Friends of An Enemy: Identifying Local Members of Peer-to-Peer Botnets Using Mutual Contacts

Baris Coskun[*]
AT&T
33 Thomas Street
New York, NY
baris@att.com

Sven Dietrich
Stevens Inst. of Technology
Castle Point on Hudson
Hoboken, NJ
spock@cs.stevens.edu

Nasir Memon
Polytechnic Institute of NYU
Six Metrotech Center
Brooklyn, NY
memon@nyu.edu

ABSTRACT

In this work we show that once a single peer-to-peer (P2P) bot is detected in a network, it may be possible to efficiently identify other members of the same botnet in the same network even before they exhibit any overtly malicious behavior. Detection is based on an analysis of connections made by the hosts in the network. It turns out that if bots select their peers randomly and independently (i.e. unstructured topology), any given pair of P2P bots in a network communicate with at least one mutual peer outside the network with a surprisingly high probability. This, along with the low probability of any other host communicating with this mutual peer, allows us to link local nodes within a P2P botnet together. We propose a simple method to identify potential members of an unstructured P2P botnet in a network starting from a known peer. We formulate the problem as a graph problem and mathematically analyze a solution using an iterative algorithm. The proposed scheme is simple and requires only flow records captured at network borders. We analyze the efficacy of the proposed scheme using real botnet data, including data obtained from both observing and crawling the Nugache botnet.

Categories and Subject Descriptors

C.2.0 [**Computer Communication Networks**]: General—*Security and protection (e.g., firewalls)*; C.2.3 [**Computer Communication Networks**]: Network Operations—*Network Monitoring*

General Terms

Security

Keywords

P2P Botnet, IDS, Network Security

[*]This work was carried out while Baris Coskun was with Polytechnic Institute of NYU

1. INTRODUCTION

Botnets, which are networks of compromised hosts (bots) under the control of a botmaster, have become a major threat in recent years. Botnets are used to perform various malicious activities such as spamming, phishing, stealing sensitive information, conducting distributed denial of service (DDoS) attacks, scanning to find more hosts to compromise, etc. Bots which perform such malicious activity, occasionally go over the radar and get detected by Intrusion/Anomaly/Behavior Detection Systems present within the network. In fact, network administrators routinely discover bots which are then immediately quarantined or removed. However, some interesting and important questions remain, such as: ***"Does the network contain more bots of the same type which haven't been exposed yet?" "Can the discovered bot be leveraged to find other dormant bots of the botnet before they commit any malicious activity?" "Can all this be done without any access to backbone traffic and only from netflow data from the edge router?"***

A common and fairly obvious approach to find dormant bots is to characterize the Command and Control (C&C) channel from the discovered bot's recent traffic and identify hosts that exhibit similar C&C traffic characteristics. For example, botnets with a centralized C&C architecture, where all bots receive commands from a few central control servers, the source of the C&C messages can be used to characterize the corresponding C&C channel and reveal potential dormant bots [23].

However, characterizing the C&C channel is in general not a trivial task for botnets that utilize a *peer-to-peer* (P2P) architecture involving no central server. For example, this kind of source analysis falls short for P2P botnets as here the botmaster can use any node to inject C&C messages. To receive and distribute C&C messages, each P2P bot communicates with a small subset of the botnet (i.e. peer list) [30, 14, 18] and maintains its own peer list independently. Hence, no obvious common source of C&C messages can be observed, thereby preventing the linking of the discovered bot with the dormant bots. Furthermore, features based on packet sizes and timings, such as packets per flow, bytes per flow, flows per hour, etc. may not be useful in characterizing a C&C channel, since botmasters can easily randomize such features thereby obtaining different feature values for each bot [29]. Botnets such as Nugache, Storm, Waledac and Conficker employ advanced encryption mechanisms [30, 14, 18, 28, 27] making characterization based on packet contents infeasible.

In this paper we propose an efficient technique to discover additional P2P bots in a network after one such bot has been discovered. Specifically, the proposed technique provides a list of hosts ordered by a degree of certainty, that potentially belong to the same P2P botnet as the discovered bot. Network administrators can use this list as a starting point of their investigation and potentially identify more bots in their network once they discover one. The proposed technique is based on the simple observation that peers of a P2P botnet communicate with other peers in order to receive commands and updates. Although different bots may communicate with different peers, we show that for P2P botnets with an unstructured topology, where bots randomly pick peers to communicate with, there is a surprisingly high probability that any given pair of P2P bots communicate with at least one common external bot during a given time window. In other words, there is a significant probability a pair of bots within a network have a **mutual contact**. We present a precise mathematical derivation of this probability as a function of the size of a botnet and the number of peers a bot contacts. Notice that we focus on P2P botnets with unstructured topology in this work and the term "P2P botnet" refers to unstructured P2P botnets in the rest of the paper unless stated otherwise.

In order to discover dormant bots, we first construct a mutual contacts graph where every host is a node and two nodes share an edge if they share a mutual contact. The weight or capacity on an edge is the number of mutual contacts shared between the corresponding hosts incident on the edge. Then given a discovered bot or seed bot, we present an iterative algorithm, which identifies other potential members of the botnet by iteratively computing a level of confidence to each host on the graph. We declare the hosts which have confidence levels higher than a threshold as potential members of the same P2P botnet as the seed-bot. We present experimental results with real and simulated traffic to measure the effectiveness of our technique. We also present mathematical analysis characterizing the structure of a mutual contact graph.

In addition to being simple and effective the proposed scheme has the following desirable properties:

- The proposed method is not an anomaly detection scheme and hence doesn't require P2P bots to exhibit any overtly malicious activity.
- Similarly, it is not a behavior clustering algorithm and therefore doesn't require any common behavior exhibited by all the bots.
- It utilizes the pairwise mutual-contact relationships between pairs of bot peers, which arise due to P2P C&C communications. We validate the existence of such relationships both mathematically and experimentally.
- The proposed method is generic and doesn't depend on specific properties of specific botnets. Therefore, it doesn't require reverse engineering bot binaries or C&C protocols [3].
- Contrary to existing graph-based network traffic analysis methods [26] [19], the proposed method doesn't require any access to backbone traffic. Mutual-contact relationships are deduced locally at an edge router.

In the rest of this paper, we explain the basic idea and details of the proposed method in Section 2. Following that,

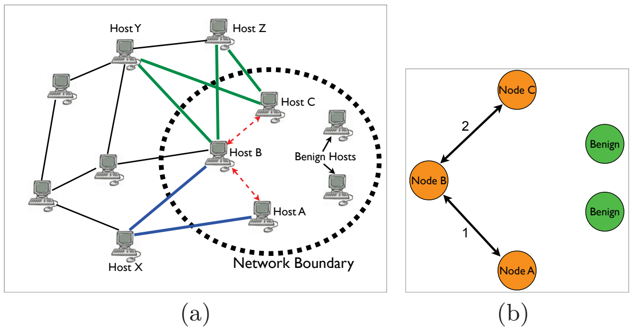


Figure 1: Illustration of P2P Botnet communications for a network (a) and its corresponding mutual-contact graph (b). The network contains 2 benign hosts and 3 bots (Hosts A, B, and C). The bots are members of a P2P botnet with 9 bots in total. Mutual-contact relationship among hosts, which is indicated by red dashed arrows in (a), are represented by the mutual-contact graph in (b). The edge capacities are determined by the number of mutual contacts between nodes.

we present our experimental results with the Nugache botnet in Section 3. In Section 4, we present a mathematical analysis that provides insights on why the proposed scheme works and its limitations. Then in Section 5, we discuss practical limitations of the proposed scheme, possible evasion techniques and their implications on P2P botnets. We present the related work in Section 6. Finally, we conclude the paper and discuss future work in Section 7.

2. FINDING FRIENDS OF AN ENEMY

In this section, we present the basic idea and the details of the proposed algorithm. We first begin with an intuitive explanation in the next subsection and then provide a more detailed and formal explanation in subsequent subsections.

2.1 Basic Idea

Consider the botnet illustrated in Figure 1(a). The basic idea of the proposed method is that, *Host A* can be linked to *Host B* since they both communicate with *Host X* (the mutual contact). Similarly *Host B* and *Host C* are linked together through *Host Y* and *Host Z*. As a result, if *Host A* becomes known as a member of a P2P botnet, then by examining its connections, one may suspect that *Host B* is likely to be a member due to the presence of a mutual contact with the known bot *Host A*. Similarly, if *Host B* is likely to be a member, then *Host C* is likely to be a member as well.

Now it is clear that, aside from P2P botnet traffic, legitimate traffic probably includes several mutual-contacts among hosts as well. For instance, there are some very popular servers that almost every host in the network communicates with such as *google.com*, *microsoft.com*. etc. As a result, every host in the network would be linked to most of the other hosts through such popular mutual-contacts. However, if *H*ost A is a known bot and both *H*ost A and *H*ost B have been in communication with *H*ost X, and *H*ost X has not talked to almost anyone else within our network, then it is likely that *H*ost B is a member of the same botnet as *H*ost A. Hence in our mutual contact based analysis we restrict ourselves to **private mutual-contacts**. Private mutual contacts are mutual contacts which communicate with less than k internal hosts during an observation window. Here, k is the **privacy-threshold**. Private mutual contacts capture the intuition that it is very unlikely that external peers which are part of a botnet will be communicating with many internal hosts that do not belong to the botnet. Therefore, private mutual-contacts can be strong indicators of peer relationships among hosts within a botnet. **In the rest of this paper, we use the term mutual-contacts to mean private mutual contacts.**

The question then remains that given a known bot, how do we systematically rank all the hosts in our network based on their likelihood of being a member of the same P2P botnet using private mutual contact relationships they exhibit? To do this, we first extract mutual-contacts from the flow records captured at the network border for a time window prior to discovering the seed-bot. We then represent the mutual-contact relationships among hosts by a directed graph called the *mutual-contacts graph*, such that: 1. *Nodes represent the hosts in the network.* 2. *There is a bidirectional edge between two nodes if the corresponding hosts have at least one mutual-contact during the given time window.* 3. *Each edge has a capacity determined by the number of mutual-contacts between corresponding nodes.*

As an example, the mutual-contact graph for the network illustrated in Figure 1(a) is shown in Figure 1(b). Now intuitively speaking, it is expected that hosts which are likely to be P2P bots are at a short distance from the seed-bot on a mutual-contacts graph since such hosts are expected to have mutual-contacts with the seed-bot itself and/or with other hosts which have mutual-contacts with the seed-bot. In fact, we observe this behavior in various real world botnets as presented later in Table 1. Furthermore, the more the mutual contacts that a host has with the seed bot and other suspected bots, the more likely it is that this host is also a bot. The mutual contacts graph illustrated in Figure 2(a) displays such behavior (black edges). Based on these two intuitions, we propose a scheme that iteratively computes a confidence level of being a member of the same P2P botnet as the seed bot for each node. This iterative process can be illustrated as pumping red dye into the mutual-contacts graph from the node representing the seed-bot as depicted in Figure 2(b). During the process, the dye coming to a node is distributed across its outgoing edges proportional to their capacities. Therefore, the dye accumulated in a node reflects our confidence for that host being a part of the same botnet as the seed-bot. Inspired by this illustration, we named our proposed algorithm the "Dye-Pumping Algorithm".

In Figure 2(b), it is also observed that along with the P2P bots, few benign hosts also share mutual-contacts with P2P bots (via green edges in Figure 2(a)), and therefore receive some amount of dye. Such hosts potentially result in false positives. However, we expect the capacity of the edges connecting these benign hosts to P2P bots to be usually lower thereby keeping the dye accumulated on these benign hosts below a threshold in most cases. In later sections we provide detailed experimental and mathematical analysis, that supports our intuition that a majority of the false positives can be eliminated while maintaining reasonable false negatives, by choosing a suitable threshold. But first, in the following subsections, we present each step of this algorithm in greater detail.

2.2 The "Mutual-Contacts" Graph

We denote the mutual-contacts graph constructed from the flow records of a network by $G = (N, E)$, where the nodes correspond to hosts and the edges indicate private mutual-contacts. Each edge on the graph has a capacity which is determined by the exact number of mutual-contacts between corresponding hosts. More formally, if E_{ij} represents the capacity of the edge between nodes N_i and N_j, then we can write:

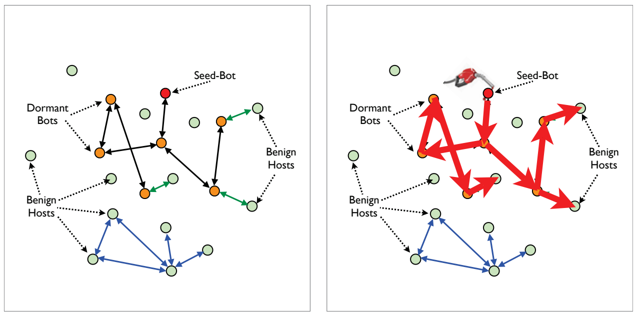


Figure 2: (a) Illustration of a mutual-contacts graph. P2P bots tend to share mutual-contacts with each other (black edges). Also some benign hosts share mutual-contacts among themselves (blue edges), which may be due to a legitimate P2P application. (b) Illustration of the dye-flow in the Dye-Pumping algorithm.

$$E_{ij} = E_{ji} = |S(N_i) \cap S(N_j)|$$

where $S(N_i)$ represents the set of mutual-contacts which N_i was in communication with during the observation period and $| \cdot |$ represents the cardinality of a set. Notice that, if nodes N_i and N_j don't share any mutual-contacts then there is no edge between them on the graph or equivalently $E_{ij} = 0$.

2.3 The "Dye-Pumping" Algorithm

Once the mutual-contacts graph is constructed, the dye-pumping algorithm is executed to compute the confidence levels of hosts being part of the P2P botnet. The dye-pumping algorithm iteratively pumps dye to the mutual-contacts graph from the seed node and picks the nodes which accumulates more dye than a threshold. During the process, dye coming to a node is distributed to other nodes proportional to a heuristic called the **dye-attraction coefficient**, which helps the algorithm to funnel more dye towards the nodes which are more likely to be P2P bots.

Dye-Attraction Coefficient is denoted by γ_{ji}, and indicates what portion of the dye arriving at Node j will be distributed to Node i in the next iteration. Intuitively, it represents our confidence on Node i being a P2P bot given that Node j is a P2P bot. Such confidence gets higher as Node i and Node j share more private mutual-contacts with each other. On the other hand, our confidence reduces if Node i shares mutual-contacts with many other nodes in the graph. The reason is that we expect to have few bots in our network and therefore if a host shares mutual-contacts with many other hosts, then these mutual-contacts are probably due to a different application other than botnet C&C. Consequently, we compute the dye-attraction coefficient from Node j to Node i as follows:

$$\gamma_{ji} = \frac{E_{ji}}{(D_i)^{\beta}}$$

where D_i is the degree of Node N_i (i.e. number of neighbors or edges that N_i has) and β is the **Node Degree Sensitivity Coefficient**. Basically, nodes with high degrees receive less and less dye as β increases.

The Dye-pumping Algorithm has three inputs, namely the edge capacities (E_{ji}) of the mutual-contacts graph ($\mathbf{E}$ represents the matrix containing all E_{ji} values), the index (s) of the seed node N_s, and the number of iterations ($maxIter$). Given these inputs, the dye-pumping algorithm first computes the dye-attraction coefficients from edge capacities and forms the transition matrix $\mathbf{T}$ such that:

$$\mathbf{T}(i,j) = \gamma_{ji} = \frac{E_{ji}}{(D_i)^\beta}$$

where $i = 1, ..., v$ and $j = 1, ..., v$. Also $\mathbf{T}(i,j) = 0$ if $i = j$. Notice that the transition matrix of a mutual-contacts graph with v nodes is a $v \times v$ square matrix.

Following that, the algorithm normalizes $\mathbf{T}$, so that each of its columns sums to 1 (i.e. stochastic matrix). If $\overline{\mathbf{T}}$ indicates the normalized transition matrix, the normalization procedure can be written as $\overline{\mathbf{T}}(i,j) = \frac{\mathbf{T}(i,j)}{\sum_{i=1}^{v} \mathbf{T}(i,j)}$. After normalization, the algorithm iteratively pumps dye to the mutual-contacts graph from the seed node. For this purpose, let the column vector L is the dye level vector, where $L(i)$ indicates the dye level accumulated at node i. The pumping begins with filling the seed node with dye and leaving the others empty such that:

$$L(i) = \begin{cases} 1, & \text{if } \ s = i \\ 0, & \text{elsewhere} \end{cases}$$

Once the seed node is filled with dye, the algorithm pumps dye from the seed node across the mutual-contacts graph. Since the outgoing edges distribute the dye accumulated within a node proportional to their capacities, the dye levels at next iteration can be computed as:

$$L(i) = \sum_{j=1}^{v} \overline{\mathbf{T}}(j,i)L(j)$$

which can be also written in matrix form as $L = \overline{\mathbf{T}}L$. At each iteration, after updating L, the algorithm pumps more dye to the graph from the seed node by updating $L(s) = L(s) + 1$. Following that the vector L is normalized after each iteration as $L = \frac{L}{\sum_{i=1}^{v} L(i)}$. Finally after $maxIter$ iterations, the dye-pumping algorithm outputs the dye-level vector L. The steps of the dye-pumping algorithm are summarized below:

Algorithm 1 $Dye\_Pumping(\mathbf{E}, s, maxIter)$

```
1: $\mathbf{T} \leftarrow computeTransitionMatrix(\mathbf{E})$
2: $\overline{\mathbf{T}} \leftarrow normalize(\mathbf{T})$
3: $L \leftarrow [0, 0, ..., 0]^{tr}$ {initialize $L$ as a zero vector}
4: for $iter = 1$ to $maxIter$ do
5:     $L(s) \leftarrow L(s) + 1$ {Pump dye from the seed node}
6:     $L \leftarrow \frac{L}{\sum L(i)}$ {Normalize dye level vector}
7:     $L \leftarrow \overline{\mathbf{T}}L$ {Distribute dye in network for one iteration}
8: end for
9: output $L$
```

Once the algorithm outputs the vector L, the dye level of each node ($L(i)$) indicates the confidence level for the corresponding host being a member of the same P2P botnet as the seed node. To have a more conclusive result, we set a threshold *thr* such that the nodes having a dye level greater than *thr* are declared as potential members of the same P2P botnet as the seed bot.

Notice that the algorithm involves a constant number of matrix multiplications. Hence, the complexity of a naive implementation of the algorithm is cubic in the number of nodes. However, both dye-level vector (L) and transition matrix ($\mathbf{T}$) are sparse. Therefore one can implement the dye-pumping algorithm asymptotically faster by using fast sparse matrix multiplication techniques.

3. EXPERIMENTS

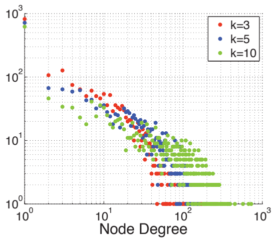
3.1 Detecting Nugache Peers

In order to systematically assess the performance of the proposed scheme against a real-world botnet, one needs to know the IP addresses of the members of a P2P botnet in a given network. Otherwise, nothing can be said about the true positive or false alarm rate without knowing the ground truth. One way to obtain the ground truth is to blend real botnet data into the network traffic and make few hosts look as if they have been infected by the botnet. This strategy essentially aggregates real botnet traffic and real user traffic on some of the hosts and therefore provides a realistic scenario. From the proposed scheme's perspective, to make a host look like a P2P bot, one can first capture the flow records of the network, which contains the host, during a time window. Then one can collect the flow records form a real P2P bot during a similar time window. Following that, one can change the bot's IP address in these botnet flow records to a selected host's IP address and append them to the flow records of the entire network so that, along with its original traffic, the selected host will appear as if it has also communicated with the external IP addresses that the real bot has talked to.

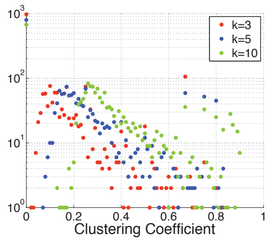
In order to establish the ground truth for our experiments, we utilize the data collected from the Nugache botnet, which has been thoroughly studied in [30][8]. Briefly speaking, Nugache is a P2P botnet that uses random high-numbered ports for its communication over TCP. The data we use in our experiments was compiled by the Nugache crawler presented in [10] and its communication between Nugache peers.

Nugache Botnet Data: Details on the Nugache botnet and Nugache crawler can be found in [30] and [8]. In summary, the C&C protocol of Nugache enables querying a peer for its list of known peers and a list of recently communicated peers. Using this functionality, the crawler starts from a series of seed peers and traverses the botnet by querying peers for their list of known peers. The crawler maintains the list of recently communicated peers for each accessible Nugache peer. Consequently, when it finishes crawling, it produces list of recently communicated peers for several Nugache peers.

In our experiments, we used the data collected by the crawler when Nugache was active. To collect data, the crawler was executed repeatedly for 9 days, where each execution lasted roughly 30 to 45 minutes. We used a 24-hour observation window for our experiments. Hence, we employed several randomly selected 24-hour segments of the crawler data from the 9-day results in our experiments to cover the botnet dynamics during all 9 days. We observed that in any of these 24-hour segments, 904 Nugache peers responded to the crawler on an average. We also observed that 34% of all possible pairs of Nugache peers communi-



(a) Node Degree Dist.



(b) Clustering Coeff. Dist.

Figure 3: Node Degree and Clustering Coefficient distributions of the mutual-contacts graph of the background traffic for different privacy threshold (k) values.

cated with at least one mutual-contact on average.

Background Traffic: In order to obtain background traffic that we could blend with Nugache traffic, we captured the flow records observed at the border of Polytechnic Institute of NYU network during a typical weekday (i.e. the observation window is 24 hours). Collected flow records indicate that there were 2128 active IP addresses in our network during the observation window. We then extracted mutual-contacts from the recorded data. To ensure a valid communication (i.e. not a scan flow), we only considered external IPs which exchanged sufficient amount of data (i.e. at least 256 bytes) in both directions with at least one internal IP. Finally we built the corresponding mutual-contacts graph to serve as a basis for our experiments.

We immediately observed in the mutual-contacts graph that DNS servers within the network shared a significantly large number of mutual-contacts with each other. As a matter of fact, DNS servers constituted the highest-magnitude entries of the first eigenvector of the matrix ($\mathbf{E}$) whose entries are the corresponding edge capacities (E_{ij}). This is not surprising since DNS servers in a network communicates with many other DNS servers around the world. Obviously this relationship among DNS servers dominates the mutual-contacts graph and taints the results of the Dye-Pumping algorithm. Hence, we removed all the edges of the 11 DNS servers in the network from the mutual-contacts graph.

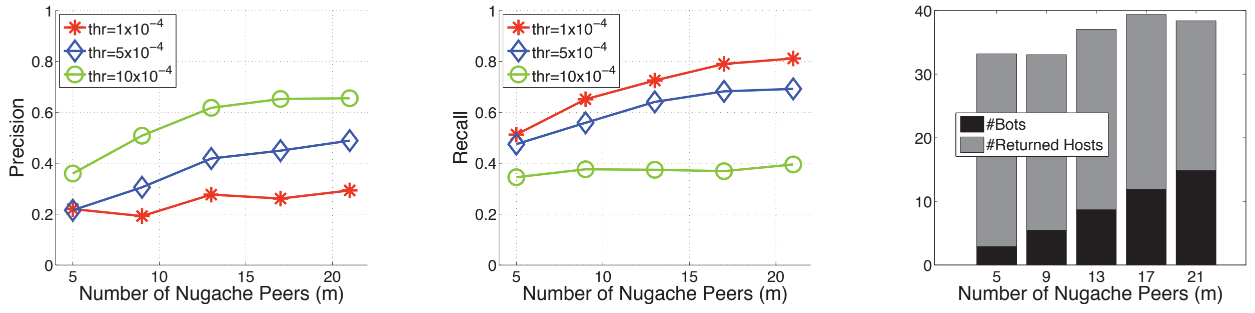
The mutual-contacts graph extracted from the background traffic suggests that majority of the hosts share none or very few mutual-contacts with other nodes. This can be observed in Figure 3(a), where we plot the distribution of node degrees (i.e. no of edge of a node). Figure 3(a) also shows, as expected, that nodes usually have a higher degree in the mutual contact graph when a higher privacy threshold (k) value is used to construct the graph.

We then looked at the clustering coefficient, which is defined as the ratio of the number of the actual edges of a node to the number of all possible edges among it's neighbors. We plot the clustering coefficient distribution of the nodes in Figure 3(b). We observe that the mutual contact-graph is a lot more clustered than a comparable random graph (i.e same number of nodes and edges). For instance the clustering coefficient distribution of a random graph comparable to the mutual-contacts graph with $k = 5$ has a mean of 0.006 and standard deviation of 0.009. This suggests that there are communities of hosts in the observed network where community members usually communicate with the same external IPs that are exclusive to the corresponding community. One can speculate that these communities may represent peers of different P2P networks (legitimate or bot) or a group of users visiting similar websites etc.

Experiments with Nugache: In order to assess the performance of the proposed scheme in detecting Nugache bots, we randomly picked m Nugache peers from a randomly selected 24-hour segment of the crawler data. Then, we computed the mutual-contacts graph corresponding to these m Nugache peers based on the recently-communicated peers field of the crawler data. We then randomly picked m internal hosts from the mutual-contacts graph corresponding to the background traffic. Finally, we superposed the mutual-contacts graph of the Nugache peers onto in the mutual-contacts graph of the background traffic where m Nugache peers coincide with m selected internal hosts. This procedure essentially blends Nugache traffic into the background traffic so that each of these m selected internal hosts looked as if they communicated with the peers that the corresponding m Nugache peers communicated with. Consequently, each of these m selected hosts becomes a real Nugache peer and constitutes the ground truth as far as the proposed scheme is concerned.

Once we obtained the superposed mutual-contacts graph, we randomly selected one of the m hosts as the seed bot and ran the Dye-Pumping algorithm to detect the other $m-1$ hosts whose flow records were modified according to the Nugache crawler data. We set the number of iterations to $maxIter = 5$ for Dye-Pumping algorithm since it is almost impossible to find P2P botnet peers more than 3 hops away from the seed node due to the Erdős-Rényi model as will be explained in Section 4. In the end, we returned the list of hosts which accumulate more dye than the threshold as P2P bots. To obtain statistically reliable results, we repeated the experiment 100 times, each time with different selection of m hosts and m Nugache peers. We also picked a different 24-hour segment of crawler data at every 20th repetition.

Results (Precision & Recall): To gauge the algorithm's performance, we computed the average precision and recall. In our context, precision can be defined as the ratio of the number of Nugache peers in the returned list of hosts to the length of the returned list. On the other hand, recall can be defined as the ratio of the number of Nugache peers in the returned list to the number of all Nugache peers in the network except the seed bot ($m - 1$). Figure 4 presents the average precision and recall values for different number of Nugache peers (m) and different threshold values (thr). We set the privacy threshold $k = 5$ and node degree sensitivity coefficient $\beta = 2$. It is observed that several dormant Nugache peers can be identified by the proposed technique when the threshold is set to an appropriate value. For instance, in Figure 4(c) we observe that, if there are 17 Nugache peers in the network, the proposed scheme on average returns 35 hosts, 11 of which are Nugache peers. As a result, upon obtaining the list of potential P2P bots, a network administrator can perform a more detailed investigation (perhaps physically) on the hosts in the list and potentially uncover several dormant P2P bots. Meanwhile, the returned list also contains some hosts which are not Nugache peers since such hosts happen to be connected to one or more Nugache bots on the mutual-contacts graph due to mutual-contacts created by other applications. Interestingly, it is observed in Figure 4 that both precision and recall values increase as the number of bots (m) increases. This is due to a property



(a) Precision (b) Recall (c) Returned host counts,thr=$5x10^{-4}$

Figure 4: Experiment results with Nugache. The parameters are $k = 5$ and $\beta = 2$

of Erdős-Rényi random graphs that—*as will be explained in the next section*— the probability of having a short path between two nodes increases with the number of nodes. It is also observed that, increasing the threshold increases precision but decreases recall, as is naturally expected for any detection system.

Effects of Privacy Threshold (k): When we repeated the experiments for different k values, we did not observe a major change in the precision performance. On the other hand, we observed, as shown Figure 5(a), that recall performance improves as we decrease k as long as the number of P2P bots in the network is low. The recall performance improves because more background traffic is filtered out for lower k, thereby removing a significant portion of the benign edges. However, if there are many P2P bots in the network and if k is small (i.e. $k = 3$), more than k of them are likely to communicate with several common external peers and therefore some of the botnet communications are likely to be filtered out as well. The effect of this phenomenon can be observed in Figure 5(a), where recall performance diminishes for large number of Nugache peers. Hence, based on Figure 5(a) we can say that $k = 5$ was an appropriate setting for our network.

Effects Node Degree Sensitivity Coefficient (β): As explained in Section 2.3, larger β values result in less dye-flow towards the nodes which have high degrees on a mutual-contacts graph. We would like to restrict the dye-flow to high-degree nodes, because edges between bots and high-degree nodes are probably not due to botnet communications but rather due to some other application which causes many of the edges that high-degree nodes have. Larger β values cause the dye to concentrate around the seed-bot and therefore improve the precision performance as observed in Figure 5(b). On the other hand, since the algorithm cannot reach far in the mutual-contacts graph for larger β values, the recall performance drops as β gets larger as observed in Figure 5(c). According to our experiments, $\beta = 2$ turned out to be an appropriate setting for our network.

In summary, different values of the parameters k and β yield a tradeoff between precision and recall. When deploying the proposed scheme, a network administrator should first decide on the minimum tolerated precision level and then set the parameters accordingly. For this purpose, artificial P2P botnet traffic generated by the Random Peer Selection model described in Section 4.1 could be used as a ground truth to determine which parameter values would result in which precision levels for a given network.

4. MATHEMATICAL ANALYSIS

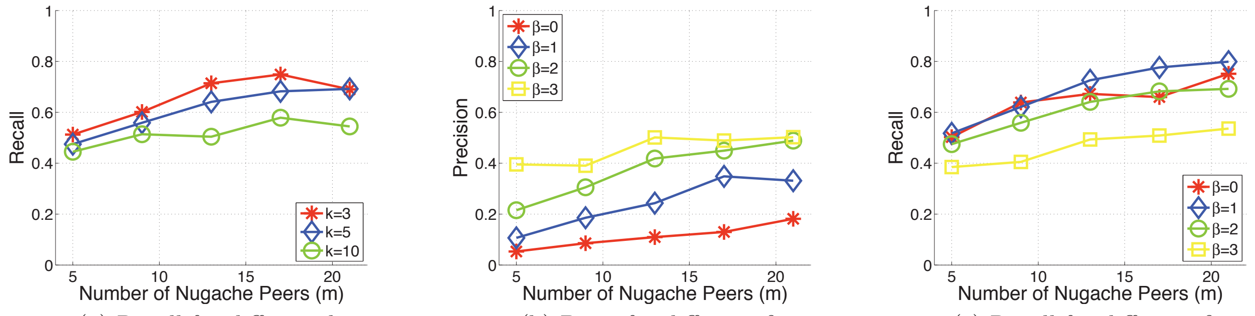
The essence of the proposed algorithm is that the members of a P2P botnet tend to have mutual-contacts and therefore are closely connected on a corresponding private mutual-contacts graph. In fact, the dye-pumping algorithm performs better if P2P bots in a network are connected to the seed node through shorter and higher-capacity paths, which yield higher volume of dye flow from the seed node to the other bots. Although our experimental results in the previous section tend to validate our intuition, some significant questions remain to be addressed to mathematically validate the approach and show its applicability to the general problem that goes beyond specific instances of P2P botnets. Question such as how likely is it that two peer bots will have a mutual contact? How does this probability vary with the size of the botnet and the number of peers contacted by each bot. Next, how likely is it that the mutual contact graph will have a connected component that spans peer bots? How does one characterize the properties of the mutual contacts graph? In this section we address these questions and present a mathematical analysis that supports our approach and validates the experimental results reported in the previous section.

4.1 Random Peer Selection Model

The first question we posed was the likelihood of peer bots having a mutual contact. But before we answer that, we would like to first justify the framework in which we examine this question. Recall that our framework assumes that bots independently and randomly select the peers with which they communicate. How does this assumption bias our analysis? In this subsection we address this question and argue that this represents the worst case situation for our analysis.

In a P2P network some peers might be more available than others and therefore they have a higher probability of being selected by other peers [14][18] [21] [1]. Obviously, having such preferred peers in a P2P botnet increases the chance finding mutual-contacts between P2P bots in a network. However, the worst case, as long as unstructured P2P botnets are considered, from our work's point of view is when there is no preferred peer in the botnet and all peers have equal probability of being contacted by any other peer, thereby minimizing the probability of private mutual-contacts between peers.

To investigate the probability of mutual-contacts in the worst case, we consider a generic botnet model, where each



(a) Recall for different k (b) Prec. for different β (c) Recall for different β

Figure 5: Effects of different parameters. The non-varying parameters are set to $k = 5$, $\beta = 2$ and $thr = 5 \times 10^{-4}$

bot picks peers independently and randomly. The model has two configurable parameters such that; "B" is the number of all peer in the botnet and "C" is the number of peers that each peer communicate with during a specific observation window. Based on these parameters, each bot (b_i) in the model communicates with a uniform random subset ($\mathcal{S}_\mathbf{i}$) of all $B-1$ available bots (excluding itself) in the model, where the cardinality of each subset is C.

Bot-Edge Probability: Having justified our framework, we now address the question about the probability of two peer bots having a mutual contact. In the random peer selection model, the probability of having an edge between two arbitrary bots b_i and b_j (i.e. bot-edge probability, p_e) is actually the probability of the intersection of the corresponding subsets being non-empty; such that $p_e = Pr(\mathcal{S}_\mathbf{i} \cap \mathcal{S}_\mathbf{j} \neq \emptyset)$. Since the number of elements in the intersection of two uniform random subsets can be computed using hypergeometric distribution, we can write the bot-edge probability as:

$$p_e = 1 - \frac{\binom{C}{0}\binom{B-1-C}{C}}{\binom{B-1}{C}} \qquad (1)$$

Bot-edge probabilities are plotted in Figure 6(a). It is observed that, similar to the Birthday Paradox, as the number of contacted peers increases, the bot-edge probability increases very rapidly. Consequently, even for a fairly large botnet with $50k$ peers, the bot-edge probability is almost 0.5 when peers contact only 200 other peers during the observation window.

Bot-Edge Capacity: Although, high bot-edge probabilities works in favor of the dye-pumping algorithm, the capacities of those edges are also important. It is obvious that, the higher the bot-edge capacities the better the dye-pumping algorithm performs. In the random peer selection model, the probability of a peer contacted by two given peers is $\left(\frac{C}{B}\right)^2$. Therefore, since there are B peers in total, we can write the expected capacity of bot edges ($E[Cp]$) as:

$$E[Cp] = \left(\frac{C}{B}\right)^2 B = \frac{C^2}{B} \qquad (2)$$

which is also the expected value of the corresponding hypergeometric distribution. Figure 6(b) plots the expected bot-edge probabilities. It is observed that, regardless of the botnet size, expected bot-edge capacity rapidly exceeds 1 and continues to increase as the number of contacted peers increases. Figure 6 suggests that the members of a P2P botnet will most probably be well connected with each other on a private mutual-contacts graph through high capacity edges, thereby allowing the dye-pumping algorithm to identify them.

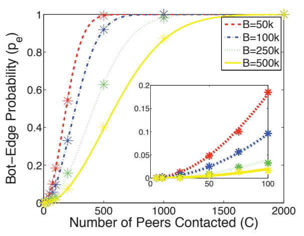
4.2 Friends Stay Closely Connected (Erdős-Rényi Subgraphs)

Having established that it is quite likely that two peer bots will have a mutual contact we now turn our attention on the expected structure of the mutual contacts graph. After all, the Dye-Pumping algorithm can only identify the P2P bots which are connected to the seed-bot via short paths on the mutual-contacts graph. Bots which are isolated from the seed-bot cannot be accessed by the algorithm. In this subsection, given a bot-edge probability, we investigate how the P2P bots are expected to be oriented on a private mutual-contacts graph and what portion of the P2P nodes can be accessed by the dye-pumping algorithm.

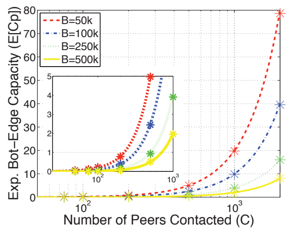
To understand the structure of the subgraph formed by members of a P2P botnet on a mutual-contacts graph, suppose that there are m bots in the network, and therefore the corresponding m nodes on the graph. Let the set $X = \{X_1, X_2, ..., X_m\}$ denote these nodes and p_e denote the probability of having an edge between any given X_i and X_j, for $i \neq j$ where $1 \leq i \leq m$ and $1 \leq j \leq m$. Since p_e is the same for any pair of X_i and X_j, the subgraph formed by the nodes$X_1, X_2, ..., X_m$ on a private mutual-contacts graph is an *Erdős-Rényi random graph* [11][12], where each possible edge in the graph appears with equal probability.

One interesting property shown by Erdős and Rényi is that, Erdős -Rényi graphs have a sharp threshold of edge-probability for graph connectivity [12]. More specifically, if the edge-probability is greater than the threshold then almost all of the graphs produced by the model will be connected. Erdős and Rényi have shown the sharp connectivity threshold is $\frac{\ln \theta}{\theta}$, where θ is the number of nodes in the graph. Therefore, if the bot-edge probability of a P2P botnet is $p_e = \frac{\ln m}{m}$, then the dye-pumping algorithm potentially identifies all other P2P bots from a given seed bot with high probability as long as there are more than m bots in the network. In other words, it gets easier for the proposed method to reveal P2P bots as the botmaster infects more hosts in the network. However, even if the bot-edge probability is below the threshold, the dye-pumping algorithm can still identify some of the P2P bots, which happen to be connected to the seed node on the private mutual-contacts graph.

In conclusion, according to the random peer selection model, members of a P2P botnet are expected to be closely connected to each other on a private mutual contacts graph despite large botnet sizes.



(a) Bot-Edge probability



(b) Expect.value of bot-edge capacity

Figure 6: Properties of random peer selection model for different botnet sizes (B) and different number of contacted peers (C) are plotted in Figure 6(a). Solid lines indicate the theoretical computation and the stars point the empirical estimation. Inner figures magnifies the region where $0 < C < 100$

5. LIMITATIONS AND POTENTIAL IMPROVEMENTS

The proposed method is able to identify P2P bots in a network as long as they are clustered through short and high capacity paths on a private mutual-contacts graph. Therefore, botmasters need to disturb this clustering structure in order to evade the proposed method. In this section, we review these possible evasion strategies, and their implications on the creation and maintenance of P2P botnets.

Eliminating Private Mutual-Contacts: One way to eliminate private mutual contacts is by increasing the popularity of private mutual-contacts that P2P bots in a network communicate with. If their popularity gets higher than the privacy threshold (k), they will be omitted by the proposed scheme and will not result in edges in private mutual-contacts graphs. However, in order to achieve this, a botmaster has to control more than k hosts in that particular network, so that they can collectively boost a contact's popularity beyond the privacy threshold. To defend against this strategy, the privacy threshold (k) needs to be set as large as possible. Although, as discussed in Section 3, high k values impairs the recall performance of the proposed scheme, for smaller networks it is often possible to find an appropriate k value since a botmaster is unlikely to have too many bots in a small network. On the other hand, for large networks which potentially contain many P2P bots, the proposed technique can be applied on smaller subnets separately and independently to increase the likelihood that the number of P2P bots in each subnet remain below the privacy threshold.

Decreasing The Probability of Mutual-Contacts: Decreasing the probability of observing mutual-contacts between P2P bots is equivalent to decreasing the bot-edge probability (p_e). As discussed in Section 4, a botmaster has to either(or both) increase the botnet size (B) or decrease the number of peers that each bot communicates with (C) in order to lower p_e. It is clear that increasing B and decreasing C will inversely affect a P2P botnet's robustness and efficiency. Although it may be possible for a botmaster to pull p_e down to a lower value, we observed in a controlled environment that peers of today's botnets such as Storm and Waledac have very high bot-edge probabilities. To collect data for Storm and Waledac, we infected two Pentium IV, 512MB RAM Windows XP hosts, which were completely isolated from the rest of the network by a firewall. The firewall was also set to block all SMTP traffic to prevent any spam traffic. We observe that both Storm and Waledac communicate with fairly high number of unique peers during 24 hours, and therefore create many mutual-contacts as presented in Table 1. On the contrary, Nugache peers are less active and create far less mutual-contacts as observed in Table 1. Nevertheless, in Section 3, the proposed scheme is shown to successfully detect several Nugache peers, which are introduced to the network using the crawler data, despite their low communication activities. To collect data for Nugache, the bots were installed on a Pentium IV, 1GB RAM, running VMware Server with a Windows XP guest, as well as on bare metal machines on comparable hardware running Windows XP. The traces were captured within the protected network using a customized honeywall [32] and also using full-packet capture on an extrusion prevention system running OpenBSD with strict packet filter rules, as described in [10] The captured packets were converted to flow records using the SiLK tools [4] for establishing mutual contact sets and validating the algorithm.

Table 1: Summary of observed P2P botnet behavior. Δ : Average number of unique IP addresses that a bot communicates with each day. $\bigcirc$: the number of mutual-contacts (the bot-edge capacities) between the two bots during 24 hours.

	Day 1		Day 2		Day 3	
	Δ	$\bigcirc$	Δ	$\bigcirc$	Δ	$\bigcirc$
Storm	5180	2861	4681	2886	4022	2323
Waledac	1145	341	775	300	1012	358
Nugache	45	0	53	1	49	0

Using a Structured P2P Topology: A botmaster can adopt a structured P2P topology to decrease the probability of mutual contacts by making peers in a same network to communicate with different sets of peers from each other. To achieve this, peers in a same network have to coordinate with each other so that they won't communicate with the peers in each other's peer list. In some sense, peers in a same network have to form their own tiny botnet among themselves and appear as a single node to the remaining of the P2P botnet. These intra-network communications among the peers in a same network, however, would potentially yield new means of detecting P2P bots in a network. Nevertheless, even if a botmaster manages to deploy a mutual-contact-free P2P architecture, two or more networks can choose to share their flow records to exploit the mutual-contacts among P2P bots in different networks, which are unavoidable since the botmaster cannot know which networks would collaborate in the first place. For such mitigation strategies, cooperating networks can use privacy-preserving set operations such as [7] to share data between networks without revealing any sensitive information.

Poisoning Clusters: The purpose of cluster poisoning for P2P networks is to destroy clustering structure of a graph by creating bogus edges [5]. Cluster poisoning appears to be very hard to achieve in our context. In order to perform poisoning, a botmaster has to create an edge between a P2P bot and a benign node on a mutual-contacts graph. For this purpose, she needs to make both the bot and the benign host communicate with a mutual external IP. To do so, the botmaster has to listen to the traffic of the benign host and make the P2P bot contact with an external host which the benign host has communicated with. But this is not a trivial task for a botmaster, unless she also possesses a router or a proxy in the same network.

6. RELATED WORK

Early botnets employed centralized *command and control* (C&C) servers to distribute commands and updates to individual bots, usually through IRC or HTTP protocols [9]. Although a centralized structure is simple and easy to manage, it suffers from a single point of failure and is susceptible to traditional defenses such as domain revocation, DNS redirection, blacklisting etc. Therefore, botmasters have begun to adopt a P2P architecture for C&C channels. In [20], authors argue that it is harder to detect P2P bots especially with a limited view of their traffic from a single Autonomous System. In P2P botnets each bot acts both as a server and a client allowing botmasters to publish commands and updates from any point in the botnet[14][18]. In [6], authors investigate the effects of different botnet structures.

There have been numerous techniques proposed to detect botnets. In [25] and [24], the authors employ machine learning techniques where they train classifiers with different features to detect botnet C&C flows. In [31], Strayer *et. al.* proposed a technique to detect botnet activity by exploiting temporal correlations between C&C activities of the bots belonging to the same botnet. Binkey and Singh proposed a technique to detect IRC botnets in [2] using botnet-related anomalies in TCP and IRC statistics. Another IRC botnet detection scheme was proposed by Goebel and Holz in [13], where the authors exploited the structure and evolution of IRC nicknames used by IRC bots. In [23], Karasaridis *et. al.* combined traffic aggregates with IDS alarms to identify centralized botnets within a Tier-1 ISP. In [16], Gu *et. al.* proposed BotHunter, which searches for a specific pattern of events in IDS logs to detect successful infections caused by centralized botnets.

All the above schemes were designed to detect either specific botnets that they were trained for, or centralized botnets. In general, detecting P2P bots in a network is harder since there is no trivial correlation that allows us to link together the P2P bots in a network, especially when bot peers communicate with each other using encryption and through random ports [14, 18, 10].

As a completely different problem from ours, crawler based methods were proposed to enumerate P2P bots globally in [22] and [18]. Since crawlers cannot enumerate P2P bots behind NAT and/or firewall in [21] Kang *et. al.* proposed a sybil attack based passive monitoring scheme to enumerate P2P bots even behind NAT or firewall. However, P2P bot enumeration methods are not intended to identify local P2P bots in a network. Also, they require implantation of bot peers which requires reverse engineering of a bot binary and its C&C protocol.

Coming back to our problem, there have been few techniques proposed which are able to detect local P2P bots assuming that P2P bots exhibit similar malicious activities and similar connection patterns. In [17], Gu *et. al.* proposed BotSniffer to detect bots based on spatial-temporal correlation between bot responses to commands. Following that, in [15], Gu *et. al.* proposed BotMiner which clusters the hosts in a network by their malicious activity and communication patterns. Their results showed that members of a botnet usually fall within the same cluster. Similarly, in [33], Yen and Reiter proposed a scheme called TAMD, where traffic containing similar external IPs, similar payloads and similar internal platform types are aggregated to detect botnets in a network. Although clustering based botnet detection schemes are successful in detecting many current P2P bots, botmasters can evade them by assigning different tasks to the bots in the same network or by randomizing their communication patterns as acknowledged in [15]. In [29], authors systematically investigate such evasion techniques. Also, clustering based schemes fall short in detecting idle P2P bots which haven't exhibited any overt behavior yet.

7. CONCLUSION AND FUTURE WORK

In this paper, we presented a simple and efficient method to identify local members of a P2P botnet in a network, starting from a known member of the same botnet in the same network. The basic idea of the proposed method is that, the members of a botnet are more likely to have mutual-contacts with each other than with benign hosts. We evaluate the proposed method using real P2P botnet (Nugache) data captured by a crawler. We also provide a mathematical analysis of the C&C structure of P2P botnets to characterize the performance of the proposed method. Both our analysis and experiments show that the proposed scheme is able to identify several dormant P2P bots in a network.

There are some limitations of the proposed scheme as discussed in Section 5. Perhaps the most important one is that, a botmaster can evade detection if she employs a structured P2P topology which ensures that her bots avoid mutual-contacts while communicating with each other. However, developing such a mechanism is not trivial for today's botnets and currently available P2P topologies. Nevertheless, even if a botmaster achieves such a topology, two or more networks can mitigate this by sharing their network traffic, possibly in a privacy-preserving manner, to exploit the mutual-contacts which will possibly occur between peers in different networks. We leave the exploration of the benefits of data-sharing as future work. In addition, we plan to study on a new P2P botnet architectures, which potentially evade the proposed scheme at least in some scenarios. This will allow us to further improve the proposed scheme to withstand potential evasion strategies, which might be employed by next generation botnets.

8. ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their comments and David Dittrich for his valuable contributions.

9. REFERENCES

[1] R. Bhagwan, S. Savage, and G. M. Voelker. Understanding availability. In *The 2nd International Workshop on Peer-to-peer systems*, 2003.

[2] J. R. Binkley and S. Singh. An algorithm for anomaly-based botnet detection. In *SRUTI'06: Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet*, 2006.

[3] J. Caballero, P. Poosankam, C. Kreibich, and D. Song. Dispatcher: Enabling active botnet infiltration using automatic protocol reverse-engineering. In *Proceedings of the 16th ACM Conference on Computer and Communication Security*, Chicago, IL, November 2009.

[4] CERT Coordination Center. SiLK: System for internet-level knowledge. Available at `http://tools.netsa.cert.org/silk/`.

[5] D. R. Choffnes, J. Duch, D. Malmgren, R. Guierma, F. E. Bustamante, and L. Amaral. Swarmscreen:

Privacy through plausible deniability in P2P systems. Technical report, Northwestern EECS Technical Report, March 2009.

[6] D. Dagon, G. Gu, C. Lee, and W. Lee. A taxonomy of botnet structures. In *Proceedings of the 23 Annual Computer Security Applications Conference (ACSAC'07)*, December 2007.

[7] L. K. Dawn and D. Song. Privacy-preserving set operations. In *in Advances in Cryptology - CRYPTO 2005, LNCS*, pages 241–257, 2005.

[8] D. Dittrich and S. Dietrich. Discovery techniques for P2P botnets. In *Stevens Institute of Technology CS Technical Report 2008-4*, September 2008.

[9] D. Dittrich and S. Dietrich. New directions in peer-to-peer malware. In *Sarnoff Symposium, 2008 IEEE*, April 2008.

[10] D. Dittrich and S. Dietrich. P2P as botnet command and control: A deeper insight. In *MALWARE 2008. 3rd International Conference on Malicious and Unwanted Software*, 2008.

[11] P. Erdos and A. Renyi. On random graphs I. *Publ. Math. Debrecen 6*, pages 290–297, 1959.

[12] P. Erdos and A. Renyi. The evolution of random graphs. *Magyar Tud. Akad. Mat. Kutato Int. Kozl 5*, pages 17–61, 1960.

[13] J. Goebel and T. Holz. Rishi: identify bot contaminated hosts by IRC nickname evaluation. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, 2007.

[14] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon. Peer-to-peer botnets: overview and case study. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, 2007.

[15] G. Gu, R. Perdisci, J. Zhang, and W. Lee. BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *Proceedings of the 17th USENIX Security Symposium (Security'08)*, 2008.

[16] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee. BotHunter: Detecting malware infection through ids-driven dialog correlation. In *Proceedings of the 16th USENIX Security Symposium (Security'07)*, August 2007.

[17] G. Gu, J. Zhang, and W. Lee. BotSniffer: Detecting botnet command and control channels in network traffic. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*, February 2008.

[18] 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 *LEET'08: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, 2008.

[19] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese. Network monitoring using traffic dispersion graphs (TDGs). In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 315–320, 2007.

[20] M. Jelasity and V. Bilicki. Towards automated detection of peer-to-peer botnets: On the limits of local approaches. In *Proceedings of the 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats LEET'09*, April 2009.

[21] B. B. Kang, E. Chan-Tin, C. P. Lee, J. Tyra, H. J. Kang, C. N. Z. Wadler, G. Sinclair, N. Hopper, D. Dagon, and Y. Kim. Towards complete node enumeration in a peer-to-peer botnet. In *Proceedings of ACM Symposium on Information, Computer and Communications Security (ASIACCS 2009)*, March 2009.

[22] C. Kanich, K. Levchenko, B. Enright, G. M. Voelker, and S. Savage. The Heisenbot uncertainty problem: challenges in separating bots from chaff. In *LEET'08: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, pages 1–9, 2008.

[23] A. Karasaridis, B. Rexroad, and D. Hoeflin. Wide-scale botnet detection and characterization. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, 2007.

[24] S. Kondo and N. Sato. Botnet traffic detection techniques by C&C session classification using svm. *Advances in Information and Computer Security*, pages 91–104, 2007.

[25] C. Livadas, R. Walsh, D. Lapsley, and W. Strayer. Using machine learning technliques to identify botnet traffic. *Local Computer Networks, Annual IEEE Conference on*, 0:967–974, 2006.

[26] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, and N. Borisov. BotGrep: Finding P2P bots with structured graph analysis. In *USENIX Security Conference*, August 2010.

[27] P. Porras, H. Saidi, and V. Yegneswaran. Conficker C P2P Protocol and Implementation, September 2009. `http://mtc.sri.com/Conficker/P2P/`.

[28] G. Sinclair, C. Nunnery, and B.-H. Kang. The waledac protocol: The how and why. In *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on*, pages 69 –77, October 2009.

[29] E. Stinson and J. C. Mitchell. Towards systematic evaluation of the evadability of bot/botnet detection methods. In *WOOT'08: Proceedings of the 2nd conference on USENIX Workshop on offensive technologies*, 2008.

[30] S. Stover, D. Dittrich, J. Hernandez, and S. Dietrich. Analysis of the storm and nugache trojans: P2P is here. In *;login: The USENIX Magazine*, volume 32-6, December 2007.

[31] W. Strayer, R. Walsh, C. Livadas, and D. Lapsley. Detecting botnets with tight command and control. *Local Computer Networks, Annual IEEE Conference on*, 0:195–202, 2006.

[32] The Honeynet Project. Honeywall, 2009. `https://projects.honeynet.org/honeywall/`.

[33] T.-F. Yen and M. K. Reiter. Traffic aggregation for malware detection. In *DIMVA '08: Proceedings of the 5th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 207–227, 2008.