spam forwarding, DDoS), but rather that are basic properties of Plotters that operate over P2P networks. At a high level, these characteristics include:

- **Volume:** Since Traders generally perform large multi-media file transfers (e.g., MP3, movies), but Plotters almost never do, traffic volume should be a good indicator of suspicious activity. However, as we will show in §V, examining volume alone yields many false positives.
- **Persistence:** Most Plotters are required to maintain connectivity to their peers to receive and execute commands from the botmaster. Moreover, since the Plotter cannot control when network access would be available, it is often opportunistic in communicating

¹http://wiki.amule.org/index.php/FAQ_eD2k-Kademlia

²http://bittorrent.org/beps/bep_0005.html

with peers, i.e., whenever it has a chance.

- **Peer churn:** The peer membership of a file-sharing network is very dynamic, due to peers constantly joining and leaving the network, the availability of the desired file, and connections between hosts being terminated soon after the completion of the file transfer. Previous studies [3], [4], [5] also showed that most Traders appear only once a day, and remain connected for short durations (minutes). Plotters, by contrast, are likely to experience less churn in peer membership, since each individual host maintains a list of known peers with which to communicate, and also because they are more persistent than Traders, as described above.
- **Human-driven versus Machine-driven:** Perhaps a more basic difference between Plotters and Traders is that, while file-sharing activities are mainly human-driven, Plotters are almost entirely automated. This causes much of their traffic to exhibit significant regularity and periodicity that is rarely seen among those from human activities. In a slightly different context, previous studies on distinguishing humans and bots in Internet chat rooms also observed that human behaviors are more complex than bots [6].

We construct measures of each of these characteristics, framing them into tests that distinguish Plotters from Traders. To our knowledge, our work is the first to target Plotters from the perspective of their commonality (or the lack thereof) with other P2P protocols.

We use these tests to build a technique for separating Plotters from Traders (and other hosts), and evaluate the ability of our technique to identify Plotters within traffic observed at the border of a university campus network. Our results show that Storm bots can be identified with up to 87.50% true positive rate and only 0.81% false positives, despite the fact that Traders using the *same* P2P substrate were present in our tests. We also perform tests with Nugache bots, where we show that for the same false positive rate, we can detect 30% of the bots. We will explore the reasons behind our lower — though still substantial — detection rate in this case.

A final contribution of our work is to examine how much malware behavior would need to change to evade our technique. We quantify for each of our component tests the degree to which Plotters would need to alter their behaviors to evade them. The results suggest that evading our techniques would require significant behavioral changes of existing botnets. Moreover, due to the way in which our tests are constructed, it would typically not be evident to the Plotters how much change would be sufficient to evade them.

II. RELATED WORK

Much work to date has focused on detecting the centralized command-and-control architecture utilized by early botnets [7], [8], [9], [10], [11], [12]. But as malware increasingly takes advantage of peer-to-peer networks as their main communication channel, i.e., the Plotters, these approaches become largely ineffective, since their basic assumptions about the malware control architecture or protocol no longer hold true. Recent efforts from the research community on understanding Plotters, including Storm [1], [13], [14], [15], Nugache [13], Waledac [16], and Conficker [17], [18], provided valuable insight to the operations of these malware, but effective techniques to detect them and future variants are still a subject of ongoing research.

Early work on disrupting Plotters (targeting Storm, in particular) injected a large number of fake nodes into the network to perform various Sybil attacks [15], [19], [20], such as content-poisoning or eclipsing certain nodes from the rest of the P2P network. These studies showed that the effectiveness of the attack depends on the attack duration as well as the number of Sybils. Kang et al. [21] developed a P2P monitor that infiltrated the Storm botnet to identify the IP addresses of infected hosts. They showed that the monitor was able to detect bots behind firewalls or NAT devices, achieving a broader coverage than others that actively crawl the network. Wurzinger et al. [22] constructed network intrusion detection signatures to identify botmaster commands by examining bot binaries running in controlled environments. Their main observation is that changes in the network behaviors of a Plotter are indications of it having received commands from the botmaster. They examine network traffic immediately preceding behavior changes, and identify common substrings in the payload that can be used as signatures. This approach is hindered by Plotters that encrypt their communication, and may miss previously unknown attacks.

Many approaches have also been proposed that detect malware by examining the behavioral characteristics of the network traffic. BotHunter [23] detects compromised hosts by identifying a series of events that takes place when a vulnerable host is infected, and which shows evidence of coordinated activities between the infected host and the botmaster. However, since they specifically focused on detecting events related to certain suspicious behaviors, including scanning, binary download, and control channel establishment, Plotters not conforming to this profile would go undetected. Other works correlated traffic characteristics to identify hosts exhibiting similar network behaviors, such as per-

forming suspicious activities (e.g., scanning, spamming) and sharing common communication contents [24], exhibiting similar traffic statistics and suspicious activities [25], or contacting the same new destinations, exchanging similar payload, and involving hosts of similar software platform [26]. These approaches can be evaded by changes in malware behavior, many of which have already taken place, such as turning to social engineering as an infection vector instead of scanning, or using encryption to make payload analysis difficult. Still others (e.g. [27]) use behavioral analysis to identify certain P2P-bot behaviors exhibited over non-P2P protocols.

In contrast to previous work, we focus specifically on distinguishing Plotters, whose command-and-control channel is overlaid on top of a P2P protocol, from Traders. We do so by observing network-level characteristics inherent to P2P applications, but that are able to distinguish Plotters from Traders due to the different goals and circumstances behind how they utilize the P2P protocol. For example, Plotters communicate over P2P networks mainly for subtlety and resilience, instead of large file exchanges. They are also incentivized to maintain persistent connections to other peers in the network, in contrast to Traders, who have been observed to go offline after the completion of file transfers [5].

Jelasy et al. [28] studied techniques that can be deployed by Plotters to evade P2P traffic detection. However, they only consider the case where traffic dispersion graphs (TDGs) [29] are used to identify P2P traffic. The TDG-approach assumes a global view of the network, constructing a communication graph between all nodes to check if the average degree and the fraction of nodes with both incoming and outgoing connections are above a threshold. To evade such detection, the authors specifically focused on reducing the number of peers each Plotter contacts, such that most of the botnet’s traffic are routed through a few fixed nodes. While this approach may limit the number of detectable Plotters using TDGs, its impact on other methods for identifying P2P traffic (that does not require the communication graph) is not evaluated.

One of the characteristics explored in this work is the difference between human-driven and machine-driven traffic. This observation has also been applied in other contexts, including cheat detection in online games [30], distributed denial-of-service attack defenses [31], [32], and chat bot detection in Internet chat rooms [6]. While most approaches to identifying automated traffic were host-based, Gianvecchio et al. [6] found that the network traffic from human activities shows a higher entropy

than those from bots, for the case of Internet chat room traffic. Lu et al. [33], [34] assume Plotter activities to be more synchronized than human activities, and detect Plotters by looking for hosts with similar byte frequency distributions in their payload within the same time window, e.g., one second. This approach can thus be evaded with encryption. Similar to our observation of the persistence in Plotters’ traffic, Giroire et al. [35] proposed a method to detecting centralized botnet command-and-control traffic by monitoring persistent and regular connections made to the same group of destination IP addresses, i.e., the command-and-control server. Since legitimate user traffic can also appear to be persistent and regular, this approach requires whitelisting common sites the user frequently visits, and is not suitable for detecting Plotters that communicate over P2P.

Another line of work, orthogonal to ours, include techniques for identifying behaviors involving certain operations of Plotters. Ramachandran et al. [36] observed that botmasters lookup DNS blacklists to determine whether their Plotters are blacklisted. The authors thus monitor lookups to a DNS-based blacklist to identify infected hosts. Fast-flux is a technique used by botnets to hide the backend control server [37]. It operates by using dynamic DNS to establish a proxy network based on the infected hosts, such that a single domain is associated with many different IP addresses. Methods for identifying fast-flux include observing the geographic diversity in the IPs associated with a domain and the heterogeneity of those hosts [38], [39], [40]. Since fast-flux networks are often used to host spam campaigns or phishing websites, Hu et al. [41] also proposed to detect hosts participating in fast-flux networks by identifying HTTP redirection activity.

III. DATA COLLECTION

In this work, we assume the role of a network administrator that aims to identify Plotters internal to her network, by observing only traffic crossing the border of the network. The network traffic utilized in our analysis is organized into bi-directional flow records by Argus (<http://www.qosient.com/argus>), which is a real-time flow monitor based on the RTFM flow model [42], [43]. Argus inspects each packet and groups together those within the same connection into one bi-directional record. In particular, TCP and UDP flows are identified by the 5-tuple (source IP address, destination IP address, source port, destination port, protocol)³, and packets

³Since Argus records are bi-directional, the source and destination IP addresses are swappable in the logic that matches packets to flows. However, the source IP address in the record is set to the IP address of the host that initiated the connection.

in both directions are recorded as a summary of the communication, namely, an Argus flow record. The fields contained in each Argus record include the source and destination IP addresses and ports, the protocol, the start and end times of the flow, the packet and byte counts, and the first 64 bytes of the payload on the connection. This payload is used solely for determining ground truth for our tests, that is, determining whether the host is a Plotter or a Trader.

We use the following datasets in our analysis:

CMU dataset: This dataset consists of anonymized network traffic obtained from the edge routers of the Carnegie Mellon University (CMU) campus network, which has two /16 subnets. The rate of this traffic is about 5000 flows per second, and is collected daily from 9 a.m. to 3 p.m. over eight days in November 2007. We focus on only TCP and UDP traffic in this dataset.

Trader dataset: We identified those hosts in the CMU dataset that are participating in known P2P file-sharing networks, i.e., the Traders, using the 64 bytes of payload in each flow record available to us. Specifically, we focus on the three popular file-sharing applications, Gnutella, eMule, and BitTorrent. Hosts running Gnutella were identified by the protocol keywords “GNUTELLA”, “CONNECT BACK”, and “LIME” in their payload.⁴ eMule hosts were identified by the initial byte ‘0xe3’ or ‘0xc5’, followed by various byte sequences as specified in the protocol specification [44]. BitTorrent hosts were identified by the protocol keyword “BitTorrent protocol”, web requests to trackers beginning with “GET /scrape” or “GET /announce”, and distributed hash table control messages with the substrings “d1:ad2:id20” or “d1:rd2:id20”.⁵

Plotter dataset: We also obtained Plotter traffic traces gathered from honeynets running in the wild in late 2007. These include a 24-hour trace of Storm, which contains traffic from 13 bots, and a 24-hour trace of Nugache, which contains 82 bots. Spamming and scanning activities were blocked during the collection of these traces, and so the remaining traffic consists mostly of botnet control traffic, e.g., for peer discovery. As we will describe in §V, these traces were used in our evaluation, where they were overlaid onto the CMU traffic by assigning them to randomly selected internal hosts that are active in the CMU dataset.

IV. METHODOLOGY

Given network traffic observed at the border of an enterprise network, our goal is to identify internal

⁴http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html

⁵<http://wiki.theory.org/BitTorrentSpecification>

Plotters, where the main challenge in doing so is to distinguish them from Traders. We construct a set of tests that quantify the characteristics described in §I: volume, peer churn and persistence, and human-driven versus machine-driven, which aim to take advantage of the different goals and circumstances behind how Plotters and Traders utilize P2P networks. Each test takes as input a collection of traffic, Λ , which involves a group S of internal hosts over time window D , and outputs a subset of hosts in S that exhibit characteristics for which the test evaluates. In the following, we detail the rationale behind each of the characteristics, how they can be useful indicators for distinguishing Plotters from Traders in particular, and the construction of the corresponding test functions. We then describe how multiple tests can be combined to refine the results to narrow in on Plotters within the local network.

A. Volume

The first distinguishing characteristic we consider between Plotters and Traders is the amount of traffic each host contributes to the network. A common purpose of Traders is to exchange data, and much of the data found on popular P2P file-sharing applications, such as Gnutella, eMule, and BitTorrent, are large multimedia files (e.g., several MBytes in size [4]). By contrast, the use of P2P architectures in Plotters is not so much for the sharing of information as for resilience and subtlety. Their traffic hence tends to be much lower in volume. In fact, the Storm botnet has been observed to use the P2P protocol only for exchanging control messages, while file transfers were still performed over HTTP [1], [13].

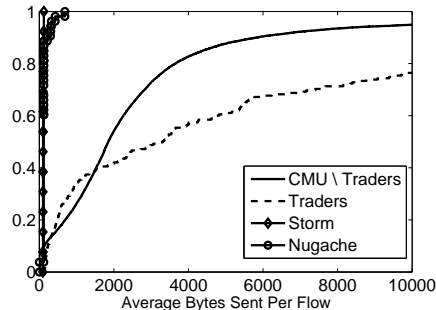


Figure 1. Cumulative distribution of the average flow size per host in each dataset over one day.

We examine traffic volume for a host in terms of the average number of bytes per flow that it contributes to the network (i.e., uploaded by the host). Compared to the cumulative byte count, this metric is less likely to be biased by the number of flows generated by a host, since

a Plotter that is chatty can accumulate a large byte count over a short time window, while each individual flow is quite light-weight. Figure 1 shows the cumulative distribution of this value per host, plotted from a single day of traffic from the CMU dataset, the Trader dataset, and the Plotter traces. This figure shows that the amount of data contributed by the Plotters (i.e., the Storm and Nugache bots), is significantly smaller than Traders.

Tests on Volume: By quantifying a host’s traffic volume using the average number of bytes sent per flow, we can define a test function θ_{vol} that uses this characteristic to distinguish between Traders and Plotters. The function takes as input a collection of traffic, Λ , which involves a group S of internal hosts over time window D , and a threshold τ_{vol} . Hosts whose average flow size is less than τ_{vol} are returned in the set S_{vol} .

In practice, τ_{vol} can be set dynamically depending on the current traffic makeup, for example, by setting it to the median value observed across all hosts in S . This can make it more difficult for a Plotter to manipulate its traffic to masquerade itself as a Trader; e.g., the amount of data it sends per flow needs to be larger than the majority of the Traders in the local network, though the Plotter would presumably be unaware of that amount that it must exceed.

B. Peer Churn and Persistence

Peer churn refers to the dynamics of peers joining and leaving the network, and is a common phenomenon both among Traders and Plotters. This characteristic is often reflected in the high ratio of failed connections observed in P2P networks [45], [46]. Previous studies on P2P file-sharing networks have shown that peers are often connected for only short durations (a few minutes on average) [3], [4], [5], and many of them leave the network permanently after requesting a single file [5].

We hypothesize that even though the dynamics of peer membership is present in both systems, peer churn is less significant among Plotters compared to Traders. This is because Plotters have motivation to keep up persistent communications with each other and maintain the connectivity of the botnet, since the botmaster needs to be able to control her bots. The Plotter also cannot control when network access would be available on the infected machine, and so it is often opportunistic in initiating communications, i.e., whenever it has a chance, making a Plotter’s network activities more persistent in doing so. In addition, most Plotters store a list of known peers with which it maintains communications, both for bootstrapping itself into the network [1], [13], [14], [15], [16] and to limit the number of active connections. This allows the Plotter to remain stealthy, and also

limits the disclosure of other Plotters if one of them is captured [13]. Such behaviors make it more likely for Plotters to contact the same hosts than Traders, whose set of peers is mainly determined by file availability.

This observation allows us to characterize peer churn using the set membership of the destinations that a host contacts. We quantify this by the fraction of new IP addresses that a host contacts per day, or more specifically, the ratio of (i) the number of IP addresses that a host first contacts after its first hour of activity on that day, and (ii) the total number of IP addresses it contacts in that day. A higher percentage of new contacts indicates a higher amount of churn. As an example, we illustrate the percentage of new addresses contacted by a Storm bot (from our Plotter traces) and a Trader (in one-day’s worth of traffic from the Trader dataset) in Figure 2. Clearly, churn is significant for the Trader, where over 55% of the IPs it contacted appear to be new. In contrast, generally more than 60% of the peers contacted by the Storm Plotter have been contacted previously by the same host.

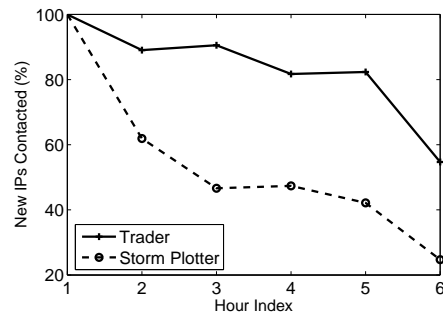


Figure 2. New IPs contacted by a Trader and a Storm Plotter over one day.

Tests on Peer Churn and Persistence: Similar to the tests for volume, we also approach the problem of distinguishing Plotters from Traders using churn by performing a coarse separation between the two sets of hosts. In this case, the test function for peer churn, θ_{churn} , identifies hosts that have a relatively “low” churn (which are likely Plotters) using a threshold τ_{churn} . By taking as input a collection of traffic Λ involving hosts S and a threshold τ_{churn} , the churn test $\theta_{churn}(\Lambda, S, \tau_{churn})$ outputs a set S_{churn} of hosts that contact a percentage of new IP addresses less than τ_{churn} .

In practice, a Plotter could attempt to evade being grouped into S_{churn} by increasing the fraction of new hosts it contacts, for example, by performing random scanning or initiating connections to different peers on its peer list at every communication attempt. This ap-

proach is risky, since it could make the Plotter detectable via other means (e.g., by identifying scanning activities) and reduces the stealthiness of the Plotter. We discuss evasion techniques that can be carried out by Plotters and quantify their induced costs in §VI.

C. Human-driven vs. Machine-driven

Several works on botnet detection have studied the difference between human and machine-driven activities [47], [24], [26], [25], [33], [34]. Only a few of these previous works have applied their technique to detecting P2P Plotters [24], [25]. However, these approaches rely on the presence of specific attack activities performed by the infected hosts, such as scanning.

We approach this problem by directly using timing-related information to characterize the *regularity* of machine-driven activities, for example, periodic keep-alive/status messages exchanged between peers or scheduled checks performed by the Plotters to download new commands, as well as the commonality between the behaviors of Plotters belonging to the same botnet. Specifically, for each host, we examine the interstitial time distribution of its flows to the same destination IP. This distribution is observed across all destinations contacted by the host, since we do not know which ones are P2P peers. Since Plotters in the same botnet are likely to be running similar versions of the bot binary, the timers used in triggering their activities should also follow the same algorithm. From this observation, the per-destination flow interstitial time distributions for Plotters should not only stand out from those of Traders, whose activities lack the regularity seen in automated traffic, but also appear “similar” to each other.

Figures 3(a) and 3(b) show the patterns that appeared in the per-destination flow interstitial time distribution of two Plotters, a Storm bot and a Nugache bot, from our Plotter traces. These Plotters exhibit significant periodicity in their communications. For example, Nugache can be observed to communicate at intervals of around 10 seconds, 25 seconds, and 50 seconds. By contrast, it is not clear that the same pattern exists among Traders, as shown in Figures 3(c) and 3(d) for a BitTorrent host and a Gnutella host from the Trader dataset.

Tests on Human-driven v.s. Machine-driven: To compare the per-destination flow interstitial time distribution between hosts, we define a function, θ_{hm} , that uses a non-parametric approach to construct a histogram that approximates the underlying distribution for each host [48]. The Earth Mover’s Distance [49] is then applied as the distance metric for comparing distributions. This allows us to identify clusters of hosts who exhibit similar timing patterns in their network

traffic, where hosts whose traffic are mainly machine-driven, e.g., Plotters, should have different interstitial time distributions from hosts that are human-driven, e.g., Traders, and thus fall within separate clusters.

- **Constructing Histograms.** Given a collection of the observed interstitial time samples $v(s)$ for a host s , we approximate its underlying distribution by constructing a histogram. The choice of histogram bin width is critical in this approximation, since a large value leads to over-smoothing, and a small value increases the sampling error. Moreover, applying a fixed bin width makes it straightforward for a Plotter to manipulate its traffic to evade detection.

In this work, we follow a method proposed by Freedman et al. [48] to identify the optimal bin width, whose goal is to minimize the mean-squared error between the true distribution and the histogram. They show that the bin width can be computed as a function of the sample size $|v(s)|$ (i.e., the number of observed interstitial time values for host s) and the “spread” of the samples, as represented by the inter-quartile range of the sample values, $IQR(v(s))$. Specifically, the bin width b_s is calculated by $b_s = 2 \times IQR(v(s)) \times |v(s)|^{-1/3}$.

- **Clustering Histograms.** One of the metrics for comparing distributions is the Earth Mover’s Distance (EMD) [49]. Briefly, EMD is defined as the amount of work that is required to change one distribution into the other by moving “distribution mass” around. It is based on the transportation problem [50], where the challenge is to find routes that will minimize the cost of shipping goods from a group of suppliers I to a group of consumers J . That is, find a set of routes f_{ij} to minimize $\sum_{i \in I} \sum_{j \in J} c_{ij} f_{ij}$, where c_{ij} is the cost of shipping from supplier i to consumer j . By defining c_{ij} as the distance between the i^{th} and j^{th} bins in the histograms, EMD is especially useful in cases where the distributions are simply shifts of each other, but otherwise identical.

This allows us to form clusters of hosts whose histograms are “close”, which correspond to hosts whose network traffic exhibit similar timing patterns. Clustering is performed using an agglomerative hierarchical algorithm, where each step merges the two hosts with the closest distributions. This iterative process constructs a hierarchical clustering tree (i.e., a dendrogram), with the weight of each link being the average distance between the pair of nodes it connects. The final set of clusters is formed by cutting the top 5% links with the largest weights.

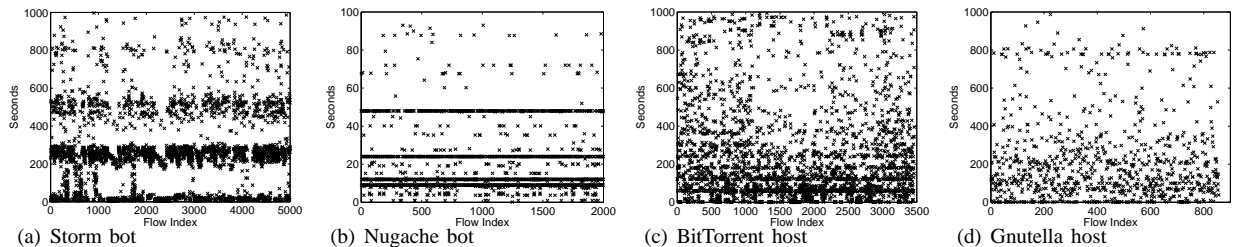


Figure 3. Interstitial time between flows to same destination in one day's worth of traffic.

```

FindPlotters( $\Lambda, S$ )
100:  $S_{vol} \leftarrow \theta_{vol}(\Lambda, S, \tau_{vol})$ 
      /* Returns hosts with low traffic volume */
101:  $S_{churn} \leftarrow \theta_{churn}(\Lambda, S, \tau_{churn})$ 
      /* Returns hosts with low peer churn */
102:  $S_{hm} \leftarrow \theta_{hm}(\Lambda, S_{vol} \cup S_{churn}, \tau_{hm})$ 
      /* Returns hosts with similar timing patterns in their traffic */
103: return  $S_{hm}$ 

```

Figure 4. The algorithm used to find suspected Plotters by combining the tests on volume (line 100), peer churn and persistence (line 101), and human-driven versus machine-driven traffic (line 102), described in §IV-A, IV-B, and IV-C.

θ_{hm} also takes as input a threshold parameter, τ_{hm} , which filters out clusters whose diameters exceed τ_{hm} . Similar to the two previous tests, τ_{hm} can be set dynamically as a function of the diameters across all clusters. The output from the human-driven versus machine-driven test, $\theta_{hm}(\Lambda, S, \tau_{hm})$, is the union of the host clusters not filtered out in this way.

D. Data Fusion

Each of the above tests, θ_{vol} , θ_{churn} , and θ_{hm} , aims to find Plotters using behavioral characteristics of a host's network traffic. Alone, each test provides only a coarse way of separating Plotters from Traders, and may not be an effective tool for identifying Plotters. In §V, though, we show that when used in combination, they can narrow in on the Plotters, while largely eliminating the non-Plotter hosts.

Specifically, we combine the tests into an algorithm, FindPlotters, shown in Figure 4. The algorithm takes as input a collection of traffic Λ involving a set of hosts S observed within a time window D (e.g., one day), and outputs hosts who pass our various tests, indicating that they are likely to be Plotters.

V. EVALUATION

We present an evaluation of the tests described in §IV, using traffic from Plotters overlaid onto flow records recorded at the edge of the CMU campus network (the CMU dataset). For each day of traffic in the CMU

dataset, we overlay the bot traces by assigning them to randomly selected internal hosts that are active during that day (including possibly Traders). This makes our testing scenario more realistic, since the internal hosts still exhibit their normal connection patterns, in addition to Plotter activities.

A. Initial Data Reduction

To serve as an initial data reduction step in our analysis, we first deploy a simple method to filter out hosts that are unlikely to be running P2P applications at all, by considering only hosts that have relatively high failed connection rates. Failed connection rate has been utilized in previous works on identifying P2P traffic (e.g., [45], [46]), and here we use it simply as a coarse data-reduction step for eliminating hosts that are likely not running P2P applications at all, i.e., that are neither a Trader nor a Plotter.

Figure 5 shows the cumulative distribution of the percentage of failed connections per host, plotted from a single day of traffic from the CMU dataset, the Trader dataset, and the Plotter traces. Only hosts that initiated successful connections within that day were included. There is a clear distinction between the curves for the CMU\Trader and Trader datasets, pointing out that P2P hosts do exhibit significantly higher failed connection rates compared to non-P2P hosts. A closer examination of the Traders with a small percentage of failed connections (e.g., less than 10%) revealed that they are BitTorrent hosts downloading Torrent files from trackers over HTTP, but that are not otherwise involved in P2P file-sharing activities.

Surprisingly, the Plotter traces also exhibit very different failed connection rates. In particular, many of the peer discovery messages sent by Nugache Plotters in our trace were unsuccessful, because the remote peer was either not active or not responding. This causes almost all Nugache Plotters to have more than 65% failed connections. Note that the curves for Storm and Nugache in Figure 5 are generated from the Plotter

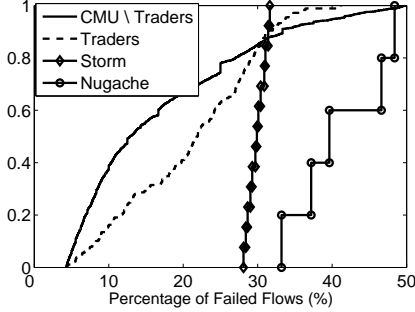


Figure 5. Cumulative distribution of the percentage of failed connections per host in each dataset over one day.

traces *only*. When they are overlaid onto the CMU dataset (§V-B), the percentage of failed flows can be biased by the traffic from the CMU host to which we assigned the Plotter traces.

As a data-reduction step to filter out those hosts who are likely *not* involved in P2P activities—while retaining all hosts that are in fact running P2P applications—we use the median value among hosts in the CMU dataset with Plotters overlaid (and that initiated successful flows) as the threshold for deciding which hosts to remove from consideration. This value is determined anew for each day of traffic. For example, for the case of Figure 5, the threshold for failed connection rate would be roughly 25% (i.e., 25.74%, the median value for the CMU dataset, then adjusted due to the overlaid Plotter data). Hosts with failed connection rates higher than the threshold are selected as “possibly P2P”. This approach not only allows us to eliminate half of the hosts that are not likely to be Plotters, but is also more difficult for a Plotter to evade compared to setting a fixed threshold.

B. Identifying Plotters

We overlaid the Storm and Nugache Plotter traces onto each day of traffic in the CMU dataset by assigning them to originate from randomly selected internal hosts in the CMU campus network observed to be active on that day (i.e., the detection window D is one day). This combined traffic is then given as input to the tests, where each returns a set of hosts that survived the test.

Figures 6, 7, 8 show ROC (Receiver Operating Characteristic) curves for the volume, churn, and human-driven vs. machine-driven tests. The input to the volume and churn tests is the set S of hosts that passed the initial data reduction step described in §V-A. The ROC curves are generated by setting the threshold τ_{vol} to be the 10, 30, 50, 70, or 90th percentile of the average bytes sent per flow per host, and τ_{churn} to be the 10,

30, 50, 70, or 90th percentile of the fraction of new IP addresses contacted per host. The input to the human-driven vs. machine-driven test, θ_{hm} , are those hosts that were retained by one of the volume or churn tests (i.e., $S_{vol} \cup S_{churn}$) with their respective thresholds set at the 50th percentiles (and by the initial data reduction step). To generate the ROC curve in Figure 8, the threshold τ_{hm} for θ_{hm} is set to be the 10, 30, 50, 70, or 90th percentile of the cluster diameters. We emphasize that each ROC curve plots the true and false positive rates *relative to its input set* (i.e., S for θ_{vol} and θ_{churn} , and $S_{vol} \cup S_{churn}$ for θ_{hm}), as opposed to the overall CMU dataset with Plotters overlaid, in order to highlight the independent discriminating power that each test contributes.

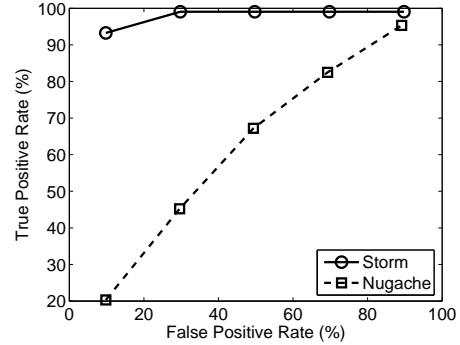


Figure 6. ROC curves for the volume test θ_{vol} when the Storm and Nugache traces are overlaid onto hosts in the CMU dataset, after filtering as in §V-A. Results are averaged over the eight days in the CMU dataset.

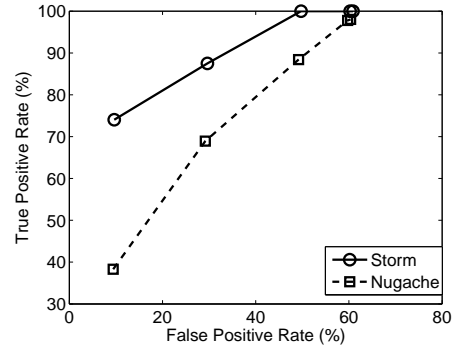


Figure 7. ROC curves for the peer churn test θ_{churn} when the Storm and Nugache traces are overlaid onto hosts in the CMU dataset, after filtering as in §V-A. Results are averaged over the eight days in the CMU dataset.

Two observations from Figures 6–8 are evident. First, the true positive rates for Storm are higher than Nugache across all three tests, often reaching 100%. We will

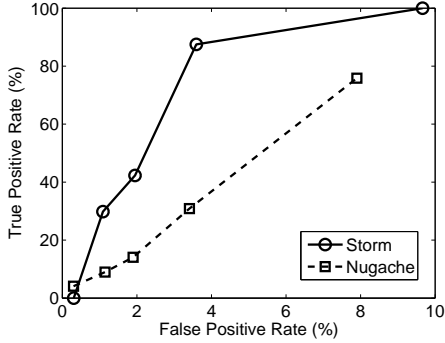


Figure 8. ROC curves for the human-driven vs. machine-driven test when the Storm and Nugache traces are overlaid onto hosts in the CMU dataset, after filtering as in §V-A and by θ_{vol} and θ_{churn} . Results are averaged over the eight days in the CMU dataset.

explore the reasons for this difference at the end of this section. The second observation is that alone, each of the tests would be too coarse to be effective at identifying Plotters, producing high false positive rates that can reach to 90% (e.g., the volume test).

In combination, however, they can be powerful at extracting Plotters from Trader-like hosts. To show this, we utilized the tests together as in the algorithm FindPlotters (Figure 4). To strike a balance between the true positive and false positive rates, we use the 50th percentile among the hosts as the threshold for both τ_{vol} and τ_{churn} , and the 70th percentile of the cluster diameters for τ_{hm} . Figure 9 shows how the results are refined at each step, where the false positive rate is reduced to 0.81%, while maintaining a 87.50% true positive rate for Storm and 30% for Nugache. The percentage of Traders (from the Trader dataset) that remain after each test (averaged over all trials) is also shown for comparison. On average, 5.40% of the Traders remained after applying the tests, which comprises 7.11% of all the hosts returned by FindPlotters.

We now return to the differences in detection rates between Nugache and Storm. As shown in Figure 9, most false negatives for Nugache resulted from θ_{hm} . Further investigation into these results showed that each test, but particularly θ_{hm} , tended to filter out less communicative Plotters, as shown in Figure 10. Since these Plotters generated fewer connections, their behaviors were more obscured by traffic from the CMU hosts onto which they were overlaid. At present, we have been unable to confirm a reason behind the large variance in the activity levels of the Nugache bots in our trace, though those who originally recorded the trace suggested that this may be due to the limited viability of

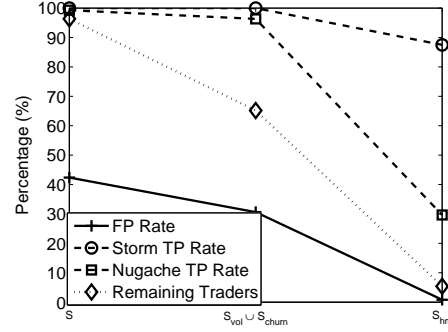


Figure 9. Results after applying the tests in sequence. The numbers are averaged over eight days in the CMU dataset with overlaid Plotter traffic.

the Nugache botnet at the time this trace was recorded.⁶ This variability, and the generally lower rate of Nugache flows in comparison to Storm, rendered all of our tests less effective, as shown in Figures 6–8.

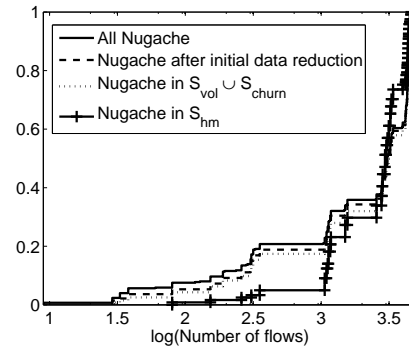


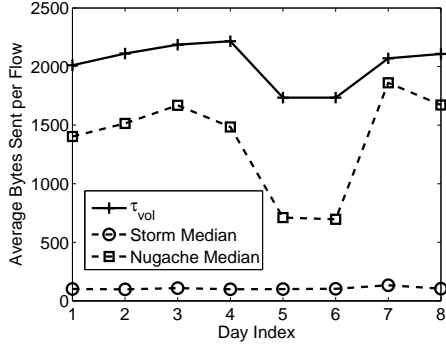
Figure 10. Cumulative distribution of the number of flows generated by the Nugache Plotters that remain after each test, in base-10 log scale. Results are accumulated over the eight days in the CMU dataset.

VI. EVASION

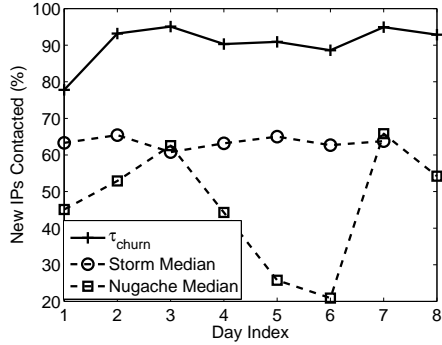
A Plotter could attempt to change its network behaviors to evade our tests, e.g., by increasing its traffic volume so that it will escape the volume test. However, since the thresholds used in our tests are not fixed at set values, but instead are dependent on traffic statistics from *all* hosts observed to be active in the local network (specifically, the hosts that remain after the initial data reduction step described in §V-A), a Plotter would have difficulty in determining the precise thresholds that will allow it to masquerade as a Trader.

Figures 11(a) and 11(b) show, for the volume test θ_{vol} and churn test θ_{churn} conducted on each day of traffic

⁶Guofei Gu, personal communication, October 2009.



(a) The threshold τ_{vol} in the test θ_{vol} compared to values observed from hosts with overlaid Plotter traffic.



(b) The threshold τ_{churn} in the test θ_{churn} compared to values observed from hosts with overlaid Plotter traffic.

Figure 11. Challenges for Plotters to evade θ_{vol} or θ_{churn} .

in the CMU dataset, the detection threshold used (i.e., the median among the hosts) versus the median value among the Plotters, once assigned to hosts. To evade the volume test, θ_{vol} , the median Storm Plotter would need to generate more than *five* times its original traffic volume per flow. The corresponding multiplicative factor for the median Nugache Plotter is roughly 1.3. To evade the churn test, θ_{churn} , a Plotter can either refrain from contacting hosts it had previously communicated with, or generate connections to a large number of new hosts it talks to only once. As an example of the latter case, a Plotter who wants to raise its percentage of new IPs to 90% (a typical value of τ_{churn}), while still maintaining communications with the same number of peers, would need to increase the fraction of new hosts it contacts by a factor of 1.5 or more. Such evasion attempts from Plotters that increase their traffic volume or the number of new hosts (such as through random scanning) can compromise their stealthiness, making their presence in the network observable through other means (e.g., scan detection) or even by the owner of the infected machine.

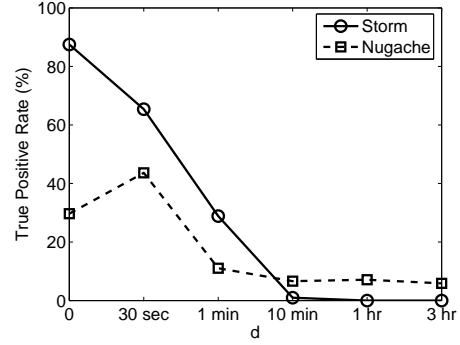


Figure 12. Challenges for Plotters to evade θ_{hm} . The y-axis is the true positive rate averaged over eight days of the CMU dataset with overlaid Plotter traffic.

The human-driven vs. machine-driven test, θ_{hm} , clusters hosts based on the distribution of their per-destination interstitial flow times, and identifies hosts that have the similar timing patterns in their communications. Plotters belonging to the same botnet can avoid falling into the same cluster or increase the cluster diameter, for example, by having each Plotter select a different frequency at which to contact peers. This could affect our choice of bin width in histogram construction — which is dependent on both the number of interstitial time samples observed and the inter-quartile range of the samples (see §IV-C) — and therefore alter the Earth Mover’s Distance (EMD) between Plotters.

To quantify the operational cost for Plotters that want to evade detection by θ_{hm} , we simulated Plotters who, instead of initiating communications at regular intervals (see §IV-C), always add (or subtract) a random delay before making each connection. By manipulating the distribution from which the flow interstitial times are drawn, the Plotters may be able to disrupt our histogram construction and clustering algorithm, such that they no longer fall within the same cluster, or that the cluster diameter exceeds the threshold τ_{hm} .

We use the same Plotter traces that were used in the evaluation for this simulation, but add (or subtract) a random delay before every connection a Plotter makes to a peer with which it had previously communicated. The delay is drawn from a uniform distribution over the interval $\pm d$, for each connection. Figure 12 shows the decay in the true positive rate as a function of d , ranging from 30 seconds to three hours. The slight increase at 30 seconds for Nugache is due to the Plotters being separated into multiple clusters of small diameter, resulting in their retention by θ_{hm} . While they were able to cluster better (i.e., being grouped together into

fewer clusters) before adding the random delay, these larger clusters often had larger diameters and so were filtered out by θ_{hm} . This figure suggests that Plotters must randomize their connections to other Plotters by minutes in order to evade detection via this test, potentially slowing the responsiveness of the botnet (c.f., Figure 3(b)). Moreover, the per-destination interstitial flow time distribution of other machines in the local network affects the needed value of d , which may be difficult for Plotters to measure.

Since our tests focus on characteristics that describe differences in Plotter and Trader behavior, a limitation of this approach is in identifying Plotters that only affect Traders, e.g., a Plotter binary that spreads through P2P file-sharing networks. In this case, the Plotter traffic could be obscured by activities from the Trader, if the Trader is a heavy file-sharing user generating high volumes of traffic, for example. One method of distinguishing between Plotter and Trader traffic on a host might be to separate traffic by application, such as determined using port numbers. Traffic from each port, or a group of associated ports, can then be applied individually to the tests in §IV. While in our evaluations the hosts to which we assigned bot traces were sometimes Traders, and were still effectively identified by the FindPlotters algorithm, a more comprehensive study as to how Plotters can selectively infect only Traders that will obscure its traffic is part of ongoing work.

VII. CONCLUSION

In networks where P2P file-sharing is commonplace, a challenge in identifying bots managed via P2P infrastructures is the similarities that their network behaviors share with P2P file-sharing applications. In this paper we developed a series of tests for separating the two classes of P2P applications, and in particular for identifying bots within a network prior to their engaging in overt attacks. Our tests work on flow records, without access to packet payloads or packet interstitial times. As such, our techniques are scalable to busy networks where packet capture (or even packet header capture) is not cost-effective. For the same reason, our techniques are immune to bot payload encryption.

Using bot traces and traces of traffic collected at the edge of a university network, we showed that our technique enabled the identification of Storm and Nugache bots with a false positive rate of only 0.81% on average. At this false positive rate, our techniques identified 87.50% of the implanted Storm bots, and 30% of the Nugache bots. Our lower detection rate for Nugache derives from the low and variable activity of the bots in our data (see §V-B), and so we believe this number to

be conservative. We further evaluated the changes in bot behavior needed to evade our detection technique in our tests, and found that bots would need to increase their average flow size by roughly a factor of 1.3; increase the fraction of new IP addresses they contact by a factor of 1.5 or more; or randomize their interstitial connection times significantly (e.g., in a range of minutes) in order to avoid detection. Moreover, the bots would need to accomplish such changes despite other traffic from the host it occupies, and since we defined our tests' thresholds relative to the background traffic (e.g., the median), the behavior necessary to evade detection in any given network would typically be unknown to the attacker.

REFERENCES

- [1] J. Grizzard, V. Sharma, C. Nunnery, and B. Kang, "Peer-to-peer botnets: Overview and case study," in *Wksh. Hot Topics in Understanding Botnets*, 2007.
- [2] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *Intern. Wksh. Peer-to-Peer Systems*, 2002.
- [3] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *ACM SIGCOMM Internet Measurement Conf.*, 2006.
- [4] S. Saroiu, P. Gummadi, and S. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Multimedia Computing and Networking*, 2002.
- [5] K. Gummadi, R. Dunn, S. Saroiu, and S. Gribble, "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," in *ACM Symp. Operating Systems Principles*, 2003.
- [6] S. Gianvecchio, M. Xie, Z. Wu, and H. Wang, "Measurement and classification of humans and bots in internet chat," in *USENIX Security Symp.*, 2008.
- [7] J. Binkley and S. Singh, "An algorithm for anomaly-based botnet detection," in *Wksh. Steps to Reducing Unwanted Traffic on the Internet*, 2006.
- [8] E. Cooke, F. Jahanian, and D. McPherson, "The zombie roundup: Understanding, detecting, and disrupting botnets," in *Wksh. Steps to Reducing Unwanted Traffic on the Internet*, 2005.
- [9] J. Goebel and T. Holz, "Rishi: Identify bot contaminated hosts by IRC nickname evaluation," in *Wksh. Hot Topics in Understanding Botnets*, 2007.
- [10] C. Livadas, B. Walsh, D. Lapsley, and T. Strayer, "Using machine learning techniques to identify botnet traffic," in *IEEE LCN Wksh. Network Security*, 2006.
- [11] P. Bäcker, T. Holz, M. Kötter, and G. Wicherski, "Know your enemy: Tracking botnets," The HoneyNet Project and Research Alliance, Tech. Rep., 2005.
- [12] A. Karasaris, B. Rexroad, and D. Hoeflin, "Wide-scale botnet detection and characterization," in *1st Wksh. Hot Topics in Understanding Botnets*, 2007.
- [13] S. Stover, D. Dittrich, J. Hernandez, and S. Dietrich, "Analysis of the Storm and Nugache trojans: P2P is here," *USENIX ;login*, vol. 32, no. 6, 2007.
- [14] P. Porras, H. Saidi, and V. Yegneswaran, "A multi-perspective analysis of the Storm (Peacomm) worm,"

- Computer Science Laboratory, SRI International, Tech. Rep., 2007.
- [15] 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 *USENIX Wksh. Large-Scale Exploits and Emergent Threats*, 2008.
- [16] L. Borup, "Peer-to-peer botnets: A case study on Waledac," Master's thesis, Technical University of Denmark, 2009.
- [17] P. Porras, H. Saidi, and V. Yegneswaran, "An analysis of Conficker's logic and rendezvous points," Computer Science Laboratory, SRI International, Tech. Rep., 2009.
- [18] G. Lawton, "On the trail of the Conficker worm," *Computer*, vol. 42, no. 6, 2009.
- [19] C. Davis, J. Fernandez, S. Neville, and J. McHugh, "Sybil attacks as a mitigation strategy against the Storm botnet," in *Intern. Conf. Malicious and Unwanted Software*, 2008.
- [20] D. Ha, G. Yan, S. Eidenbenz, and H. Ngo, "On the effectiveness of structural detection and defense against P2P-based botnets," in *IEEE/IFIP Intern. Conf. Dependable Systems and Networks*, 2009.
- [21] B. Kang, E. Chan-Tin, C. Lee, J. Tyra, H. Kang, C. Nunnery, Z. Wadler, G. Sinclair, N. Hopper, D. Dagon, and Y. Kim, "Towards complete node enumeration in a peer-to-peer botnet," in *ACM Symp. Information, Computer and Communications Security*, 2009.
- [22] P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, and E. Kirda, "Automatically generating models for botnet detection," in *European Symp. Research in Computer Security*, 2009.
- [23] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "BotHunter: Detecting malware infection through IDS-driven dialog correlation," in *USENIX Security Symp.*, 2007.
- [24] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting botnet command and control channels in network traffic," in *ISOC Network and Distributed System Security Symp.*, 2008.
- [25] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *USENIX Security Symp.*, 2008.
- [26] T. Yen and M. K. Reiter, "Traffic aggregation for malware detection," in *Conf. Detection of Intrusions and Malware and Vulnerability Assessment*, 2008.
- [27] S. Chang and T. E. Daniels, "P2P botnet detection using behavior clustering & statistical tests," in *2nd Wksh. Security and Artificial Intelligence*, Nov. 2009.
- [28] M. Jelasity and V. Bilicki, "Towards automated detection of peer-to-peer botnets: On the limits of local approaches," in *USENIX Wksh. Large-Scale Exploits and Emergent Threats*, 2009.
- [29] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese, "Network monitoring using traffic dispersion graphs (TDGs)," in *ACM SIGCOMM Internet Measurement Conf.*, 2007.
- [30] T. Schluessler, S. Goglin, and E. Johnson, "Is a bot at the controls? detecting input data attacks," in *Wksh. Network and Systems Support for Games*, 2007.
- [31] S. Kandula, D. Katabi, M. Jacob, and A. Berger, "Botz-4-sale: Surviving organized DDoS attacks that mimic flash crowds," in *USENIX Symp. Networked Systems Design and Implementation*, 2005.
- [32] R. Gummadi, H. Balakrishnan, P. Maniatis, and S. Ratnasamy, "Not-a-bot: Improving service availability in the face of botnet attacks," in *USENIX Symp. Networked Systems Design and Implementation*, 2009.
- [33] W. Lu, M. Tavallae, and A. Ghorbani, "Automatic discovery of botnet communities on large-scale communication networks," in *ACM Symp. Information, Computer and Communications Security*, 2009.
- [34] W. Lu, M. Tavallae, G. Rammidi, and A. Ghorbani, "BotCop: An online botnet traffic classifier," in *Communication Networks and Services Research Conf.*, 2009.
- [35] F. Giroire, J. Chandrashekar, N. Taft, E. Schooler, and K. Papagiannaki, "Exploiting temporal persistence to detect covert botnet channels," in *Intern. Symp. Recent Advances in Intrusion Detection*, 2009.
- [36] A. Ramachandran, N. Feamster, and D. Dagon, "Revealing botnet membership using DNSBL counter-intelligence," in *Wksh. Steps to Reducing Unwanted Traffic on the Internet*, 2006.
- [37] HoneyNet Project, "Know your enemy: Fast-flux service networks," The HoneyNet Project and Research Alliance, Tech. Rep., 2008.
- [38] T. Holz, C. Gorecki, F. Freiling, and K. Rieck, "Measuring and detecting fast-flux service networks," in *ISOC Network and Distributed System Security Symp.*, 2008.
- [39] T. Holz and J. Nazario, "As the net churns: Fast-flux botnet observations," in *Intern. Conf. Malicious and Unwanted Software*, 2008.
- [40] E. Passerini, R. Paleari, L. Martignoni, and D. Bruschi, "FluXOR : Detecting and monitoring fast-flux service networks," in *Conf. Detection of Intrusions and Malware and Vulnerability Assessment*, 2008.
- [41] X. Hu, M. Knysz, and K. Shin, "RB-Seeker: auto-detection of redirection botnets," in *ISOC Network and Distributed System Security Symp.*, 2009.
- [42] N. Brownlee, C. Mills, and G. Ruth, "Traffic flow measurement: Architecture," RFC 2722, 1999.
- [43] S. Handelman, S. Stibler, N. Brownlee, and G. Ruth, "New attributes for traffic flow measurement," RFC 2724, 1999.
- [44] Y. Kulbak and D. Bickson, "The eMule protocol specification," School of Computer Science and Engineering, The Hebrew University of Jerusalem, Tech. Rep., 2005.
- [45] M. P. Collins and M. K. Reiter, "Finding peer-to-peer file-sharing using coarse network behaviors," in *European Symp. Research in Computer Security*, 2006.
- [46] G. Bartlett, J. Heidemann, and C. Papadopoulos, "Inherent behaviors for on-line detection of peer-to-peer file sharing," in *IEEE Global Internet Symp.*, 2007.
- [47] S. Racine, "Analysis of internet relay chat usage by DDoS zombies," Master's thesis, Swiss Federal Institute of Technology Zurich, 2004.
- [48] D. Freedman and P. Diaconis, "On the histogram as a density estimator: L2 theory," *Probability Theory and Related Fields*, vol. 57, no. 4, 1981.
- [49] Y. Rubner, C. Tomasi, and L. Guibas, "A metric for distributions with applications to image databases," in *IEEE Intern. Conf. Computer Vision*, 1998.
- [50] G. B. Dantzig, "Application of the simplex method to a transportation problem," in *Activity Analysis of Production and Allocation*. John Wiley and Sons, 1951, pp. 359–373.