Optimising Sybil Attacks against P2P-based Botnets

Carlton R. Davis, José M. Fernandez
École Polytechnique de Montréal,
Montréal, QC, Canada.
{carlton.davis, jose.fernandez}@polymtl.ca.

Stephen Neville
University of Victoria,
Victoria, BC, Canada.
sneville@ece.uvic.ca.

Abstract

Addressing and mitigating modern global-scale botnets is a pressing Internet security issue, particularly, given that these botnets are known to be provide attackers with the large-scale low-cost computing infrastructure required to engage in major spam campaigns, larger-scale phishing attacks, etc. *Over time, botnets have evolved toward using decentralized peer-to-peer (P2P) command and control (C&C) infrastructures in order to increase their resilience against defender countermeasures,* i.e. *as seen in Storm's use of Overnet and more recently in the appearance of HTTP-tunneled P2P botnets, such as Waledac and Conficker. The obvious question is, what are effective countermeasures against these modern botnets? This work focuses on evaluating, via simulation, sybil attack-based countermeasures and how such sybil-based strategies should be tailored to allow them to both be effective and implementable on global-scales. Slower-rate sybil infection strategies with random placement of sybils are shown to be nearly as effective as higher-rate infection strategies with targeted placement. This somewhat counter-intuitive result is important, as the former strategy is easier to implement by a loosely co-ordinated collective of globally scattered defenders.*

1 Introduction

Botnets have evolved into effective base infrastructures for cybercrime. They provide the low-cost large-scale army of compromised machines required to engage in large-scale spam and phishing campaigns, to harvest at-scale users' private information, *e.g.* credit card numbers, bank account access information, *etc*., and complex networks in which cyber-criminals can hide their tracks. Between 85 to 95 percent of spams originated from botnets [19], with spamming being highly profitable to cybercriminals, with estimates existing that $3.5M per year was genetered from Storm-produced pharmaceutical focused spam campaigns [14]. "Pump-and-dump" stock spam campaign, designed to manipulate stock values also appears to be quite lucrative [12]. The underground cyber-crime economy supported by botnets has reached the extent that botnets operators (or masters) have become middlemen who then lease their botnets out to other cyber-criminals.

The security community, historically, has had significant success in developing countermeasures for traditional IRC based botnets, primarily through exploiting their highly centralized nature [1, 2, 11, 21, 25]. This success, though, drove the botnet operators to move to decentralized command and control (C&C) infrastructures offered by the emergence of peer-to-peer (P2P) networks. More particularly, the botnet operators move to the use of the P2P's distributed hash table (DHT) capabilities provided, for example, via Overnet's [15] integration of the Kademlia protocol [17], which provided an increased level of decentralization even over the more commonly available file sharing-based P2P protocols. The inherent resilience of these P2P-based botnets to defensive countermeasures combined with their profitability means that they are likely to persist for the foreseeable future, with minor modifications, such as employing tunneling over HTTP, to circumvent basic firewall-based countermeasures.

From a defensive standpoint it therefore makes sense to explore effective mitigation and disruption strategies for such P2P-based botnets. More particularly, given that a given P2P botnet can include operational nodes located around the world it makes sense to explore the trade-off between loosely coupled mitigation strategies, *i.e.* uninformed strategies where sybils are placed at random in the P2P network and more tightly coupled ones, *i.e.* informed strategies where sybil are strategically placed based on information globally collected and shared. The former are more easily implemented by globally scattered defenders acting independently, while the latter require structured collaborations and potentially hard to obtain global knowledge of the botnet's structure.

Prior literature [4, 7, 13] has demonstrated the general effectiveness that Sybil attacks have on P2P-based botnets, where a Sybil attack denotes the injection of large number

of active nodes into the botnet that provide erroneous responses to other botnet nodes' queries. However what has remained largely unexplored in this work is the issue of how to best orchestrate these Sybil attacks or, more precisely, the trade-offs inherent in core tunable Sybil attack parameters, *i.e.* the Sybil injection rate and their placement have on the effectiveness of the Sybil attack, *i.e.* the degree of disruption to the botnet's operation. In this work we explore this question by through stochastic simulations based of such attacks using a precise graph representation of the P2P network generated by such botnets. The paper focused on Kademlia-based P2P botnets as: (a) the details of this protocol are well known and understood, (b) it has formed the core of past P2P botnets, *i.e.* Storm, (c) for practical purposes future P2P networks are likely to follow similar operational characteristics, *e.g.* a DHT structure, a command set with functionality,*etc*., and (d) previous work [5] seems to indicate that it is more resilient to targeted attacks than other P2P network protocol choices. Thereby, we expect that results obtained with respect to Kademlia in this paper are likely to extend to such future P2P botnets.

The main contributions of this paper are the following:

1. A more refined and accurate graph model is introduced for describing the behaviour and performance Kademlia-based P2P botnet C&C infrastructure.

2. The time duration of the Sybil attack is shown to have more bearing on the efficacy of Sybil attacks than the number of sybils that are employed in the attack, *i.e.* slower rate Sybil attacks are more effective than faster rate attacks.

3. A sybil-to-bot ratio of approximately 1:10 is shown to be largely sufficient for maximizing the effectiveness of the Sybil attacks, *i.e.* a higher ratio does not improve the effectiveness of the attack though it does increase the ability of the botmaster(s) to detect the Sybil attack.

4. Finally, randomly sybil placement is shown to be nearly as effective as placing sybils close to the botnet search keys, *i.e.* little appears to be lost through the use of an uninformed placement strategy over an informed one. This is an important result as it implies that globally distributed, uncoordinated defenders each engaging in their own sybil attacks will be nearly as effective as a co-ordinated effort to disrupt the botnet would be.

The rest of the paper is organised as follows. Section 2 contains background information about the Kademlia protocol and the nature of Kademlia-based botnet C&C infrastructures. Section 3 highlights related work and outlines differences between this and previous work. We present the graph model we utilised for our simulations in Section 4, followed by the simulation setup in Section 5. Finally, in Sections 6 and 7 we discuss the implications of our results, summarise our findings and present directions for future work.

2 Background

Storm provides a convenient exemplar from which to explore Kademlia-based P2P botnets in that its inner workings are now well understood. The information presented below, although flavoured in terms of Storm, also pertains to P2P botnets which follow similar core operating principals as exhibited by the Kademlia protocol.

Storm utilised a modified version of the Overnet P2P overlay network, *i.e.* an application layer network, for its C&C infrastructure. Overnet is a proprietary implementation of the Kademlia algorithm, which is a P2P $\langle \text{key}, \text{value} \rangle$ storage and look-up system. In the Overnet implementation, *keys* are 128-bit opaque quantities, and *values* are typically strings containing the IP address and other necessary information of servers where given electronic data can be found. Each node in an Overnet overlay network has a random 128-bit ID that the node generates when it initially joins the network. This 128-bit ID serves as part of the node's unique identifier, which consists of the triplet $\langle \text{IP address}, \text{port}, \text{ID} \rangle$; where port is the port number that the Overnet protocol listens for network traffic. Kademlia stipulates that the $\langle \text{key}, \text{value} \rangle$ pairs should be stored on nodes whose IDs are "close" to the associated *key*. Closeness is defined in terms of the XOR metric: the distance of a node with ID x from a node with ID y (or with a key y) is defined as the bitwise exclusive (XOR) of x and y interpreted as an integer, *i.e.*, $d(x, y) = (\text{int})\, x \oplus y$.

Each node in a Kademlia P2P network keeps a list of $\langle \text{IP address}, \text{port}, \text{ID} \rangle$. These lists are referred to as *peer lists*, as they contain information on how to contact other peers in the network. In Kademlia, the peer list is organised into sub-lists called k-*buckets*, containing triplets of nodes whose distance is between 2^i and 2^{i+1} from itself; where $0 \leq i < 128$. They are so-called because they are managed to contain at most k nodes, where k is a configurable parameter, typically 20 for most Kademlia implementations. Records in a k-bucket are kept sorted by time last seen, with the least-recently seen at the head and the most recently-seen at the tail. When a node (the recipient) receives a message from another node (the sender), the recipient updates the k-bucket corresponding to the sender (i.e. the one corresponding to the distance between itself and the sender) as follows: If the sender is already in the recipient's k-bucket, the sender's triplet gets moved to the tail of the k-bucket. If the sender's triplet is not in the corresponding k-bucket, then if there are less than k triplets in the k-bucket, the sender's triplet is added to the tail of the k-bucket. Otherwise, the recipient pings the node n that is at the head of

the k-bucket (the least-recently seen node); if n does not respond, it is evicted from the k-bucket and the recipient adds the sender to the tail of the k-bucket. If n does respond, the recipient moves n's triplet to the tail of the k-bucket and the sender's contact is discarded. (This, as we will see, provides a measure of protection against sybil attacks, since only inactive nodes are forced out of k-buckets.)

Kademlia utilises the following the four message types which constitute the botnet commmand set for placing and retrieving C&C information:

`PING` : This message is use to probe a node to determine if it is on-line.

`FIND_NODE` : This message type allows a node to search for a node ID. When a node receives a `FIND_NODE` message, it returns the $\langle$IP address, port, ID$\rangle$ triplet of k nodes it knows about that are closest to the ID.

`STORE` : If a node wishes to publish a $\langle$key, value$\rangle$ pair, the node locates the k closest nodes to the key (via the `FIND_NODE` message) and sends each a `STORE` message, which consequently instructs the nodes to store the $\langle$key, value$\rangle$ pair for later retrieval. This command is used by the botmasters to place the information it wants the bots to find, where the value allows the bots to find out where to go to retrieve updates or commands.

`FIND_VALUE` : A node can issue a search for a $\langle$key, value$\rangle$ pair via the `FIND_VALUE` message. When a node receives a `FIND_VALUE` message, if it has the value, it returns it; otherwise, it returns the $\langle$IP address, port, ID$\rangle$ triplet of k nodes it knows of that are closest to the key. This command is used by the bots to retreive the appropiate information, by using appropriate search keys that are either hard-coded into the bot code or can be easily generated by it.

Storm employed Overnet as a pull-structured search-query mechanism that it adapts for its C&C infrastructure. The search-query mechanism operates of follows: for any given day, each node within the Storm botnet, generates at random 1 out of the 32 possible search keys by using a secret search key generation function, implemented in the Storm bot code [13]. The botnet operators can therefore store values associated with these 32 keys on selected nodes within the botnet. The values for the search keys are believed to be obfuscated data that the nodes can use to compute contact information for re-directors, *i.e.* bots that act as proxies for the principal HTTP servers, also known as *motherships*, through the use of a fast-flux DNS network [20]. The botnet nodes can consequently pull new commands and updates from designated repositories by using `FIND_VALUE` messages to search for the key they generated for that day, and then compute the contact information for these repositories and subsequently querying them via HTTP.

3 Related work

The performance analysis contained in the original work describing the Kademlia algorithm [17] concentrated on the maximum latency for any node to obtain accurate answers to its queries; a notion that is captured by the diameter of the underlying message-passing graph. This is a natural measure in the context of the traditional file-sharing applications of P2P networks, but it is not a sufficient indicator of performance in the context of botnets. Thus, our previous work introduced and studied other measures more suited to describe the performance of C& C infrastructures, such as distribution of shortest path lengths and sizes of bounded radius neighbourhoods, in order to describe the relative performance of various mitigation strategies.

We used these measures [5] to compare four complex network structures: Overnet (a structured P2P overlay network), Gnutella (an unstructured P2P overlay network), the Erdős-Rényi random graph model and the Barabási-Albert scale-free network model. The aim of that work was to compare them and determine the relative advantages and disadvantages they offered as botnet C&C infrastructures. In latter work [4], we used the same measures and similar simulation techniques to determine: (a) the effectiveness of Sybil attacks as a mitigation strategy against Storm botnet, and (b) identify the parameters which affect the efficacy of Sybil attacks in disrupting the botnet C&C structure. The graph models we utilised to represent the Overnet P2P network presumed that the subset of nodes in the peer lists that were being contacted were chosen at random. In essence, these models only captured the peer list limiting behaviour of Storm, while making abstraction of the internal structure and peculiar use of Kademlia's k-buckets in the value search. As such, the underlying subgraphs over which Overnet messages transited were modelled as regular graphs, as a first order approximation. In this paper, we use a refined graph model that more accurately captures the Kademlia search query mechanism.

With regards to Sybil attacks being used for botnet mitigation, Holz, Steiner, Dahl, Biersack and Freiling [13] presented a case study showing how to use sybils to infiltrate the Storm botnet. The authors used an Overnet crawler which issues route requests in a breadth-first search manner in order to find peers currently participating in the Overnet or Storm network. The two main goals of their study were: (a) determine the number of active Storm nodes by infiltrating the botnet with sybils and use the sybils to "spy" on the Storm network; and (b) determine the effect of pollution attacks (index poisoning) launched from these sybils by

posting polluted values associated with Storm search keys. The authors evaluated the effectiveness of the pollution attack by simultaneously polluting the value of a key used by Storm, and crawling the Storm network and searching for that key. Their investigation indicated that by polluting the keys that Storm uses, they were able to disrupt the botnet communication.

The portion of our work related to Sybil attacks on Storm botnet is complementary to that of Holz *et al.*'s work, but it can be differentiated as follows. First, we wish to quantitatively study how sybil attack parameters such as the size of the sybil population relative to that of the botnet or the duration of the attack affect C&C effectiveness. Second, we wish to know what operational or design parameters chosen by the botmaster could potentially reduce the effectiveness of such attacks.

In addition to Holz *et al.*'s work, a number of research efforts have focused on disrupting non-botnet P2P overlay networks. The typical scenario for such attacks would be that of representatives of the Recording Industry Association of America (RIAA), or like organisations, attempting to disrupt P2P networks, presumably in order to thwart or to discourage the download of copyrighted digital files. In this context, in addition to poisoning the indexes (i.e. the DHT) with wrong values (pointing to incorrect addresses), *pollution attacks* can also be considered where fake or bogus content is inserted into the system, with the indexes being polluted with the additional extra $\langle \text{key}, \text{value} \rangle$ pairs pointing to this bogus content. This reduces performance and renders more difficult the localisation by users of the legitimate content, i.e. the *goodput* of the P2P file sharing system. We now provide a quick overview of some of the related research work in this area, and where necessary, indicate how our work differs from the work in question.

Christin, Weigend and Chuang [3] conducted a measurement study in content availability for four P2P file-sharing overlay networks: Gnutella, eDonkey, Overnet, and FastTrack. Their work investigated the impact of pollution and poisoning on content availability in file sharing networks. Their results indicate that in order for pollution and poisoning to be effective in reducing content availability of popular files in the P2P networks they considered, the polluted versions of the files need to be injected in the network on massive scales.

Dumitriu, Knightly, Kuzmanovic and Stoica [8] presented analytical modelling and simulation studies involving the Gnutella [9] P2P system. Their studies investigated the effect of file-targeted attacks and network-targeted attacks. In the former, attackers put large number of corrupted versions of a single file on the network; whereas in network-targeted attacks, attackers respond to network queries for any file with erroneous information. Their results indicate that success of file-targeted attacks depend on the clients behaviour and that the attack succeeds over the long term only if the clients are unwilling to share files and they are slow to remove corrupt files from the machine. The network-targeted attacks, however, are effective in decreasing the system goodput.

Singh, Ngan, Druschel and Wallach [23] studied the impact of Eclipse attacks in P2P overlay networks. In an Eclipse attack, a set of malicious colluding nodes arrange for targeted correct nodes to be paired with only members from the malicious coalition. If successful, the attackers can mediate and ultimately control the traffic intended for the correct nodes, and in so doing "eclipse" the nodes from the rest of the network. Their work indicates that known defences are limited in preventing Eclipse attacks. Nonetheless, Holz *et al.* [13] do show that whereas Eclipse attacks are feasible in Kad —a Kademlia-based distributed hash table (DHT)— it is infeasible in Overnet, because Overnet keys are distributed throughout the entire hash table space, rather than be restricted to a particular zone.

Liang, Naoumov and Ross [16] presented a methodology for estimating index poisoning levels and pollution levels in structured and unstructured file sharing networks.

Naoumov and Ross [18] show how index poisoning and routing poisoning can be used to create denial-of-service engines out of P2P systems. With index poisoning, the attackers insert bogus records into the P2P index system. These bogus records indicate that a popular file is located at the targeted IP address and port number. This can result in large amount of traffic which can overwhelm the targeted node. In the case of routing poisoning, the attackers attempt to make the targeted node a neighbour of a large number of P2P nodes. This can result in the targeted node receiving large amount of maintenance traffic and, hence, be the victim of a bandwidth-consuming DDoS attack. The use of such a technique in attacking a botnet is not appropriate as only a single node (or a few) would be isolated, leaving the rest unaffected.

Finally, Steiner, En-Najjary and Biersack [24] indicate how Kad can be used and misused. The authors shows how Sybil attacks can be used to perpetuate Eclipse and denial-of-service attacks in Kad. They also presented a centralised scheme for preventing sybils from gaining access to Kad.

Unfortunately, however, while the idea of pollution and poisoning attacks is not new to botnet mitigation strategies, the quantitative studies that have been conducted on their performance do not apply directly to the context of botnet mitigation. The main reason why is that performance objectives of the bot master (getting the most machines to get the update message/command in time) do not coincide with those of the file-sharing P2P network user (maximise goodput of shared material). Thus the need for botnet-specific research on the quantitative effectiveness of such techniques.

Other work on P2P botnets include the following: Griz-

zard, Sharma, Nunnery and Dagon [10], Dittrich and Dietrich [6], and Ruitenbeek and Sanders [22]. The authors of [10] provided an overview of P2P botnets, and presented a case study of Storm. In [6], the authors presented a botnet study which examine the features of Nugache P2P botnet and compare how current proposals for dealing with P2P botnets would or would not affect a pure-P2P botnet such as Nugache. Finally, [22] presented a stochastic model of P2P botnet formation. The authors used the stochastic model to examine how different factors impact the growth of botnets. The focus of our work is different from the above mentioned works on P2P botnet, in that we examine the effect of Sybil attacks on Kademlia-based P2P botnets.

4 Kademlia-based P2P botnet graph model

Kademlia-based P2P botnets can be modelled as a directed graph $G = (V, E)$, where V and E are the vertex set and edge set of G, respectively. For any nodes $u, v \in V$, the edge (u, v) exists iff the $\langle$IP address, port, ID$\rangle$ triplet for node v is in one of node u's k-buckets, *i.e.* somewhere in u's peer list. This graph G thus represents the "knowledge" that peers have of each other. As outlined in Section 2, the nodes of Kademlia-based P2P botnets utilise `FIND_VALUE` queries to find search keys whose values can be used to compute contact information for re-directors in the botnet fastflux network. When a node v receives a `FIND_VALUE` message from a node u, it selects k nodes it knows of that are closest to the search key, and sends the $\langle$IP address, port, ID$\rangle$ triplet of the k nodes back to u. In the meanwhile, u sends `FIND_VALUE` messages to the k closest nodes it knows, as it receives information about nodes closer and closer to the search keyu. It can be deduced from the above that only nodes whose IDs are relatively close to the key (whose value is being queried) will be on the search path initiated from a node u for this key. Furthermore, not all edges in E will be utilised in the search of a given key, no matter who initiates the search.

To model the messages exchanged between nodes involved in the searches of a given key z, irrespective of the initiating node, we construct the subgraph of $G_z = (V, E_z)$ as follows. For all u in V, let $d(u, z)$ be in the range $(2^i, 2^{i+1}]$, *i.e.* $2^i < d(u, z) \leq 2^{i+1}$. This implies that u will initially look for answers from nodes in the i-th k-bucket. Then we say that given a node v in u's peer list, *i.e.* $(u, v) \in E$, the corresponding edge (u, v) will also be in E_z only if one of the following conditions hold:

1. v is in the i-th k-bucket of u, containing records whose IDs are closest to z, *i.e.* $d(v, z) \in (2^i, 2^{i+1}]$
2. v is *not* in the i-th k-bucket (because it contains less than k entries), but v is one of the k closest nodes to z in u's peer list.

Note that because of the 2nd condition, any such subgraph of more than k nodes will necessarily be k-regular, *i.e.* all nodes will have exactly k neighbours in G_z.

We indicated in Section 2 that the botnet operators store $\langle$key, value$\rangle$ pairs on selected nodes. The Kademlia protocol stipulates that a $\langle$key, value$\rangle$ pair should be stored on the k nodes whose IDs are closest to the search key. Let $K_z \subseteq V$ represent this set of nodes on which the $\langle z$, value$\rangle$ pair will be so stored, where $|K_z| \leq k$. These nodes constitute the "kernel" of the graph G_z in that any node in K_z should know the value alllowing bots to find re-directors.

Nodes in the network will find eventually find a node in K_z by using the `FIND_VALUE` messages to query peers for the key z. More precisely, the `FIND_VALUE` query works as follows. The initiator node u selects α amongst the k closest nodes to z (where α is parallelisation parameter typically set to 3). It then sends each them in parallel a `FIND_VALUE` message for z. The selected nodes are taken from a single k-bucket, or if the k-bucket which contains records whose ID are closest to the key z has less than k records, the additional nodes are selected from adjacent k-bucket(s). Each of the queried nodes does the following: if it has the value node u seeks, it sends it to v, otherwise it sends the k nodes that it knows of which are closest to z. In the recursive step, u sends `FIND_VALUE` messages to nodes it learnt about through previous queries (as long as they are within amongs the k closest to z it knows of). The recursive process continues until a node returns the value node v seeks, or until v gets responses from the k closest nodes it has seen. As described in [17], this process is guaranteed to converge in $O(\log |V|)$ iterations; in other words, the diameter of G_z is bounded by $O(\log |V|)$.

Let $V_z^{(r)}$ represent the set of nodes who have a path in G_z of length r or less from them to a node in K_z, *i.e.* $u \in V_z^{(r)}$ if $\exists v \in K_Z$ such that the distance dist$(u, v) \leq r$. This means, that nodes in $V_z^{(r)}$ will necessarily get a correct answer to the query from a node in K_z within r iterations or less of the search algorithm. Hence we call this set the *reachable botnet* with parameter r.

From the size of the active botnet, we then obtain *reachable ratio* $\nu(r) = |V_z^{(r)}|/|V|$, which we postulate is an effective measure that can be used to assess the efficacy of Kademlia-based P2P botnets C&C structure. It represents the fraction of nodes in the network that can obtain the correct value (associated with a given botnet search key) as a result of `FIND_VALUE` queries terminating within r iterations, and consequently can receive updates and new instructions from the botnet operators.

5 Simulation setup

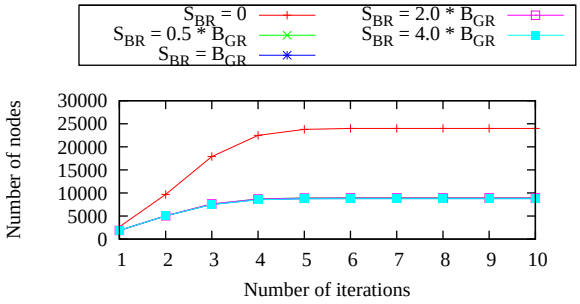
We assessed the effectiveness of Sybil attacks in disrupting Kademlia-based P2P botnet C&C structure, using the graph model we presented in Section 4. In our simulation analysis, we assumed that sybils join the botnet and send frequent `PING`, `FIND_NODE` and `FIND_VALUE` messages in attempts to get their identification triplets inserted into other nodes' k-buckets. As indicated in Section 2, when a node u participating in an Overnet network, receives a message from a another node v, if the k-bucket where the ID of v "belongs" is not full, v's identification triplet gets inserted into the k-bucket. If the given k-bucket is full, u pings the nodes (starting from the least recently seen) whose identification triplets appear in the k-bucket. If all the nodes respond to u's ping message, u discard v's information; otherwise the identification triplet of the least recently seen node which does not respond to the ping message is replaced with u's identification triplet.

In our sybil attack model, when sybils identification triplets are inserted into k-buckets and subsequently receive `FIND_VALUE` messages, they send fictitious values to the querying nodes. The Kademlia protocol stipulates that a node terminates its `FIND_VALUE` query as soon as it receives the value it searches. Consequently, if a sybil responds to node v's `FIND_VALUE` query before any of the "real" nodes, v will not find an "authentic" value for the search key it sought. For our simulation analysis, we assumed that each node has an equal probability of responding first to a `FIND_VALUE` query issued simultaneously. Our simulation addresses the following questions:

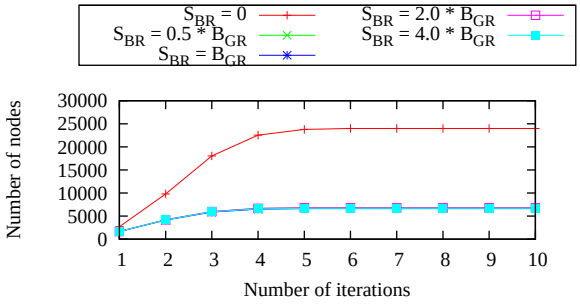
1. How does the efficacy of Sybil attacks vary as a function of the relative growth rates of the sybil and botnet populations? More precisely, if B_{GR} represents the growth rate of the botnet (new infections minus natural attrition per simulation step), and S_{BR} represents the sybil birth rate, then we looked at values of S_{BR} equal to 0, $0.5 * B_{GR}$, B_{GR}, $2 * B_{GR}$ and $4 * B_{GR}$. We assume that sybils are not subject to attrition, of course.

2. What impact does the duration of the attack have on the efficacy of the botnet?

To answer the above questions, we performed 50 sets of simulation runs. In each run, the simulated Overnet network started with 20,000 nodes and grew for Δt time steps; where Δt varied from 20 to 40. In each run, both the node IDs and the search keys are chosen uniformly at random. We assumed a botnet node birth rate of 2% per time step and a node death rate of 1% per simulation times step, where the locations in the botnet of node births or deaths is chosen at random from a uniform distribution. Hence, the net botnet growth rate per time step B_{GR} was kept constant at 1%



(a) Sybils ID chosen randomly



(b) Sybils assigned IDs that are close to the search key

Figure 1. Sybil attacks results for $\Delta t = 20$.

of the botnet's size for all the simulation runs; whereas the rate at which sybils were added to the network, *i.e.* the sybil birth rate (S_{BR}) varied from 0 to $4 * B_{GR}$. At the end of each simulations, the size $V_z^{(r)}$ of the reachable botnet and the reachable ratio $\nu(r)$ was computed for various values of r from 1 to 10. In all cases, the same Kademlia implementation parameters were chosen ($k = 20$, $\alpha = 3$) but we chose to limit the overall size of the peer list to 200 nodes.

6 Results and discussion

Figures 1 through 3 show the size of the reachable botnet (y-axis) versus the parameter r (x-axis), for various values of sybil birth rate S_{BR} shown in various lines, and for sybil attack duration of length $\Delta t = 20, 30$ and 40, respectively. Each figure contains two graphs comparing the case where a) sybil IDs are chosen at random, and b) sybils are assigned IDs that are close to the search key (informed choice). The baseline within these graphs is provided by the $S_{BR} = 0$ curve (the uppermost one), which shows the botnet's behaviour when no disruptions are present, *i.e.* no sybil attack.

What these figures make clear is that once the Sybil attack is begun: (a) the ability to reach nodes falls off quite dramatically with only approximately 20% of the botnets nodes being reachable, *i.e.* able to find a true response to their key-value pair search, (b) the degree of the impact is largely indepedent of the scale of the Sybil attack, *i.e.* the

(a) Sybils ID chosen randomly

S_{BR}	Number of sybils	Reduction in ν	Standard deviation
0	0	0%	0.04%
$0.5 * B_{GR}$	2,000	65.52%	4.19%
B_{GR}	4,000	63.36%	4.78%
$2 * B_{GR}$	8,000	62.78%	3.69%
$4 * B_{GR}$	16,000	63.47%	4.08%

(b) Sybils assigned IDs that are close to the search key

S_{BR}	Number of sybils	Reduction in ν	Standard deviation
0	0	0%	0.01%
$0.5 * B_{GR}$	2,000	71.71%	3.29%
B_{GR}	4,000	72.39%	3.79%
$2 * B_{GR}$	8,000	71.73%	2.98%
$4 * B_{GR}$	16,000	72.14%	3.46%

Table 1. Sybil attacks results, $\Delta t = 20$, final number of bots = 24,000.

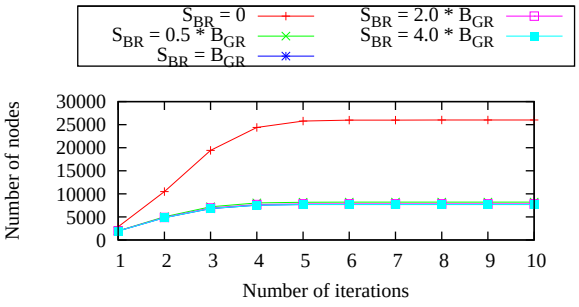
(a) Sybils ID chosen randomly

S_{BR}	Number of sybils	Reduction in ν	Standard deviation
0	0	0%	0.01%
$0.5 * B_{GR}$	3,000	68.48%	2.88%
B_{GR}	6,000	70.40%	2.97%
$2 * B_{GR}$	12,000	69.83%	3.31%
$4 * B_{GR}$	24,000	70.39%	2.78%

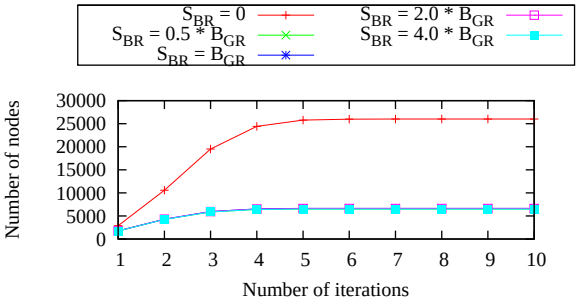
(b) Sybils assigned IDs that are close to the search key

S_{BR})	Number of sybils	Reduction in ν	Standard deviation
0	0	0%	0.01%
$0.5 * B_{GR}$	3,000	74.95%	2.81%
B_{GR}	6,000	74.46%	3.23%
$2 * B_{GR}$	12,000	74.54%	2.76%
$4 * B_{GR}$	24,000	74.93%	2.87%

Table 2. Sybil attacks results, $\Delta t = 30$, final number of bots = 26,000.

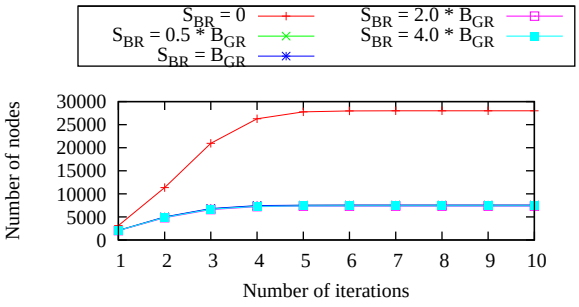


(a) Sybils ID chosen randomly

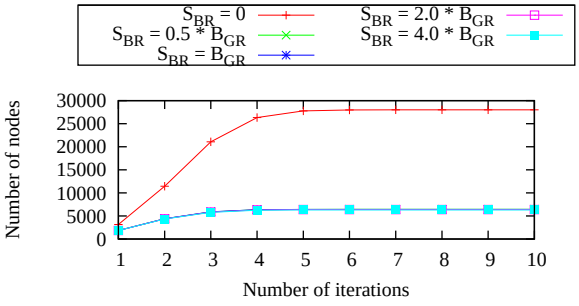


(b) Sybils assigned IDs that are close to the search key

Figure 2. Sybil attacks results for $\Delta t = 30$.



(a) Sybils ID chosen randomly



(b) Sybils assigned IDs that are close to the search key

Figure 3. Sybil attacks results for $\Delta t = 40$.

(a) Sybils ID chosen randomly

S_{BR}	**Number of sybils**	**Reduction in ν**	**Standard deviation**
0	0	0%	0.01%
$0.5 * B_{GR}$	4,000	73.14%	2.90%
B_{GR}	8,000	72.99%	2.77%
$2 * B_{GR}$	16,000	73.78%	2.81%
$4 * B_{GR}$	32,000	73.61%	3.03%

(b) Sybils assigned IDs that are close to the search key

S_{BR}	**Number of sybils**	**Reduction in ν**	**Standard deviation**
0	0	0%	0.01%
$0.5 * B_{GR}$	4,000	77.03%	2.29%
B_{GR}	8,000	77.06%	2.57%
$2 * B_{GR}$	16,000	77.19%	2.64%
$4 * B_{GR}$	32,000	77.46%	2.80%

Table 3. Sybil attacks results, $\Delta t = 40$, final number of bots = 28,000.

$S_{BR} = 0.5 * B_{GR}$ Sybil attack performs as well as the 8-times more expensive $S_{BR} = 4 * B_{GR}$ attack, and (c) the results for random and informed Sybil placements are also nearly identical.

Part of the explanation for these somewhat counter-inutitive results is that the P2P k-bucket update protocol makes sure that only the most active nodes remain on it. This is accomplished through the two-stage drop process that is used prior to removing any node from a k-bucket, with the second stage requiring that nodes be non-responsive to a direct query prior to their being dropped. Hence, sybils can only replace nodes within the botnet that have died, as active nodes are never selected for replacement. Therefore, the number of sybils active in the botnet only needs to reach a level where the probability of a sybil being selected as the node to replace a dead node is higher than the probability that a real botnet node is selected. Injecting sybils above this rate will have little to no impact on the effectiveness of the Sybil attack as the probability that one of these extra sybils will be selected as a replacement node decreases as more sybils are added, given that these sybils are more likely to be further away in XOR distance from the node that is being replaced. In other words, only a relatively small number of nodes close to the search key have a high probability of being selected as the replacement node for a dead node in the corresponding k-bucket, hence, the Sybil attack rate only needs to be high enough to make it highly probable that one of the selected close node will be a sybil and not an active botnet node.

Tables 1 through 3 detail more clearly the differences between the random Sybil placement strategy and the more informed strategy of placing the Sybils near the location of the stored key-value pair, where the former models the composite of a set of uncoordinated defenders each engaging in their own Sybil attacks whereas the latter models a coordinated Sybil attack strategy. More particularly, although an advantage of between 3.41%-9.03% is gained in effectiveness via the informed Sybil placement strategy, the drop in effectiveness via the more easily implemented random strategy is not that severe, particularly when it is viewed in light of the global-scale of modern botnets. Moreover, the degree of separation between the strategies quickly decreases as the duration of the Sybil attacks are increased. Once again this is due to the fact that only dead nodes within peer lists are eligible for replacement, *i.e.* as the attack duration increases more bot attrition occur leading to more Sybils being selected as replacement nodes. Additionally, it can be observed that slowing down the Sybil attack is a far more effective strategy than just increasing the number of Sybils, *e.g.* 4000 Sybil with $\Delta t = 40$ with random placement provides 73.14% effectivness whereas $16,000$ Sybil with $\Delta t = 20$ with random placement provides only 63.47% effectivness. This is important if the resources of the Sybil attacker are such that the total number of sybil is constrained, *e.g.* by limitation of bandwith and/or CPU. In this case, it seems that trading-off number of sybils for a slower rate of activity and a longer attack is an advantageous strategy.

An interesting observation from these tables is also that the Sybil attack effectiveness appear to be heading to asymptotic levels of around 75% and 77.5% effectiveness for the random and informed placement strategies respectively. Currently, it is unclear whether these constitue true asymptotes and if so, why do they occur at these levels.

7 Conclusion and Future Work

Within this work graph theory-based simulations have been used to explore the effectiveness of Sybil attack strategies against modern P2P-based botnets. The resulting analysis has shown, at least within the context of the simulations run, that a random Sybil placement strategy performs nearly as well as a more informed placement strategy and that extending the Sybil attack duration is more important than just increasing the number of Sybils that are used. This has important implications for real-world botnet defenses in that it strongly suggest that independent defenders need not coordinate their Sybil attack strategies in order to effectively disrupt the botnet. More particularly, the highly diffuse nature of P2P botnets, which the botmasters have exploited to gain robust networks, is also what leads to the random Sybil placement strategies being nearly as effective as informed placement strategies. Hence, it is unlikely that the botmasters can easily adapt their botnets to be more resilient

to random placement strategies without making the botnets less diffuse and, hence, easier to "roll-up" via other known botnet countermeasures.

The results also imply that there exists an optimal Sybil attack rate that is itself tied to the specific birth and death rates that occur on the given botnet. Deriving this optimal Sybil attack rate requires the development a deeper analysis of the graph theory models in order to extract the ratio of the likelihoods that a given dead node will be replace by either a Sybil or an active botnet node. Deriving these likelihoods is a non-trivial problem given their inherent dependence on the location of the dead node within the botnet as well as its location within the various peer lists to which it belongs. These likelihoods also depend directly on the issue of which new node is the first to respond to a given peer list replacement request. This in turn is affected by the underlying network fabric upon which the P2P botnet exists as on overlay; hence, packet-level simulation studies may be required to accurately assess optimal Sybil attack rates under varying conditions. Finally, there is a need to determine whether the observed asymptotic behaviors represent true asymptotes and, if so why, or whether they are merely artifacts of the specific simulation runs or methods.

References

[1] P. Barford and V. Yegneswaran. An inside look at botnets. *Advances in Information Security*, 27:171–191, March 2007.

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

[3] N. Christin, A. Weigend, and J. Chuang. Content availability, pollution and poisoning in file sharing peer-to-peer networks. In *Proceedings of the* 6^{th} *ACM conference on Electronic commerce*, pages 68–77, June 2005.

[4] C. Davis, J. Fernandez, S. Neville, and J. McHugh. Sybil attacks as a mitigation strategy against the storm botnet. In *Proceedings of the* 3^{rd} *International Conference on Malicious and Unwanted Software (MALWARE 2008)*, October 2008.

[5] C. Davis, S. Neville, J. Fernandez, J.-M. Robert, and J. McHugh. Structured peer-to-peer overlay networks: Ideal botnets command and control infrastructures? In *Proceedings of the* 13^{th} *European Symposium on Research in Computer Security (ESORICS'08)*, October 2008.

[6] D. Dittrich and S. Dietrich. P2p as botnet command and control: A deeper insight. In *Proceedings of the* 3^{rd} *International Conference on Malicious and Unwanted Software (MALWARE 2008)*, October 2008.

[7] J. Douceur. The sybil attack. In *Proceedings of the* 1^{st} *International Workshop on Peer-to-Peer Systems*, pages 251–260, March 2002.

[8] D. Dumitriu, E. Knightly, A. Kuzmanovic, I. Stoica, and W. Zwaenepoel. Denial-of-service resilience in peer-to-peer file sharing systems. *ACM SIGMETRICS Performance Evaluation Review*, 33:38–49, June 2005.

[9] Gnutella Project. Gnutella. `http://www.gnutella.com`, March 2001.

[10] J. Grizzard, V. Sharma, C. Nunnery, B. Kang, and D. Dagon. Peer-to-peer botnets: overview and case study. In *Proceedings of the* 1^{st} *Workshop on Hot Topics in Understanding Botnets (HotBots 2007)*, April 2007.

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

[12] M. Hanke and F. Hauser. On the effects of stock spam e-mails. *Journal of Financial Markets*, 11:57 – 83, 2008.

[13] 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 *Proceedings of the* 1^{st} *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET'08)*, April 2008.

[14] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. Voelker, V. Paxson, and S. Savage. Spamalytics: an empirical analysis of spam marketing conversion. In *Proceedings of the* 15^{th} *ACM conference on Computer and communications security (CCS'08)*, October 2008.

[15] K. Kutznet and T. Fuhrmann. Measuring large overlay networks the overnet example. In *KiVS*, May 2006.

[16] J. Liang, N. Naoumov, and K. Ross. The index poisoning attack in p2p file sharing systems. In *Proceedings of* 25^{th} *IEEE International Conference on Computer Communications (INFOCOM 2006)*, April 2006.

[17] P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the XOR metric. In *Revised Papers from the* 1^{st} *International Workshop on Peer-to-Peer Systems (IPTPS '02)*, March 2002.

[18] N. Naoumov and K. Ross. Exploiting p2p systems for DDoS attacks. In *Proceedings of the* 1^{st} *international conference on Scalable information systems (INFOSCALE 2006)*, May 2006.

[19] A. Pathak, Y. Hu, and Z. Mao. Peeking into spammer behavior from a unique vantage point. In *Proceedings of the* 1^{st} *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET'08)*, April 2008.

[20] P. Porras, H. Saidi, and V. Yegneswaran. A multi-perspective analysis of the storm (peacomm) worm. Technical report, Computer Science Laboratory, SRI Internatal, October 2007.

[21] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. In *Proceedings of the* 6^{th} *ACM SIGCOMM conference on Internet measurement*, October 2006.

[22] E. Ruitenbeek and W. Sanders. Modeling peer-to-peer botnets. In *Proceedings of the* 5^{th} *International Conference on Quantitative Evaluation of Systems (QEST '08)*, pages 307–316, September 2008.

[23] A. Singh, T.-W. Ngan, P. Druschel, and D. Wallach. Eclipse attacks on overlay networks: Threats and defenses. In *Proceedings of* 25^{th} *IEEE International Conference on Computer Communications (INFOCOM 2006)*, pages 1–12, April 2006.

[24] M. Steiner, T. En-Najjary, and E. Biersack. Exploiting kad: possible uses and misuses. *ACM SIGCOMM Computer Communication Review*, 37:65–70, October 2007.

[25] W. Strayer, R. Walsh, C. Livadas, and D. Lapsley. Detecting botnets with tight command and control. In *Proceedings of the* 31^{st} *IEEE Conference on Local Computer Networks*, November 2006.