On the Effectiveness of Structural Detection and Defense Against P2P-based Botnets

Duc T. Ha [†] (Contact Author) Guanhua Yan [‡] Stephan Eidenbenz [‡] Hung Q. Ngo [†]

[†]Dept of Computer Science and Engineering
University at Buffalo
Buffalo, New York 14200
{ducha,hungngo}@buffalo.edu
Phone: +1-716-541 8748, Fax: +1-212-409-8803

[‡] Information Sciences (CCS-3)
Los Alamos National Laboratory
Los Alamos, NM 87545
{ghyan, eidenben}@lanl.gov

Abstract

Recently, peer-to-peer (P2P) networks have emerged as a covert communication platform for malicious programs known as bots. As popular distributed systems, they allow bots to communicate easily while protecting the botmaster from being discovered. Existing work on P2P-based botnets mainly focuses on measurement of botnet sizes. In this work, through simulation, we study extensively the structure of P2P networks running Kademlia, one of a few widely used P2P protocols in practice. Our simulation testbed incorporates the actual code of a real Kademlia client software to achieve great realism, and distributed event-driven simulation techniques to achieve high scalability. Using this testbed, we analyze the scaling, reachability, clustering, and centrality properties of P2P-based botnets from a graph-theoretical perspective. We further demonstrate experimentally and theoretically that monitoring bot activities in a P2P network is difficult, suggesting that the P2P mechanism indeed helps botnets hide their communication effectively. Finally, we evaluate the effectiveness of some potential mitigation techniques, such as content poisoning, Sybil-based and Eclipse-based mitigation. Conclusions drawn from this work shed light on the structure of P2P botnets, how to monitor bot activities in P2P networks, and how to mitigate botnet operations effectively.
Keywords: *Botnets, Kademlia, structural analysis, monitoring, mitigation*

1 Introduction

Botnets, which are networks of compromised machines that are controlled by one or a group of attackers, have emerged as one of the most serious security threats on the Internet. With an army of bots at the scale of tens of thousands of hosts or even as large as 1.5 million PCs, the computational power of botnets can be leveraged to launch large-scale DDoS (Distributed Denial of Service) attacks, sending spamming emails, stealing identities and financial information, etc. For instance, it is reported that six botnets contribute 85% of all spamming emails seen on the Internet [26], and botnets have been used to launch DDoS attacks against DNS service [13].

As detection and mitigation techniques against botnets have been stepped up in recent years, attackers are also constantly improving their strategies to operate these botnets. The first generation of botnets typically employ IRC (Internet Relay Chat) channels as their command and control (C&C) centers . Though simple and easy to deploy, the centralized C&C mechanism of such botnets has made them prone to being detected and disabled. Against this backdrop, peer-to-peer (P2P) based botnets have emerged as a new generation of botnets which can conceal their C&C communication. Without a centralized C&C server, a P2P-based botnet does not suffer from a single point of failure, and its traffic, when buried in the enormous normal P2P traffic in the Internet, is extremely hard to detect.

Current work on P2P-based bonets mainly focuses on measuring existing P2P-based botnets, such as the highly publicized Storm botnet. In this work, we take one step further to investigate the structural characteristics of P2P-based botnets, explore the challenges of monitoring bot activities inside a P2P network, and evaluate the effectiveness of several attack techniques against P2P networks for botnet mitigation. Achieving all these goals calls for an experimental testbed with high flexibility and controllability. Towards this end, we build a P2P-based botnet simulation testbed, which uses the actual implementation code of a real P2P client soft-

ware for great realism, as well as distributed event-driven simulation techniques for high scalability. Using this testbed, we analyze the structures of P2P-based botnets and evaluate several monitoring and mitigation strategies.

The key contributions we make from this work can be summarized as follows: *First*, we analyze the scaling, reachability, clustering, and centrality properties of P2P-based botnets from a graph-theoretical perspective. *Second*, we demonstrate, from both experimental and theoretical aspects, that monitoring bot activities in a P2P network is difficult, suggesting that the P2P mechanism indeed helps botnets hide their C&C communication effectively. *Third*, we evaluate the effectiveness of some existing well-known attacks against P2P, but now used for botnet mitigation, such as content poisoning, Sybil-based mitigation, and Eclipse-based mitigation. Conclusions drawn from this work shed many insights on the structure of P2P botnets, how to monitor bot activities in P2P networks, and also how to mitigate botnet operations effectively.

The remainder of the paper is organized as follows. Section 2 presents related work. Section 3 provides some background on the Kademlia protocol and its variant Kad, which is used in our simulation-based study. Section 4 gives an overview of the design of our P2P-based botnet simulation testbed. Next, Section 5 analyzes P2P-based botnets from a graph-theoretic perspective, including their scaling, reachability, clustering, and centrality properties. Section 6 discusses the challenges of monitoring bot activities in a P2P network, both experimentally and theoretically. We further evaluate the effectiveness of three different mitigation techniques against botnet operations in Section 7. Finally, Section 8 concludes the paper.

2 Related Works

Existing studies on botnets mainly fall under the following three main categories: (i) Perform case-study of botnet behaviors and structures; (ii) Model hypothetic botnets to gain insights on their dynamics and defense schemes against them; (iii) Propose techniques for botnet detection and disruption. In the *first* category, Freiling et al. [14] infiltrated a real botnet to identify C&Cs and study bot commands. Rajab et al. [24] employed a multifaceted and distributed infrastructure to study botnet behaviors. Others works focused on measuring botnet size by various techniques such as by DNS redirection [10] or DNS cache snooping [23]. Our work on analyzing P2P-based botnets is inspired by some observations made from real botnets. In the *second* category, Dagon et al. analyzed intensively the impact of different botnet structures on some network metrics such as inverse geodesic length [9]. In [29], Vogt et al. also used simulation to shed light on the feasibility of using super-botnets for their command and control communication. Wang et al. proposed a hybrid peer-to-peer botnet structure that is more robust against server shutdown and hijack attacks than traditional botnets [30]. Dagon et al. observed diurnal patterns that impact the propagation speed of the botnet and thereby proposed a diurnal propagation model to capture this phenomenon [11]. Recently, a stochastic activity network model has been used to characterize peer-to-peer botnets in the M*ö*bius software tool [25]. Our work in this paper relies on a P2P-based botnet simulation testbed, in which a P2P-based botnet is modeled using discrete-event simulation techniques. Unlike previous work, our simulation model uses actual implementation code of a popular P2P cllient software to achieve great realism. In the *third* category, several techniques recently also been proposed to detect botnet existence: machine learning [20], anomaly detection [4], traffic or network activity statistics analysis [16]. In this work, we explore the difficulty of monitoring bot activities in a P2P network and also evaluate existing attack techniques against P2P network on mitigation against botnet operations. Conclusions drawn from this are complementary to existing botnet detection and mitigation techniques.

Kad is the first widely deployed peer-to-peer system based on a Distributed Hash Table. Stutzbach et al. were among the first to study the performance of lookup operation on Kad [28]. Steiner et al. [27] investigated several attacks on Kad, but they did not focus on the efficiency of the attacks, especially in the context of botnets. Some results in their paper were later contradicted by other work [17]. Finally, attacks against DHT (Distributed Hash Table) P2P networks have long been discussed since the dawn of DHT-based networks [7], [12]. However, they are mostly studied from a hypothetical perspective, and there is little work that explores their actual performance on real networks. Here we explore the potential of employing these attacks for good use against botnets, and from a practical perspective.

3 Primer on Kademlia and Kad

Kademlia, a peer-to-peer (P2P) protocol, was proposed by Maymounkov and Mazieres in [21]. Based on Distributed Hash Table (DHT), it provides a structured approach to P2P applications, where file storing and lookup operations can be efficiently performed with some resource-to-location mapping functions. In Kademlia networks, each node or resource (e.g., file) is associated with a 160-bit identifier in a circular ID space of size 2^{160}. These IDs are generated in a pseudo-random fashion (usually with a cryptographic hashing function), so that they can be deemed as uniformly distributed in the ID space. The distance d between two IDs X and Y is defined as the integral value of the bitwise-XOR result between X and Y, namely $X \oplus Y$. Each resource is stored on nodes whose IDs are closest to the resource's ID based on this distance metric.

Routing in Kademlia is done in an iterative fashion based on distances. A node, when searching for an ID (either a re-

source ID or a node ID), queries its neighbors for new nodes whose IDs are closest to the target ID. Upon receiving the answers, it continues to query those that are closer to the target. This process repeats until no closer node IDs can be found from the answer set.

For routing purposes, each node maintains a routing table containing information about its neighbors, such as their IDs, IP addresses, contact ports, etc. The routing table is organized as a tree of subtables called *bins*, each of which stores information about nodes with the same ID prefix. The contents of these bins are frequently updated whenever the host node receives a query, to ensure that only information of alive neighbors is stored. When a bin is full, new entries are added only if some old entries appear to be dead and can thus be removed. In this sense, Kademlia prefers existing nodes to newly joined ones. Further details regarding how routing tables are constructed can be found in [21].

Kad is a variation of the Kademlia protocol that has been adopted by the P2P community on several major P2P networks, including Overnet[1] and eMule [1]. Beside using 128-bit IDs, Kad supports more types of messages than Kademlia, and also handles the routing process in a slightly different fashion. More specifically, the search process in Kad consists of two phases:

- **Routing phase**: Similar to the original Kademlia routing protocol, the searching node asks its neighbors for nodes closest to the key ID in an iterative fashion. To accelerate the process, each peer in Kad simultaneously asks for the three closest peers so far in each round. The messages used in this phase are KADEMLIAREQUEST and KADEMLIARESPONSE, in the parlance of Kad.

- **Item querying phase**: After a certain amount of time since the query starts, the searching node selects some nodes that have responded and then queries for the key. The messages used in this phase are key-specific query messages, such as KADEMLIA_SEARCH_REQ and KADEMLIA_PUBLISH_REQ.

Each unit of information stored in a Kad network is associated with a unique key (i.e. an ID). There are three main types of keys. (1) *Source Key*: A source key identifies the content of a file and associates with information about the source node to download that file. Each instance of a file is associated with a unique source key (there can be multiple copies, thus multiple source keys). (2) *Keyword Key*: A keyword key identifies a textual keyword and associates with information of the source keys for files related to the keyword. (3) *Note Key*: A note key identifies a comment related to a specific file, and associates with information about a file (i.e its source key).

[1]Overnet was shut down due to copyright violations in late 2006.

In Kad, keys are not just published on a single peer that is closest to that key, but instead stored on at most 10 different peers close to the key. Keys are periodically republished, every 5 hours for a source key and 24 hours for a keyword or note key. Keys are removed from their resident peers if they have not been republished within their respective lifetime.

4 P2P-Based Botnet Simulation Testbed

The distributed architectures and existence of obfuscation techniques such as encryption in P2P-based botnets not only pose a serious challenge to studying their operational characteristics, but also hinder development of effective defense schemes to detect and even disrupt their operations. Current work on P2P-based botnets mainly focuses on monitoring, either passively or actively, behaviors of some existing P2P-based botnets, such as the Storm botnet [18, 19]. Although these bodies of work offer insights on how those botnets operate underground in reality, they have the following disadvantages. *First*, botnet monitoring usually takes place from a single or a few vantage points, thus cannot provide a full and consistent picture of the entire network. *Second*, researchers who attempt to actively measure an existing botnet may interfere with each other, potentially render information collected highly biased. For instance, the Storm botnet may have been overestimated due to interference from researchers performing their poisoning mitigation scheme on the botnet [19]. *Third*, even without considering the interference caused by other researchers, evaluating the performance of a proposed mitigation scheme accurately on a real botnet is difficult because the effect is usually observed from one or a few vantage points. *Fourth*, as an intelligent botmaster may dynamically change his strategy to evade detection, it is difficult to evaluate the effectiveness of a countermeasure on a real botnet. *Last but not least*, performing research on a real botnet may involve ethical and legal issues that are often neglected by cyber-security researchers [6].

With these challenges, it is thus necessary to have a testbed with such flexibility and controllability that we can use it to understand the operational dynamics of botnets and also evaluate effectiveness of different mitigation schemes. For this purpose, we develop a simulation-based virtual environment in which we can investigate P2P-based botnets extensively. In this simulation testbed, we simulate behaviors of P2P protocols with high fidelity. In contrast to some existing P2P simulators which often model P2P protocols at an abstract level, we use the implementation code of aMule [2], a real and popular P2P client software for eMule [1]. The aMule P2P client implements the Kad protocol, as described in Section 3. It is noted that the Storm botnet uses a modified version of the Overnet protocol for communication, which also implements the Kademlia algorithm. The code migrated from directly from aMule, however, cannot work straightforwardly in a simulation environment because timers in aMule

are associated with real wallclock time but in the simulation time is virtual and simulated. To address this problem, we intercept all time-related system calls and replace them with functions that use virtual simulation time.

As we port code from a real P2P client software, we simulate all details of the P2P protocol without any abstraction. Although this provides great realism in our simulation, it also brings the scalability challenge: simulating a botnet at the scale of a realistic one (e.g., a botnet with tens of thousands of bots like the Storm botnet) is so computationally prohibitive that it cannot be finished on a single commodity PC within a reasonable amount of time. To improve the scalability of our botnet simulator, we resort to distributed simulation techniques. Our botnet simulator is developed on top of PRIME SSF [22], a distributed simulation engine based on conservative synchronization techniques. Our distributed computing platform consists of 30 machines, each with 2 Pentium III CPUs and 4Gb RAM. Using this platform, our botnet simulator can simulate botnets with hundreds of thousands of bots within hours.

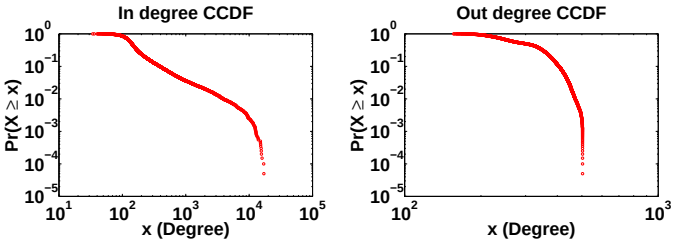
The simulation testbed offers great controllability with regard to how the botmaster of a bot dynamically changes operational strategies. For instance, a botnet using the Kademlia protocol may not strictly stick to the original protocol; instead, it can tweak some protocol behaviors for its own good. Modeling different botnet operational strategies can easily be done in our virtual botnet testbed. On the other hand, evaluating the performance of a specific mitigation scheme in the simulation testbed does not interfere with the normal operation of an existing botnet, and we do not need to consider those legal and ethic issues that result from working on a real botnet. In the following sections, we shall present results of using this simulation testbed to investigate characteristics of P2P-based botnets and defense schemes against them.

5 Structural Analysis

As the P2P protocol is the bedrock of a P2P-based botnet, operations of such a botnet and the corresponding defense schemes are inevitably implicated by its distinguishing P2P structure. In this section, we analyze the characteristics of P2P networks. If the whole P2P network is used exclusively for botnet operations, the properties of the P2P network presented in this section offer insights on the structure of such a P2P-based botnet; otherwise if a botnet uses only an existing P2P network to hide its communication (e.g., the Storm botnet which used Overnet for its command and control), conclusions drawn from the structural analysis of P2P networks will used later to decide which nodes we should monitor to detect botnet traffic. In the following discussion, we perform structural analysis on a P2P network with 20,000 nodes from a graph-theoretical perspective. Given this network, we form a *directed* graph $G(V, E)$ as follows: each node in the network is also a vertex in graph G and if a node b appears in node a's routing table, an edge from vertex a to b is added to the graph.

5.1 Scaling Property

Many real-world networks, such as the world wide web (WWW) and social networks, have been shown to be scale-free networks, whose degree distributions follow the power law. We are interested in whether a network built on the Kademlia protocol is also a scale-free network. In Figures 1(a) and 1(b), we depict the cumulative distribution of the in-degree and out-degree of graph $G(V, E)$, respectively. Visually, if the degree distribution follows a power law distribution, the curve should appear linearly. From the figures, it seems that both the indegree and outdegree of $G(V, E)$ do not follow the power law.



(a) In-Degrees (b) Out-Degrees

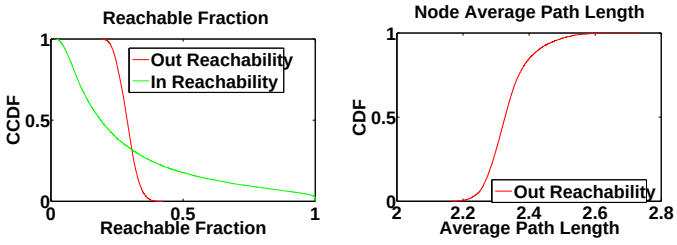
Figure 1. CDF of the Degrees (log-log scale)

To verify this observation rigorously, we apply the statistical method developed by Clauset et al. [8], which is based on maximum likelihood methods and the Kolmogorov-Smirnov statistic. Using this technique, we observe that the p-value, which is used to measure the goodness-of-fit, is 0 for both the in-degree and out-degree distributions. This further confirms that neither indegree nor outdegree of $G(V, E)$ follows the power law.

5.2 Reachability and Clustering Property

Figure 2(a) shows the complementary cumulative density function (CCDF) of the fraction of the reachable population from each node (out-reachability). In the network, 80% of all the nodes can reach 60% or more of the node population, but only 10% of the nodes can reach more than 68% of the node population. Interestingly, none of the nodes can reach more than 70% of all possible destinations. This observation, however, does not hold for in-reachability, which shows the fraction of nodes that can reach one specific destination. The CCDF of in-reachability is also depicted in Figure 2(a). We note that nearly 20% of the nodes can be reached from more than 90% of the population and some nodes can even be reached by all the other nodes. This implies that the P2P network can still work well if the resources are stored on these nodes with high in-reachability. Figure 2(b) shows the cumulative distribution of the average path length to reachable

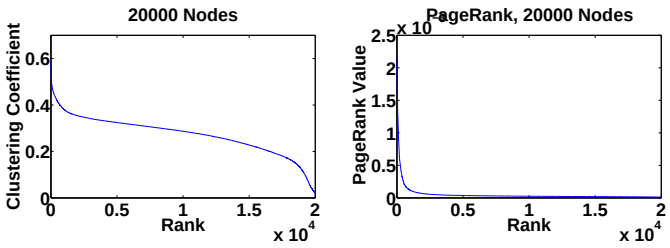
nodes. The average path length of the overall network is only 2.5 hops, suggesting that the average path length between two nodes is very small if there exists such a path.



(a) CCDF Reachable Fraction (b) CDF of Node Average Path Length

Figure 2. CDF and CCDF of the Path Length and Reachable Fraction

The network is also highly clustered as can be seen from Figure 4(a), which shows the individual clustering coefficient of each node. Almost 95% of all the nodes have clustering coefficient more than 0.3, and the average network clustering coefficient is 0.4136. This is much higher than the theoretical value 0.027 in an Erdos-Renyi random graph with the same number of nodes and edges.



(a) Clustering Coefficients (b) PageRank

Figure 4. Clustering Coefficients and PageRanks

5.3 Centrality Property

In this section, we studies various measures to identify important nodes in a P2P network. The goal is to find good metrics to identify strategic points. From a practical perspective, they should also be easy to use, yet able to give a fairly accurate picture of critical nodes. We consider common measures widely used in other domains.

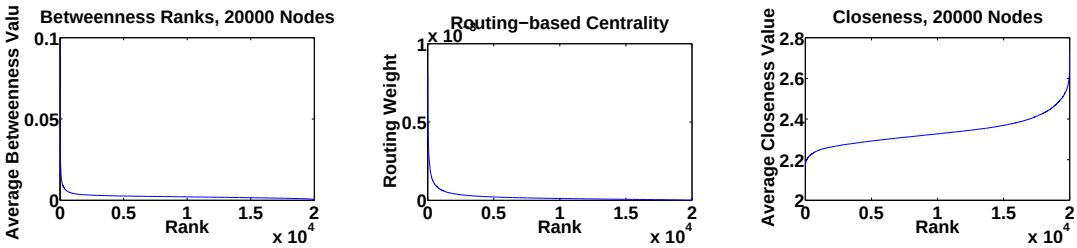
Degree centrality. We already showed the distribution of the in- and out-degrees in Figure 1. In many networks degree is an effective measure of the importance of a node. In the Internet, for example, nodes with more connections typically tend to receive more connections by new nodes.

Eigenvector Centrality. A more sophisticated measure of node importance is the eigenvector centrality. While degree centrality gives a simple count of the number of connections a vertex has, eigenvector centrality acknowledges that all connections are of equal importance. Nodes connected to more significant nodes, however, have more influence than those connected to less significant nodes. This effect can be represented by defining the centrality of a node to be proportional to the average centrality of its neighbors. Under this definition, the centrality measures of all the nodes form an eigenvector of the network adjacency matrix. This approach has shown to be an effective measure in many situations, for example in studies of the Internet topologies [15], finding clusters in information retrieval context [3], or ranking Web pages [5].

At the first glance, it seems that the same spectral analysis techniques used to study the Internet topologies [15] can be easily adapted to our network. This is not true because, unlike the Internet model in that paper, the P2P network is directed. Furthermore, the computation cost of the spectral analysis method poses a serious challenge for studying large scale networks (even the Internet topology studied in [15] was only a small fraction of the real Internet). These two factors render spectral analysis an unsuitable tool for our case.

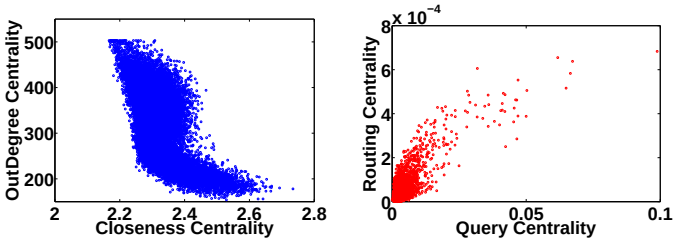
To circumvent this problem, we instead use the method employed for ranking Web pages [5] to compute centrality measure. The ranking method is the following: let $\mathcal{A}'$ is the directed adjacency matrix. For each node i define the out degree of i as $d_{out}(i) = |j : a_{ij} = 1|$. Now consider the stochastic matrix P where $p_{ij} = \frac{a_{ij}\alpha}{d_{out}(i)} + \frac{(1-\alpha)}{n}$. The eigenvector with eigenvalue 1 of P gives the ranks of the nodes. Note that since each of the nodes in our network has non zero outdegree, we have no dangling nodes, one important condition for the existence of this vector. Figure 4(b) shows the PageRank values of the nodes against their ranks. We note from the figure that the PageRank measure decays rapidly with the node rank, suggesting that a small fraction of nodes bear very high PageRank value compared to the remaining ones.

Betweenness Centrality. The betweenness centrality of vertex i is defined as the fraction of geodesic paths between other vertices on which it falls. That is, we find the shortest path (or paths) between every pair of vertices in the network and then derive the fraction of paths on which vertex i appears. In the P2P network, however, the geodesic paths do not reflect the actual routing paths, as explained in Section 3. To make betweenness a more reasonable centrality measure in our context, we define it based on the concept of *query set* from a source to a destination, which gives the set of nodes that will be queried by the source node in order to get to the destination. The betweenness centrality is thereby defined as the fraction of query set between other vertices that contain i. Figure 3(a) plots the betweenness measures against node ranks in the network. From the figure, we observe a similar



(a) Betweennes (b) Routing Weight (b) Closeness

Figure 3. Betweenness,Routing Weight and Closeness Centrality Measures



(a) OutDegree vs. Closeness (b) Betweenness vs. In-Degree

Figure 5. Centrality Correlations

pattern to PageRank centrality: only a small number of nodes appear frequently in the query sets of all source-destination pairs. For instance, only 20 nodes appear more than 0.5% of the paths between all source-destination pairs. This seems intuitive as P2P protocol is designed to be fully distributed so that each node in the network should have equal importance.

Closeness Centrality. The closeness centrality of vertex i is defined as the mean distance from vertex i to every other reachable vertex. We have already showed the cumulative distribution of the average distance per node in Figure 2(a). Figure 3(b) shows these distances against their ranks.

Routing-based Centrality. We define another metric called the *routing weight* based on the actual routes in the network. These actual routes are computed in accordance with the Kademlia routing protocol. For each route of length L (excluding the source and target node), we define a *routing weight* for each node in the route as: $1 + \frac{i}{L}$. Here i is the order of the current node, starting from 0 with the node next to the target, and increases towards the nodes near the source. The intuition behind this scheme is to put more weight on nodes that appear close to the source, yet still taking into account the number of routes they are on. The application of this metric will be explained further in the following sections.

Correlations between different centrality measures. Is there any relation between the above measures? Some of the above centrality measures can be computed easily, while others require significant efforts. From a practical perspective, if there are strong connections between two measures, one can estimate the measure that is hard to compute based on the more simple measure. We compute the correlation coefficients between these centrality measures with significance level 5% and the results are shown in Table 1. There are three interesting strong correlations: between betweenness and routing centrality, between closeness and out-degree centrality, and between betweenness and in-degree centrality. To confirm this, two of these correlations are further depicted in Figure 5.

6 Monitoring

To detect bot behaviors hidden in a P2P network, we need to monitor P2P traffic for some distinguishing features of botnets. The monitors can be placed at the enterprise gateway or ISP backbone routers. Given the scale of current P2P networks and the intensive P2P traffic, it is extremely difficult, if not impossible, to monitor all P2P traffic. We thus consider the following problem: suppose that we already know the topology of a P2P network through crawling, find the nodes to monitor so that botnet communication traffic can be checked as much as possible. To answer this question, we simulate a P2P network whose a portion are bots. The botmaster publishes bot commands as keywords in the network. Other bots, meanwhile, search for the bot command based on the keyword ID they already knew in advance (hardcoded) or can easily be computed (e.g, in the case of Peacomm bots).

We record all queries generated by these bots during the routing phase and rank the nodes in the network based on the traffic (number of queries) each node receives. Monitoring malicious traffic provides a venue to detect botnet command and control since each routing query packet carries a unique keyword ID. If we know a keyword ID has been used for bot communication in advance, a surge of query packets associated with this ID is a good indicator of intense bot activities. Due to the dynamics of routing queries in a P2P network, one may wonder whether it is possible to use centrality measures discussed in Section 5 to predict the importance of the nodes. This knowledge can then be used for monitoring traffic efficiently. To address this question, we first define the *overlapping ratio* metric as follows. Given two rankings of all the nodes in the P2P network, their overlapping ratio for

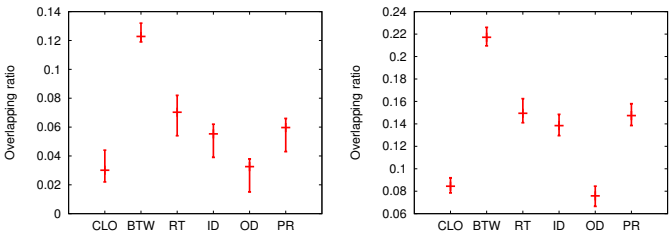
Table 1. Correlation Coefficients Between Centrality Measures

	Closeness	Betweenness	Routing	In-Degree	Out-Degree	PageRank
Closeness	1.0000	-0.1112	-0.3684	-0.2572	-0.6392	0.4261
Betweenness	-0.1112	1.0000	0.7744	0.6933	0.0722	-0.3127
Routing	-0.3684	0.7744	1.0000	0.8540	0.4383	-0.4751
In-Degree	-0.2572	0.6933	0.8540	1.0000	0.3606	-0.3217
Out-Degree	-0.6392	0.0722	0.4383	0.3606	1.0000	-0.5144
PageRank	0.4261	-0.3127	-0.4751	-0.3217	-0.5144	1.0000

the top n nodes is given by the fraction of common nodes that appear in the top n nodes of both rankings. For instance, if 20 nodes appear among the top 100 ones of both rankings, the overlapping for the top 100 is 20%.

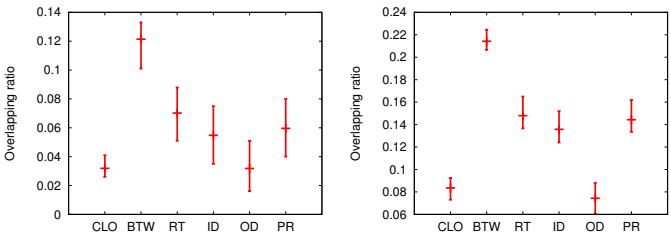
We next discuss possible strategies from the botmaster's point of view. As described in the previous section, a peer goes through two phases to publish (or query for) a key: routing phase for finding nodes closer to the key, and publish (or query) phase when some responded nodes are requested to publish (or query for) the key. In practice, the set of nodes involved in the second phase is much smaller than the set of responded nodes in the first phase. We also observe one key feature of the protocol: keywords are published in a *passive* fashion, namely only the original node publishes keywords, whereas other nodes only passively accept the keyword without republishing it elsewhere. Consequently, bot commands will only be published by bots, not by regular benign Kad clients. Thus the prevalence of a bot command in the network depends only on how the bots publish it. As the chance of finding a keyword successfully is proportional to its availability in the network, the bot master can design the bots in such a way that the prevalence of the keyword in the network is significantly improved, although such behavior does not conform to the standard Kad protocol. In our study, we consider three different strategies by the botmaster in terms of the aggressive levels with which bots publish/query the command keyword: in the publish/query phase, a bot selects only a small set of responded nodes as in the standard protocol (*strategy 1*), 50% of all responded nodes (*strategy 2*), or all responded nodes (*strategy 3*).

In the experiments, we consider two different stable routing table snapshots. From each snapshot, we select 10 sets of random bot nodes and simulate the communication within 1.8 hour. In each simulation, we select a set of top m nodes that receive the most querying bot traffic and compare it with top m nodes defined by each centrality measure. The performance of each centrality measure in terms of the overlapping ratio for the top 1,000 and 2,000 nodes is depicted in Figures 6, 7, and 8, under the three strategies, respectively. In this figures, CLO, BTW, RT, ID, OD, and PR stand for the closeness, Betweenness, Routing-based, Indegree, Outdegree, and PageRank centrality measures, respectively. At each data point, we give the mean overlapping ratio and its observed minimum and maximum over the 10 sample runs.



(a) 1000 nodes (b) 2000 nodes

Figure 6. Overlapping ratios under strategy 1



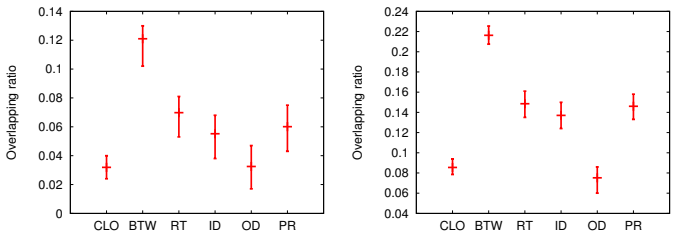
(a) 1000 nodes (b) 2000 nodes

Figure 7. Overlapping ratios under strategy 2

From these plots, it can be seen that none of the standard centrality measures provide good prediction on those important nodes in terms of their routing query traffic, regardless of the strategy applied by the botmaster. In fact, their performance is close to the case when the nodes are randomly selected, as can be seen from the following lemma.

Lemma 6.1. *Let X be a fixed set of size m in a space $N, |N| = n \geq m$. Let Y be another set of the same size selected randomly from N. The expected number of common elements between X, Y can computed as:*

$$E[|X \cap Y|] = \frac{m^2}{n} \quad (1)$$



(a) 1000 nodes (b) 2000 nodes

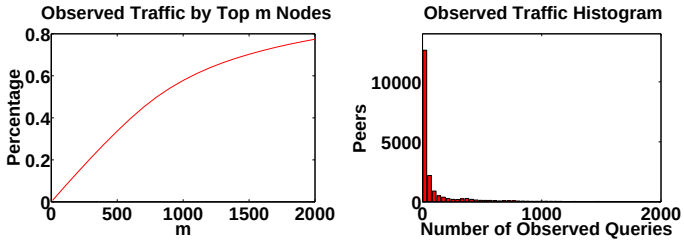
Figure 8. Overlapping ratios under strategy 3

Proof. Denote $c = |X \cap Y|$. We have:

$$E[c] = \sum_{k=0}^{m} k\mathrm{P}[c=k] = \sum_{k=1}^{m} \frac{k\binom{m}{k}\binom{n-m}{m-c}}{\binom{n}{m}}$$
$$= \frac{m\binom{n-1}{m-1}}{\binom{n}{m}} = \frac{m^2}{n}$$

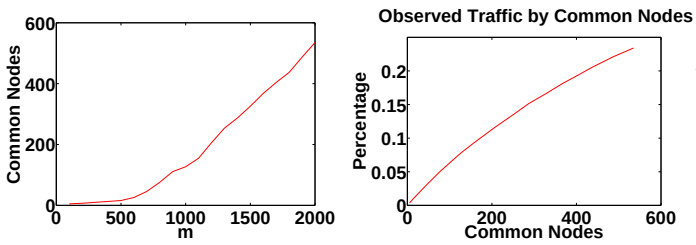
□

Plug in the value of $n = 20000$, $m = 1000$ and $m = 2000$, we have $E[|X \cap Y|] = 50$ and $E[|X \cap Y|] = 200$, respectively. From these numbers, it can be seen that all centrality measures except Betweenness do not yield better results than a random approach.



(a) By Top m Nodes (b) Histogram

Figure 9. Observed Traffic



(a) Common Set Size (b) By Common Nodes

Figure 10. Observed Traffic

However, is it possible that the traffic is well distributed over the whole network, rendering any ranking methods ineffective? To answer this question, we look at the traffic observed by top m nodes with heaviest traffic when m varies. Figure 9(a) shows this relation (averaged over 10 different bot sets). It can be seen that for each key, its query traffic is disproportionally distributed. For example, nearly 80% of all traffic is observed (received) by 10% of the population (2,000 nodes out of 20,000 in our case). This is further confirmed in Figure 9(b), which shows the histogram of the number of observed queries by each node. It is noted that for the majority of the nodes, they only observe a few queries.

In Figure 10(a), we depict the number of common nodes shared among the top m ones across the scenarios with different bot sets; the fraction of traffic observed by these common nodes against their numbers is shown in Figure 10(b). These results show one important characteristic of Kad networks: the routing query traffic is very dynamic and is significantly affected by where those querying peers are distributed in the network. As a result, centrality measures discussed in Section 5 all perform poorly in predicting those important nodes. Table 2 shows the number of common elements across all 10 scenarios with different bot sets that can be predicted by each centrality measure. It can be seen that the betweenness centrality outperforms all others but is still only able to capture a small number of common nodes. However, even if there is a measure that can capture most of the common nodes, the amount of observed traffic by these nodes is still small as shown in Figure 10(b).

Bot communication interdiction. The results from the above section show that the set of traffic-critical nodes depends on the locations of the bots. Note that given the routing tables of all nodes in the network, one can roughly estimate the path between any source node and destination key (or nodes). In some cases, we can collect a list of suspicious nodes that are or may be bots. An interesting question is: if we know the paths among these nodes, which nodes should we monitor so that communication traffic generated between these suspicious nodes can always be interdicted? We formulate it as the following problem:

Problem 1. (Bot Communication Interdiction – BCI) Given a set of communication paths $P = \{p_1, p_2, \ldots, p_n\}$ in a directed graph $G = (V, E)$, where each path p_i is an ordered list $< v_{i1}, \cdots >$ of nodes in V. A subset $I \subseteq V$ interdicts a path p_i if $\exists v_j \in I$ and $v_j \in p_i$. Find the minimum interdiction set I for P such that $\forall p_i \in P$, I interdicts p_i.

Here, each path represents a communication path estimated between two suspicious nodes. One would hope to find the smallest set of nodes that lay on all of these paths so that the monitoring cost can be minimized. This problem, unfortunately, is intractable, as shown by the following theorem.

Theorem 6.2. *The general version of* BCI *is* **NP***-hard.*

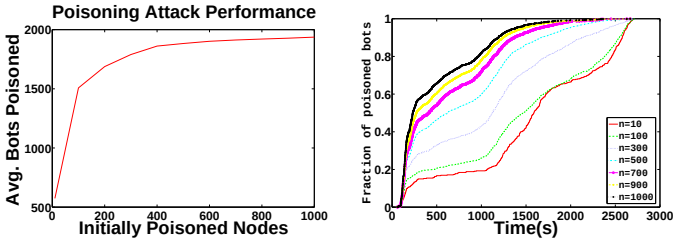
Proof. We reduce HITTING SET to BCI. Consider an instance of the decision version of HITTING SET where we are given a collection $\mathcal{C}$ of subsets of a finite set $\mathcal{S}$, and a positive

Table 2. Number of common nodes among the top m under different rankings

m	Total number of common nodes	Closeness	Betweenness	Routing	In-Degree	Out-Degree	PageRank
1000	127	2	46	29	23	12	26
2000	535	40	219	152	142	75	152

integer k. A hitting set for $\mathcal{C}$ is defined as a subset $\mathcal{S}' \subseteq \mathcal{S}$ such that $\mathcal{S}'$ contains at least one element from each subset in $\mathcal{C}$. It is **NP**-hard to determine if there is a hitting set of size at most k.

An instance of BCI is constructed as follows. For each element in $\mathcal{S}$, create a corresponding vertex in V. From each subset $s_i \in \mathcal{C}$, create a path p_i consisting of elements in s_i. The order of the vertices in p_i is not important for our proof. It is straightforward to see that HITTING SET has a hitting set of size at most k if and only if BCI has an interdiction set of size at most k. □



(a) Avg. num. of bots poisoned (b) Frac. of bots poisoned

Figure 11. Performance of Poisoning Schemes

7 Mitigation

In the previous section, we have shown that monitoring bot activities in P2P networks is a challenging task. In the section, we shall use simulation to further explore effectiveness of defense schemes against botnet operations that have already been proposed in the literature. The three schemes discussed here include content poisoning, Sybil-based mitigation, and Eclipse-based mitigation.

7.1 Poisoning-Based Mitigation

As bot command keys are used by bots to search new commands in a P2P-based botnet, they are actually one Achilles' heel of such botnets: once these keys are known, we can inject benign contents with the same key into the network so that some bots may not receive the original bot commands. This technique is thus called *poisoning attack* and was first experimented in [17].

In our simulation, we publish the bot command key with benign contents on a set of nodes. Bots that make search requests to these special nodes are considered to be "poisoned" with the content we provided. This is because the poisoned nodes reply to any request for the bot command with the content they have (which we provided). One interesting question that follows is how effective this method is, with respect to the size of the initially poisoned set. To answer this question, we vary the size of the initially poisoned set among 10, 100, 300, 500, 700, 900, and 1000. The poisoned nodes are chosen from those with the top betweenness values, since as shown in the previous section, the betweenness centrality measure seems to provide more nodes with high query traffic than other measures.

Figure 11(a) shows the average number of affected bots that received poisoned bot commands under all scenarios. It can be seen that although the nodes chosen based on the betweenness centrality measure are not efficient from the traffic monitoring perspective, they are still effective in poisoning the bots. For instance, using only 10 initially poisoned nodes we can poison on average 25% of the total number of bots (more than 500 bots), regardless of their locations. With 100 nodes, we can poison more than half of the bots. However, it is more costly to poison a large portion of the bot population. After 400 nodes, the size of the initially poisoned set just makes little difference in the final number of poisoned bots.

To understand the impact of the size of the initially poisoned set on the speed of the poisoning process, we show the fraction of poisoned bots (over all poisoned bots eventually in each scenario) against the simulation time in seconds under different sizes of the initially poisoned set, denoted by n. It is noted that higher number of initially poisoned nodes helps speed up the poisoning process, but again the difference becomes smaller when the size is sufficiently large.

7.2 Sybil-Based Mitigation

Table 3. Sybil-based Mitigation Scenarios

	Mitigated Nodes	IDs per Mitigated Nodes	Total No. of IDs
Scen. (1)	10	1000	10000
Scen. (2)	100	100	10000
Scen. (3)	1000	10	10000
Scen. (4)	100	1000	100000
Scen. (5)	1000	100	100000
Scen. (6)	1000	300	300000

Passively monitoring traffic of a large number of nodes in a P2P

Table 4. Traffic Observed by Sybil Agent

Scen. (1) (10x1000)	Scen. (2) (100x100)	Scen. (3) (1000x10)	Scen. (4) (100x1000)	Scen. (5) (1000x100)	Scen. (6) (1000x300)
0	2.6017e-06	0.0023	1.6122e-05	0.0142	0.021112

Table 5. Bot Coverage by Sybil Agent

Scen. (1) (10x1000)	Scen. (2) (100x100)	Scen. (3) (1000x10)	Scen. (4) (100x1000)	Scen. (5) (1000x100)	Scen. (6) (1000x300)
0	6.7	1935.3	26.6	1998.8	1999

network is challenging and resource-intensive. A more aggressive approach is to introduce a set of node IDs that are actually controlled by a small number of physical processing nodes. In this way, traffic routing through these IDs are available for full analysis. Moreover, these nodes do not have to conform to the standard protocol, and may even reduce the processing overhead. For example, the processing nodes may ignore IDs that they are not interesting, or just selectively reply with fake IDs to trick the querying node to publish valuable resources on them. This is a form of *Sybil attack*, as describe in [12].

In our simulation, we inject a varied number of IDs into the routing tables of a selected set of nodes in the network. Again, these nodes are chosen based on the betweenness centrality measure. All injected Sybil IDs are associated with one extra node designated as the central processing node called *Sybil Agent*. This agent is responsible for all requests directed to a Sybil ID. However, we only simulate a passive agent that collects requests without further replying any IDs. Using this setup, we study the performance of this approach with respect to some key parameters, including the number of nodes to which the Sybil IDs will be introduced into and the number of Sybil IDs per node.

We consider three different sets of Sybil IDs with size 10000, 100000 and 300000. For the first two sets, we consider several subscenarios when the set of nodes to be mitigated varies. Table 3 details the configuration of each of these scenarios. Table 4 shows the fraction of queries that were sent to the Sybil agent through these IDs. The results reveal that the traffic received by the Sybil Agent is very small even when the number of injected nodes is large.

A better metric to assess the performance of Sybil-based mitigation is the number of bots that queried at least one Sybil ID. Table 5 shows the average number of such bots, out of 2000 bots. Here we can see the Sybil Agent has very good coverage of the bot population when we send Sybil IDs to a large number of nodes. Hence, given the same number of Sybil IDs, it is more effective to spread out the IDs on more nodes than sending more IDs to a small number of nodes. Note that this is not obvious because sending more IDs also increases the chance to catch more keys.

7.3 Eclipse-Based Mitigation

Eclipse attacks against P2P networks were first described by Castro et al [7]. In this type of attacks, a set of nodes collaborate to separate a node from the rest of the network by cutting off all information towards and out of the node. This can only be done when all incoming and outgoing traffic is directed through the malicious nodes. For outgoing traffic, the victim's routing table needs to contain only information about malicious nodes. For incoming traffic, the malicious nodes must be able to interfere with all queries towards the victim. If any of these conditions fails, the victim will still be able to contact benign nodes, get updated with fresh information and break out of the quarantine.

We only look at incoming traffic towards the victim, since poisoning the victim's whole routing table requires significant time, even not possible if existing nodes in the table are still alive (due to Kad's preference for existing nodes). In our experiments, we use our Sybil IDs as the malicious nodes. Any bot that queries at least one of these IDs within 120 seconds are considered to be fooled by our nodes. Table 6 shows the number of bots among the 2,000 bots in the network that can be fooled. It shows that eclipse-based mitigation is not efficient against botnets, even although Sybil-based mitigation can achieve very high coverage. This is because we require the malicious nodes to be contacted at a much earlier state of the querying process. Our simulation result also agrees with the outcome in [17], where the authors tried to eclipse bot commands with very large number of Sybil IDs.

8 Conclusions

In this work, we build a P2P-based botnet simulation testbed, which uses actual implementation code of the Kad P2P protocol to achieve great realism. This simulation testbed employs distributed event-driven simulation techniques for high scalability. With this testbed, we analyze the structural characteristics of P2P-based botnets, explore the challenges of monitoring bot activities inside a P2P network, and evaluate the effectiveness of mitigation techniques that are already proposed in the literature. Conclusions drawn from this work shed many insights on the structure of P2P botnets, how to monitor bot activities in P2P networks, and also how to mitigate botnet operations effectively.

References

[1] http://www.emule-project.net.

[2] http://www.amule.org.

[3] Y. AZAR, A. FIAT, A. KARLIN, F. MCSHERRY, AND J. SAIA, *Spectral analysis of data*, in STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing, New York, NY, USA, 2001, ACM, pp. 619–626.

Table 6. Bot Coverage under Eclipse-Based Mitigation

Scen. (1) (10x1000)	Scen. (2) (100x100)	Scen. (3) (1000x10)	Scen. (4) (100x1000)	Scen. (5) (1000x100)	Scen. (6) (1000x300)
0	0.1	23.6	1.2	36.4	39.9

[4] 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, Berkeley, CA, USA, 2006, USENIX Association.

[5] S. BRIN AND L. PAGE, *The anatomy of a large-scale hypertextual web search engine*, vol. 30, Amsterdam, The Netherlands, The Netherlands, 1998, Elsevier Science Publishers B. V., pp. 107–117.

[6] A. J. BURSTEIN, *Conducting cybersecurity research legally and ethically*, in LEET'08: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats, 2008.

[7] M. CASTRO, P. DRUSCHEL, A. GANESH, A. ROWSTRON, AND D. S. WALLACH, *Secure routing for structured peer-to-peer overlay networks*, in Proceedings of OSDI'02, 2002.

[8] A. CLAUSET, C. R. SHALIZI, AND M. E. J. NEWMAN, *Power-law distributions in empirical data*, Jun 2007.

[9] D. DAGON, G. GU, C. LEE, AND W. LEE, *A taxonomy of botnet structures*, in Proceedings of ACSAC'07, 2007.

[10] D. DAGON, C. ZOU, AND W. LEE, *Modeling botnet propagation using time zones*, in NDSS, The Internet Society, 2006.

[11] D. DAGON, C. C. ZOU, AND W. LEE, *Modeling botnet propagation using time zones*, in Proceedings of NDSS'06, 2006.

[12] J. R. DOUCEUR, *The sybil attack*, in IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems, London, UK, 2002, Springer-Verlag, pp. 251–260.

[13] `http://securitywatch.eweek.com/exploits_and_attacks/everydns_opendns_under_botnet_ddos_attack.html`.

[14] FREILING, HOLZ, AND WICHERSKI, *Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks*, in ESORICS: European Symposium on Research in Computer Security, LNCS, Springer-Verlag, 2005.

[15] C. GKANTSIDIS, M. MIHAIL, AND E. ZEGURA, *Spectral analysis of internet topologies*, vol. 1, March-3 April 2003, pp. 364–374 vol.1.

[16] G. GU, V. YEGNESWARAN, AND MARTIN, *Bothunter: Detecting malware infection through ids-driven dialog correlation*, pp. 167–182.

[17] 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, Berkeley, CA, USA, 2008, USENIX Association, pp. 1–9.

[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] 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, 2008.

[20] C. LIVADAS, R. WALSH, D. LAPSLEY, AND W. T. STRAYER, *Using machine learning techniques to identify botnet traffic*, Proceedings 2006 31st IEEE Conference on Local Computer Networks, (2006), pp. 967–974.

[21] MAYMOUNKOV AND MAZIERES, *Kademlia: A peer-to-peer information system based on the XOR metric*, in International Workshop on Peer-to-Peer Systems (IPTPS), LNCS, vol. 1, 2002.

[22] `http://lynx.cis.fiu.edu:8000/twiki/bin/view/Public/PRIMEProject`.

[23] M. RAJAB, J. ZARFOXX, F. MONROSE, AND A. TERZIA, *My botnet is bigger than yours (maybe, better than yours)*, Proceedings 2007 USENIX First workshop on Hot Topics in Understanding Botnets, (2007).

[24] M. A. RAJAB, J. ZARFOSS, F. MONROSE, AND A. TERZIS, *A multifaceted approach to understanding the botnet phenomenon*, in Internet Measurement Conference, J. M. Almeida, V. A. F. Almeida, and P. Barford, eds., ACM, 2006, pp. 41–52.

[25] E. V. RUITENBEEK AND W. H. SANDERS, *Modeling peer-to-peer botnets*, in Proceedings of IEEE QEST'08, 2008.

[26] `http://www.theregister.co.uk/2008/02/29/botnet_spam_deluge`.

[27] M. STEINER, T. EN-NAJJARY, AND E. W. BIERSACK, *Exploiting kad: Possible uses and misues*, ACM SIGCOMM Computer Communication Review, 37 (2007).

[28] D. STUTZBACH AND R. REJAIE, *Improving lookup performance over a widely-deployed dht*, INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, (2006), pp. 1–12.

[29] R. VOGT, J. AYCOCK, AND J. M. J. JACOBSON, *Army of botnets*, in Proceedings of NDSS'07, 2007.

[30] P. WANG, S. SPARKS, AND C. C. ZOU, *An advanced hybrid peer-to-peer botnet*, in Proceedings of HotBots'07, 2007.