

Sybil attacks as a mitigation strategy against the Storm botnet

Carlton R. Davis*, José M. Fernandez*, Stephen Neville†, John McHugh‡

*École Polytechnique de Montréal,

Email: {carlton.davis, jose.fernandez}@polymtl.ca

†University of Victoria,

Email: sneville@ece.uvic.ca

‡Dalhousie University,

Email: mchugh@cs.dal.ca

Abstract—The Storm botnet is one of the most sophisticated botnet active today, used for a variety of illicit activities. A key requirement for these activities is the ability by the botnet operators to transmit commands to the bots, or at least to the various segmented portions of the botnet. Disrupting these command and control (C&C) channels therefore becomes an attractive avenue to reducing botnets effectiveness and efficiency. Since the command and control infrastructure of Storm is based on peer-to-peer (P2P) networks, previous work has explored the use of *index poisoning*, a disruption method developed for file-sharing P2P networks, where the network is inundated with false information about the location of files.

In contrast, in this paper we explore the feasibility of *Sybil attacks* as a mitigation strategy against Storm. The aim here is to infiltrate the botnet with large number of fake nodes (*sybils*), that seek to disrupt the communication between the bots by inserting themselves in the peer lists of “regular” bots, and eventually reroute or disrupt “real” C&C traffic. An important difference with index poisoning attacks is that sybil nodes must remain active and participate in the underlying P2P protocols, in order to remain in the peer list of regular bot nodes. However, they do not have to respond to the botmaster’s commands and participate into illicit activities. First, we outline a methodology for mounting practical Sybil attacks on the Storm botnet. Then, we describe our simulation studies, which provide some insights regarding the number of sybils necessary to achieve the desired level of disruption, with respect to the net growth rate of the botnet. We also explore how certain parameters such as the duration of the Sybil attack, and botnet design choices such as the size of a bot’s peer list, affect the effectiveness of the attack.

I. INTRODUCTION

The Storm botnet is currently one of the most sophisticated botnet infrastructures. It has been garnering much attention both in the anti-virus research community and electronic media [1]–[6]. Estimates of the number of computers infected by the Storm malware vary widely and range from 40,000 to 2 million [7]–[10]. Storm covers a wide geographical region: the authors of [7] reported that they found Storm bots in over 200 countries. Disinfection efforts by Microsoft have had some success in removing the Storm malware from large number of computers. Microsoft claims that in the last four months of 2007, its Malicious Software Removal Tool (MSRT) disinfected more than 526,000 personal computers plagued with the Storm malware [11]. This prompted personnel representing Microsoft to claim that the corporation has succeeded

in crushing Storm [11]. Researchers agree that the size of the botnet has reduced compared to its size in mid 2007 [12]–[14]. However, some researchers believe that the reduction in size is due to the botnet operators’ design choices rather than due to Microsoft disinfection efforts [15]. Storm continues to recruit new bots, likely at a higher rate than bots are being disinfected. IronPort reported that Storm infects and re-infects about 900,000 computers monthly [9]. The botnet operators however, appear to be leaving large number of bots dormant for given time periods and rotate the activities of the botnet between sets of bots so that the bots can take turns being dormant and active in ways that minimise the detection of their activities.

It is becoming increasingly evident that the main goal of Storm creators and operators is to generate revenue from illicit activities perpetuated through the botnet. The illicit or criminal activities for which it is being used include email spams, phishing attacks, instant messaging attacks, “pump-and-dump” stock scams, marketing and sales of counterfeit pharmaceuticals, etc. It also appears that the operators of Storm have partitioned the botnet by assigning unique encryption keys to different segments, and rent out these segments individually (in part or in their entirety) to other criminals entities to use in their own illicit activities. A key requirement for this “business model” is that the botnet operators must be able to transmit commands to the bots, so that they may perform the tasks that they have been hired to do. Disrupting these command and control (C&C) channels therefore becomes an attractive avenue to reducing botnets effectiveness and efficiency. Previous generations of botnets, some still being currently used, employ IRC to disseminate commands to bots that are “listening in” to designated botnet C&C chat rooms. These botnets are relatively easy to detect and disrupt once the server is identified. Probably for that reason, newer botnets including Storm, use more resilient peer-to-peer (P2P) networks in order to disseminate commands to the bots.

In previous work [16], we have investigated the effectiveness of collaborative disinfection schemes against P2P-based botnet C&C infrastructures, where information regarding the connectivity of a botnet is used in the actual disinfection process. Our simulation studies revealed that this approach

works well for scale-free networks such as Barabási-Albert networks [17] and unstructured P2P overlay networks such as Gnutella [18], and to a lesser extent, even for Erdős-Rényi random networks [19]. The gain in effectiveness is, however, minimal for structured P2P overlay networks such as Overnet [20], precisely the network architecture Storm utilises. The resilience of Overnet against targeted attack—where selected nodes are removed following a deliberate method—can be attributed to the fact that the protocol stipulates an upper bound for the number of nodes that can be connected to any given node. Even if collaborative disinfection has been shown to be significantly more effective compared to random disinfection, owing to the large geographical area that Storm bots resides in, the implementation of these disinfection schemes is limited due to the jurisdiction issues involved.

There is therefore the need to explore other mitigation schemes which can be used in combination with disinfection efforts to attenuate the threat of the Storm botnet. In this paper, we present our findings pertaining to a feasibility study we conducted on the effectiveness of the *Sybil attack* [21] as a mitigation strategy against Storm. The main contributions of the paper are the following:

- 1) We provide some pointers regarding sybil population size—compared to the botnet growth rate—that are necessary to achieve desired level of disruption of the botnet C&C.
- 2) Our results give some indications about how the duration of the Sybil attack affects the degree of effectiveness of the attack. These findings can be used to fine-tune attacks to maximise efficiency.
- 3) We present results which indicate that botnet operator design choices, such as the size of bots' peer-list, do not seem to have any significant effect on the effectiveness of Sybil attacks.

The rest of the paper is structured as follows. Section II starts with background information about the Storm botnet, the Overnet P2P overlay network that it uses for its C&C infrastructure, and how it uses it. We present related works in Section III, and outline how our work differs from previous research efforts. Toward the end of this section, we outline how a practical Sybil attack can be mounted on Storm. In Section IV, we give detail about the simulation setup and the performance measures we used to assess the effectiveness of Sybil attacks. We present the simulation results in Section V, and in Sections VI and VII, we discuss the significance of the results, summarise the findings and present ideas for future work.

II. BACKGROUND

Storm uses a modified Overnet P2P protocol for its communication architecture. We present an overview of Overnet below.

A. Overview of Overnet

Overnet is a popular proprietary file sharing overlay network protocol which implements a distributed hash table (DHT)

algorithm called Kademlia [22]. Each node participating in an Overnet network generates a 128-bit ID for itself when it first joins the network. The ID is transmitted with every message the node sends. This permits recipients of messages to identify the sender's existence, as required.

Each node in an Overnet network stores contact information about some of the other nodes in the network, in order to appropriately route query messages. This information is organised in *lists* of (IP address, UDP port, ID) triplets, where each list contains those nodes whose distance is between 2^i and 2^{i+1} from itself, where $0 < i < 128$; thus there can be potentially up to 128 such lists. Here, the distance $d(x, y)$ between two IDs x and y is defined as the bitwise exclusive or (XOR) of x and y interpreted as an integer, i.e., $d(x, y) = x \oplus y$.

These lists are also referred to as k -buckets as they contain at most k nodes, where k is a configurable parameter. Records in a k -bucket are kept sorted by time last seen, ordered by 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
 - If there is room left in the k -bucket, the sender's triplet is simply added to the tail of the k -bucket.
 - Otherwise, the recipient pings the node n that is at the head of the appropriate 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 appropriate 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 provides a measure of protection against denial-of-service attacks, since only inactive nodes are forced out of k -buckets.)

The Kademlia protocol (which Overnet implements) provides the four message types outlined below:

- PING: This message is used to probe a node to determine if it is on-line.
- STORE: This message instructs a node to store a (key, value) pair for later retrieval; key is a 128-bit quantity (e.g. the hash of a file identifier) and value is the entity (e.g. the location of an audio or video file) being search for. If a node wishes to publish a (key, value) pair, it locates the k closest nodes to the key and send them a STORE message.
- FIND_NODE: This allows a node to search for a node ID (a 128-bit quantity). When a node receives a FIND_NODE message, it returns the (IP address, UDP port, ID) triplet of the k nodes it knows about that are closest to the ID. This procedure is thus

recursive.

- **FIND_VALUE:** A node can issue a search for a $\langle \text{key}, \text{value} \rangle$ 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 $\langle \text{IP address}, \text{UDP port}, \text{ID} \rangle$ triplet of the k nodes it knows of that are closest to the key.

A node n wishing to join an Overnet network must have the $\langle \text{IP address}, \text{UDP port}, \text{ID} \rangle$ triplet of at least one node which participates in the network. To join the network, n inserts its contact(s) in its k -buckets, then does a lookup for its own ID by sending the node(s) it knows of a **FIND_NODE** message to find nodes that are closest to its ID. The new node n can then populate its k -buckets based on messages it receives.

B. Storm C&C Infrastructure

The main difference between the Storm infrastructure and the Overnet P2P network is that Storm nodes XOR encrypts their messages using a 40-bit encryption key; whereas the regular Overnet nodes do not encrypt their messages. After the Storm binary installs itself on a victim's machine, the binary generates a 128-bit ID, and initialises its peer-list file (this file is called `spooldr.ini` in some versions of the binary [23] and it is the equivalent of Overnet's k -buckets) with a list of 100 to 300 (depending on the version of the binary) $\langle \text{ID}, \text{IP}, \text{port} \rangle$ triplets for peers which are hard-coded in the binary. The triplets are in the form $\langle \text{ID} \rangle = \langle \text{IP} \rangle \langle \text{port} \rangle 00$, where $\langle \text{ID} \rangle$ is the 128-bit node ID and $\langle \text{IP} \rangle \langle \text{port} \rangle 00$ is the IP address and UDP port in hexadecimal format with 00 appended [24]. For example, for the string

008052D5853A3B3D2A9B84190975BAFD=53855152054A00

the 128-bit node ID is

008052D5853A3B3D2A9B84190975BAFD,

the IP address is 83.133.81.82 (the decimal format of $0x53855152$), and the UDP port is 66890 (the decimal equivalent of $0x054A$).

The newly infected node uses the list of contacts to join the Storm botnet and the list is updated as necessary when the Storm node receives messages from other storm nodes. Communication within the botnet proceeds as follows: each Storm node generates 32 different 128-bit keys each day using a function of the form $f(d, r)$, which takes as input the current day (d) and a random number between 0 and 31 (r) [7]; and sends **FIND_VALUE** search queries to their contacts for these keys. Reverse engineering analysis performed by the authors of [24], reveals that the value associated with these keys are of the form `*.mpg;size=*`, where the asterisks are 16-bit hexadecimal numbers which are presumably used to compute the URL for a re-director—in the Storm fast-flux network [25]—which points to a machine where binary updates and commands for the bots are stored.

C. Methodology for mounting practical Sybil attacks

The information presented in the previous sections above, can be used to mount practical Sybil attacks on Storm botnet as follows. The Storm search keys for a given day can be ascertained by capturing the communication traffic from a Storm bot, as outlined in [7]. Similarly, the encryption key can also be ascertained by reverse engineering a Storm binary. Sybils can then be generated with distinguishable 128-bit IDs. Furthermore, a single IP address can be associated with several thousands Sybil IDs, and thus several thousands of sybils can be run from the same machine. First, with knowledge of the encryption key, the sybils can then infiltrate the Storm network by sending search queries for their own keys, or by sending numerous and unnecessary **PING**, **FIND_NODE** and **FIND_VALUE** messages to non-sybil nodes, in order to get their IDs in the k -buckets of as many of these nodes. Second, with knowledge of the Storm search keys for the given day, the sybils can then reply with appropriate false information about the corresponding values; this is called *index poisoning* in the P2P research literature¹. Other alternate disruption strategies are possible. For example, a “silent denial-of-service” attack is possible where the sybils simply do not reply to further **FIND_NODE** and **FIND_VALUE** queries after having infiltrated the network, thus reducing its performance. Finally, the insertion of completely passive sybils complying with the Overnet protocol but not necessarily with commands issued, while not directly disrupting the C&C infrastructure, would reduce botnet effectiveness and hurt the botnet operator's business by providing a false sense of botnet size.

III. RELATED WORK

Sybil attacks have already been used in the context of botnet mitigation. Holz, Steiner, Dahl, Biersack and Freiling [7] presented a case study showing how to use sybils to infiltrate the Storm botnet. The authors used an Overnet crawler which runs on a single machine. It 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) 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 search for that key. Their experiment showed that by polluting the keys that Storm uses, they were able to disrupt the botnet communication.

Our work is complementary to that of Holz *et al.*'s work. However, it can be differentiated in several ways. First of all, we wish to study the effect of sybil attacks on C&C as a whole, not just that of the index poisoning attacks. As mentioned

¹Note however that in the botnet research literature, the term *index poisoning* is used more specifically to refer to attacks that try to fool the bots into going to the wrong URL for getting new code and commands.

earlier, other disruption strategies are possible. Second, we wish to quantitatively study how the size of the sybil population relative to that of the botnet affects C&C effectiveness. Finally, we wish to know what design parameters chosen by the botnet master could potentially reduce the effectiveness of such attacks; in other words, we want to find out how resilient such attacks are against botnet adjustments used as counter-counter-measures.

Beyond Holz *et al.*'s work, there is an abundant line of research 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 copyright 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 (key, value) 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 differ from the work in question.

Dumitriu, Knightly, Kuzmanovic and Stoica [26] presented analytical modelling and simulation studies involving the Gnutella [18] 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.

Christin, Weigend and Chuang [27] conducted a measurement study in content availability for four (Gnutella, eDonkey, Overnet and FastTrack) P2P overlay networks. 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.

Liang, Naoumov and Ross [28] presented a methodology for estimating index poisoning levels and pollution levels in structured and unstructured file sharing networks.

Singh, Ngan, Druschel and Wallach [29] 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 member of 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.* [7] do show that whereas Eclipse attacks are feasible in Kad — a Kademlia-based [22] 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.

Naoumov and Ross [30] 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 DDos attack. The emphasis of our work is not on denial-of-service attacks.

Finally, Steiner, En-Najjary and Biersack [31] indicate how Kad, a Kademlia-base [22] DHT, 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 solution scheme for preventing sybils from gaining access to Kad.

IV. PERFORMANCE MEASURES AND ASSESSMENT

For our simulation studies, we performed discrete event simulations which captured key features of the Overnet P2P overlay network. Our simulations address the following questions:

- 1) What effects do Sybil attacks have on the botnet command and control (C & C) structure when the sybil birth rate (the rate of which sybils are added) is equal to: (a) the net growth rate (the rate at which new bots are added minus the attrition rate), (b) half the net growth rate and (c) twice the net growth rate?
- 2) What effects do time duration of Sybil attacks have on the degree of success in disrupting the botnet communication?
- 3) Do botnet design choices such as the size of the peer-list have any bearing on the effectiveness of the Sybil attacks?

In our simulation, the sybils play the following role: (a) they send PING, FIND_NODE and FIND_VALUE messages to non-sybil nodes in attempt to get their IDs in the peer-list of the nodes; (b) they respond to FIND_NODE and FIND_VALUE queries with false information. They always respond to these queries by sending the IDs of other sybils to the sender of the messages; suggesting that these nodes (the sybils) will likely point the senders to the value they search.

Each simulation run starts with an initial number of 20,000 nodes (bots). The per time-step growth rate of the botnet is assumed to be 2% of the initial 20,000, i.e. 400 nodes at each

time step, with the attrition rate being 1% of the initial number of nodes, i.e. 200 nodes at each time step; thus resulting in a net growth rate of 1% or 200 nodes per time-step. The growth and attrition rates are kept constant for each simulation run, whereas, the sybil birth rate varies from 0.5, 1 and 2 times the net botnet growth rate, i.e. 0.5%, 1%, and 2%. Simulations with the above parameters are compared when the duration of the simulation (Δt) equals 10, 20 and 30 time-steps. Also, simulations with $\Delta t = 20$ and the sybil birth rate as indicated above, are compared when the size of a node's peer-list (l) equals 100, 200 and 300. Each simulation experiment is repeated 20 times and the average of the simulation data computed.

To assess the effectiveness of the Sybil attack in disrupting the botnet C&C infrastructure, we use the reachability measure we introduced in a previous paper [16]. Consider the directed graph $G = (V, E)$, where the vertices in V are the nodes in the botnet, and an edge (u, v) is in E if v is in one of u 's k -buckets, i.e. in its peer-list. Reachability from a given node u is the number of nodes that can be reached within a given shortest-path distance from that node. More precisely, let $\Gamma_r(u)$ denote the set of nodes at distance r from a node u in a graph $G = (V, E)$.

$$\Gamma_r(u) = \{v \in V : d'(u, v) = r\},$$

where $d'(x, y)$ represents the length, i.e., number of hops, of the shortest path between node u and v (not to be confused with the XOR distance between node IDs, introduces in Section II). Let $N_r(u)$ represent the set of nodes at distance at most r from u , i.e. its neighbourhood of radius r , i.e.

$$N_k(x) = \bigcup_{i=0}^k \Gamma_i(x).$$

Then, the r -reachability of a node u is simply the cardinality of its r -radius neighbourhood divided by the total number of nodes, i.e. $|N_r(u)|/|V|$.

In order for bots to get a new version of the code or updated commands, the operator must seed new (key, value) pairs in the P2P network, pointing to the appropriate URL. As per the Kademlia algorithm, if a new pair is seeded at a node x , then x will send STORE messages to the k nodes with IDs closest to the key. This is done by first sending a FIND_NODE with the key, to (a subset of) the k closest nodes to the key, amongst x 's k -buckets. Each of those nodes will then reply with its known k closest nodes, amongst its k -buckets. The seeding node x will now have knowledge of the k closest nodes known to all of its neighbours, which is the same as to say the k closest within $N_2(x)$. In the next iterations, x will send FIND_NODE messages to the known k closest nodes so far, until no closer nodes that k closest known are found. Thus, we have shown that at the i -th step (the 1st step being the internal k bucket lookup), the initiator x knows the closest k nodes to the target ID within its neighbourhood $N_i(x)$.

Similarly, when any given bot y queries the P2P network for one of the 32 Storm search keys for the day, it will initiate a

node search with a FIND_NODE message to the k nodes with closest ID to the searched key within its k -buckets. Again, y will have found the searched key and value after i iterations, if and only if one of the storing nodes lies within its i -radius neighbourhood, i.e. within $N_i(y)$. Note that due to the caching features of Kademlia, the storing nodes for that key do not necessarily include only the initial k nodes to which the x sent STORE messages.

The diameter of the graph (maximum length of any shortest paths) is an upper bound on the number steps it will take for the lookup process to converge and terminate. It is thus directly related to the speed at which new command information is disseminated and at which bots can retrieve it when they decide to look for it. However, if the lookups are time-limited or if the number of iterations are limited to a fixed value (e.g. in order to minimise the number of message footprint and increase stealth), then reachability within small radii becomes a better measure of C&C effectiveness. In general, thus, having large neighbourhoods with a small radius r for a large fraction of "centre" nodes x is advantageous for botnet operators, because this will guarantee that that with high probability the k nodes chosen for storage will be a) the k nodes with closest ID to the search key and b) that other bot nodes can obtain the correct values from them.

In the context of Sybil attacks, however, since the sybil nodes cannot be counted upon to reliably transmit information on node queries, nor to store values, one must measure effectiveness without taking them in consideration. In other words, one must only consider the reachability within the subgraph $G' \subset G$ containing only non-sybil nodes.

In our simulations, we utilised the igraph C library [32] to construct and handle graph objects representing the Botnet C&C infrastructure. Home-made coded using igraph was used for all aspects of the simulation: generating the initial graphs, simulating the birth and death process for the regular bots and the sybils, and finally for measuring non-sybil node reachability. To do this, the sybil nodes are removed from the graph and reachability is calculated within the resulting subgraphs.

V. SIMULATION RESULTS

Fig. 1 shows the reachability histogram (for $r = 1$ radius) for the simulation when the number of time-steps (Δt) is 20 and the peer-list size (l) is fixed to 200; and the sybil birth rate (S_{BR}) varies from 0 to 2 times the net botnet growth rate (B_{GR}). As indicated in Section IV, the net botnet growth rate is assumed to be 1% of the initial 20,000 nodes, i.e. 200 nodes per time-step. Therefore, at the end of the 20 time-steps, the total number of nodes (excluding the sybils) is 24,000. When $S_{BR} = 0.5B_{GR}$, a total of 2,000 sybils are generated during the 20 time-steps. The insertion ratio of sybils in the peer-lists (I_R) is the total occurrences of sybils (S_I) in the peer-lists divided by the product of the final number of nodes (N) and the peer-list size (l), i.e., $I_R = \frac{S_I}{(N \cdot l)}$. The average insertion ratio for the 20 simulation runs is 7.88%, with the standard deviation being 0.6078%. When $S_{BR} = B_{GR}$, 4,000

sybils are generated and the average insertion ratio is 14.34% and the standard deviation 0.6668%. When $S_{BR} = 2B_{GR}$, 8,000 sybils are generated; the mean insertion ratio for the 20 simulation runs is 24.82% and the standard deviation 1.0678%.

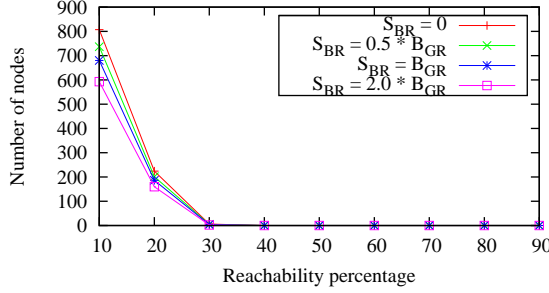


Fig. 1. Reachability histogram for $\Delta t = 20$ and $l = 200$

Fig. 1 also indicates that the number of nodes whose radius 1 neighbourhood include 10% of the nodes in the botnet, decreases by 27% when $S_{BR} = 2B_{GR}$ compared to the case where $S_{BR} = 0$. Whereas, when $S_{BR} = B_{GR}$ and $S_{BR} = 0.5S_{BR}$ the decrease in the reachability is 16% and 9%, respectively. This suggests that there is an inverse relationship between S_{BR} and the reachability. Therefore, intuitively, the potency of a sybil attack will likely increase as S_{BR} increases.

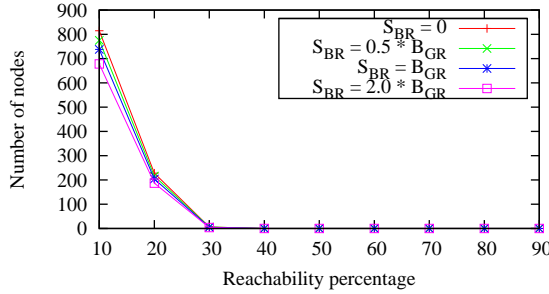


Fig. 2. Reachability histogram for $\Delta t = 10$ and $l = 200$

Fig. 2 shows the reachability histogram (for $r = 1$ radius) for the simulation when the number of time-steps (Δt) is 10 and the peer-list size (l) is 200; and as in the previous figure, S_{BR} varies from 0 to $2B_{GR}$. The number of nodes (N) at the end of the 10 time-steps is 22,000. When $S_{BR} = 0.5B_{GR}$, 1,000 sybils are generated, the average insertion ratio for the 20 simulation runs is 4.22%, the standard deviation being 0.5123%. When $S_{BR} = B_{GR}$, twice as many sybils are generated and the insertion ratio is 8.34%, the standard deviation being 0.5293%. Whereas, when $S_{BR} = 2B_{GR}$, 4,000 sybils are generated, the average insertion ratio is 15.43% and the standard deviation is 0.8730%. The plot shows that when $S_{BR} = 2B_{GR}$, the number of nodes that can reach 10 percent of the nodes in the botnet within radius 1 decreases by 17% compared to when $S_{BR} = 0$; whereas, when $S_{BR} = B_{GR}$ and

$S_{BR} = 0.5B_{GR}$, where the decrease is 10% and 5% percent, respectively.

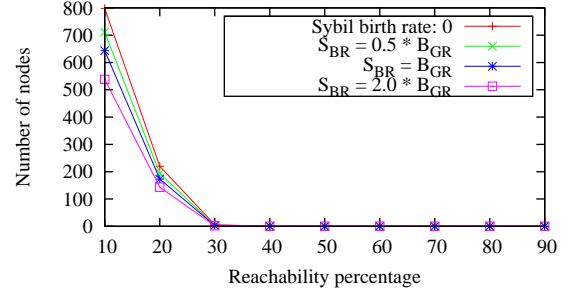


Fig. 3. Reachability histogram for $\Delta t = 30$ and $l = 200$

Fig. 3 shows the reachability histogram (for $r = 1$ radius) for the simulation when $\Delta t = 30$ and $l = 200$; and S_{BR} varies between 0 and $2B_{GR}$. At the end of the 30 time-steps, $N=26,000$, and the mean insertion rate for the 3,000, 6,000 and 12,000 sybils generated when S_{BR} equals $0.5B_{GR}$, B_{GR} and $2B_{GR}$, respectively, are 10.53%, 18.67% and 30.94%; and the standard deviation 0.5422%, 0.6922% and 1.2172%, respectively. The plot shows that when $S_{BR} = 2B_{GR}$, the number of nodes that can reach 10% of the nodes within radius 1 decreases by 37% compared to the case where $S_{BR} = 0$, whereas, when S_{BR} equals B_{GR} and $0.5B_{GR}$, the decrease is 20 percent and 11 percent, respectively. Comparison of Figures 1, 2 and 3 indicates that when the duration of the Sybil attack increases by 3 folds, the effectiveness of the attack in disrupting the communication channel increases by approximately 2 folds.

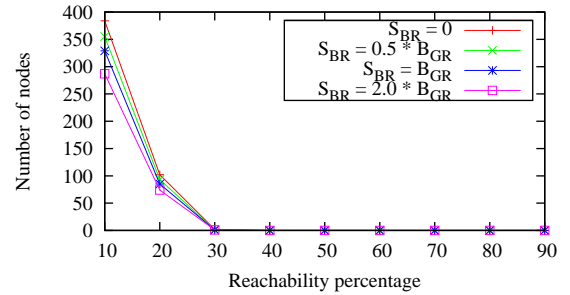


Fig. 4. Reachability histogram for $\Delta t = 20$ and $l = 100$

Fig. 4 shows the reachability histogram when Δt is 20, l is 100 and S_{BR} varies between 0 and $2B_{GR}$. At the end of the 20 time-steps $N=24,000$, and the average insertion ratio for the 2,000, 4,000 and 8,000 sybils generated when S_{BR} equals $0.5B_{GR}$, B_{GR} and $2B_{GR}$, respectively, are 7.62%, 13.94% and 24.74%, while the standard deviation are 0.8577%, 1.2987% and 1.6265%, respectively. The plot shows that when $S_{BR} = 2B_{GR}$, the number of nodes having 10% of the nodes within their radius 1 neighbourhood decreases by 25% compared to the case when $S_{BR} = 0$. Whereas, when

S_{BR} equals B_{GR} and $0.5B_{GR}$, the decrease in reachability are 14% and 8%, respectively.

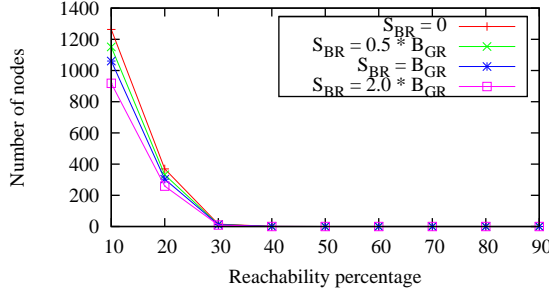


Fig. 5. Reachability histogram for $\Delta t = 20$ and $l = 300$

Fig. 5 shows the reachability histogram where $l = 300$ and Δt , S_{BR} and N are the same as for Fig. 4. The average insertion rates for the sybils are 7.88%, 14.35% and 24.83% when S_{BR} equals $0.4B_{GR}$, B_{GR} and $2B_{GR}$, respectively; and the standard deviation 0.6050%, 0.9602% and 0.7827%. The plot indicates that the number of nodes with 10% reachability for radius $r = 1$ decreases by 27% compared to the case where $B_{GR} = 0$. Whereas, the reachability drops by is 16% and 9% when S_{BR} equals B_{GR} and $0.5B_{GR}$, respectively. Figures 1, 4 and 5 indicate that there are no significant difference in the reachability measures when l changes from 100 to 300. This suggests that the size of the bots' peer-list does not have any bearing on the effectiveness of this kind of Sybil attack.

VI. DISCUSSION

S_{BR}/B_{GR} $l = 100$	Reachability Decrease (%)	No. Sybils in Botnet (%)	Diff. (%)
0.5	8.0	8.33	0.33
1.0	14.0	16.67	2.67
2.0	25.0	33.33	8.33

TABLE I
IMPACT OF SYBIL ATTACK ON REACHABILITY $\Delta t = 10$.

S_{BR}/B_{GR} $l = 100$	Reachability Decrease (%)	No. Sybils in Botnet (%)	Diff. (%)
0.5	5.0	4.54	-0.46
1.0	10.0	9.09	-0.91
2.0	17.0	18.18	1.18
S_{BR}/B_{GR} $l = 200$	Reachability Decrease (%)	No. Sybils in Botnet (%)	Diff. (%)
0.5	9.0	8.33	-0.67
1.0	16.0	16.67	0.67
2.0	27.0	33.33	6.33
S_{BR}/B_{GR} $l = 300$	Reachability Decrease (%)	No. Sybils in Botnet (%)	Diff. (%)
0.5	11.0	11.5	0.05
1.0	20.0	23.07	3.07
2.0	37.0	46.15	9.15

TABLE II
IMPACT OF SYBIL ATTACK ON REACHABILITY $\Delta t = 20$.

S_{BR}/B_{GR} $l = 300$	Reachability Decrease (%)	No. Sybils in Botnet (%)	Diff. (%)
0.5	9.0	8.33	-0.67
1.0	16.0	16.67	0.67
2.0	27.0	33.33	6.33

TABLE III
IMPACT OF SYBIL ATTACK ON REACHABILITY $\Delta t = 30$.

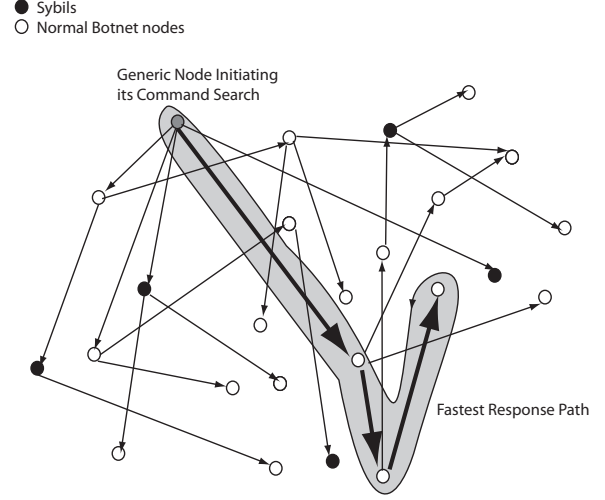


Fig. 6. The effectiveness of sybil attacks is innately constrained by the fastest responding paths.

Tables I-III summarises the results obtain in Section IV for peer list sizes $l \in \{100, 200, 300\}$ and $\Delta t \in \{10, 20, 30\}$. From these tables it is apparent that for low sybil attack birth rates, (i.e., $S_{BR}/B_{GR} = 0.5$), computing the percentage of sybils within the botnet provides a fairly accurate indication of the expected impact on the $r = 1$ reachability measure across all Δt 's and all peer list sizes. As the sybil attack birth rate increases to $S_{BR}/B_{GR} = 2.0$ then this coarse estimate of the expected effect begins to overestimate, with the worst estimate error occurring for $\Delta t = 20$ and $l = 300$.

The explanation for this result arises from how the Storm botnet and, more precisely, how Overnet-based peer-to-peer networks are structured with respect to the core command search operation. For any given node initiating a search activity, the search will terminate once the node receives the first correct response to the search from one of the k (or more) storing nodes with closest node ID to the searched key. Hence, all sybils located further away from the closest storing node, in a response time sense, will have zero impact on the botnet's ability to propagate commands, with respect to that given node's query as per Fig. 6. On average, therefore, the impact of sybil attacks is constrained by the likelihood that sybils will be positioned such that they are on average closer than the closest responding non-sybil storing node. This implies that as the sybil injection ratio grows, more of the botnet becomes sybils within a given Δt , but these additions will be unlikely to affect this critical path since most sybils will be injected beyond, in a response time sense, the fastest responding non-sybil storing

node. Note that this whole discussion depends heavily on whether nodes will cache $\langle \text{key}, \text{value} \rangle$ pairs information, even if their ID is not necessarily close to the key (something that some Kademlia implementations do).

The Sybil attack process, of course, could be improved through incorporating knowledge of estimated path response times in order to take advantage and win such race conditions. But such knowledge would need to be continually updated to reflect the network dynamics, as introduced through birth and death processes, and would, in the general case, require near global knowledge of the botnet, given its random nature. If such knowledge were available then the botnet could be disabled directly without needing to resort to a Sybil attack. Additionally, it would be fairly trivial to incorporate fault tolerance into the botnet structure, *i.e.*, though voting, to reduce the effectiveness of the modelled uninformed sybil attacks, thereby introducing an added requirement that sybils exist on majority of responding paths leading to storing nodes, where the number of paths is set through the voting scheme (similarly as the α concurrency parameter defined in Kademlia). Such an approach would also most likely require that the botnet commands themselves be injected at multiple points within the botnet. This may or may not be desirable from the botnet operator perspective due, primarily, to the associated increased risks of detection and attribution.

VII. CONCLUSIONS AND FUTURE WORK

This work has presented the initial results of a simulation study of the effects of Sybil attacks on botnet command and control infrastructures which utilise Overnet-based peer-to-peer network solutions. Overall, these results indicate that substantial and sustained Sybil attacks are required to cause significant degradations in such botnets. The core constraint on the effectiveness of Sybil attacks has been shown to be the probability that injected sybils will be, on average, on the first responding path to nodes storing the answers to the search queries. Hence, uninformed sybil attacks result in the introduction of significant numbers of sybil nodes which have near-zero impact on the botnet's operation. In particular, for $l = 300$, $\Delta t = 20$, and $S_{BR}/G_{BR} = 2.0$ sybils approach almost half of the nodes in the botnet but only produced a slightly larger than 1/3rd impact on the radius 1 reachability, with all reachability measures past radius 1 remaining unaffected at 100%. The random nature of Overnet-based peer-to-peer networks provides fairly strong protection against uninformed sybil attacks, while sybils remain in the minority of botnet nodes. Informed sybil attacks could potentially be more effective but are more impractical due to the requirement for near global information regarding path response times; even then such informed sybil attacks could likely be mitigated by the introduction of fault tolerant approaches, such as voting, within Overnet's structure.

These results suggest that if sybil attacks are to be used in practice to disable Overnet structured peer-to-peer botnets then it would be prudent to explore Quality of Service approaches, in conjunction with service providers, to ensure that the

responses from injected sybils have a high probability of existing on the fastest response paths to search queries. In this manner, significant reductions in the numbers of sybils required should be achievable. Fault tolerant voting schemes could, of course, be used to counter such approaches, though at the cost of both trading-off the responsiveness of the botnet and reducing its stealthiness through the requisite increase in generated network traffic. Additionally, the need to have sybils become dominant within the botnet in order to disable it is fairly clear and, hence, must exceed the background birth/death rates. Not explored in this work is the question of whether such high sustained sybil injection rates themselves become, by their very nature, easily detectable by the botnet operators, in which case the botnet could be shutdown and moved, negating the effect of the deployed sybils.

Exploring such questions innately requires richer simulation studies of peer-to-peer botnet behaviours and, more particularly, simulation studies which take into account actual packet-level network behaviours, inclusive of reasonable background traffic models, and at the scales at which botnets known to exist in the real-world. This is an area which the authors are actively pursuing through the use of finer-grained discrete event simulation environment (*i.e.* OMNet). Of course, the result of this work only pertains to those aspects of the botnet which are effected by the reachability measure, other unmeasured aspects of botnet performance may show less resilience to sybil attacks. As the network-level at-scale simulation environment comes on-line it will be used to more thoroughly explores such issues.

Lastly, the graphs simulated in this work model a peer-list establishment process in which a node is equally likely to be attached to any other in the graph. In fact, this is only a first-order approximation of how peer-lists and k -buckets will be populated according to the Kademlia algorithm. One of the effects of the use of the XOR distance is that nodes that are closer in ID are more likely to be in each other's peer-list, and hence the above simplification is not adequate. It remains to be verified what the effects of this approximation are on reachability and other robustness and resilience measures. Comparing the results obtained here with those obtained on graphs with node attachment models closer to Kademlia is also the object of current research.

REFERENCES

- [1] P.-M. Bureau and A. Lee, "Malware storms: a global climate change," Virus Bulletin www.virusbtn.com, November 2007.
- [2] D. Fisher, "Storm, nugache lead dangerous new botnet barrage," Search-Security.com, December 2007.
- [3] J. Stewart, "Storm worm DDoS attack," <http://www.secureworks.com/research/threats/storm-worm>, February 2007.
- [4] "Expert: Botnets no. 1 emerging Internet threat," CNN Technology news, www.cnn.com/2006/TECH/internet/01/31/furst/, January 2006.
- [5] "The botnet trackers," Washington Post Technology news, www.washingtonpost.com/wp-dyn/content/article/2006/02/16/AR2006021601388.html, February 2006.
- [6] "Attack of the zombie computers is growing threat," The New York Times Technology news, nytimes.com, January 2007.

- [7] 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 1st USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET'08)*, April 2008.
- [8] C. Kanich, K. Levchenko, B. Enright, G. Voelker, and S. Savage, "The heisenbot uncertainty problem: Challenges in separating bots from chaff," in *Proceedings of the 1st USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET'08)*, April 2008.
- [9] CISCO and IronPort, "2008 internet malware trends: Storm and the future of social engineering," www.ironport.com/malwaretrends, 2008.
- [10] MessageLabs, "Storm significantly weakens & web-based malware on the rise," MessageLabs Intelligence: www.messagelabs.com/mlireport/MLI_Report_April_2008.pdf, April 2008.
- [11] G. Keizer, "Microsoft didn't crush storm, counter researchers," <http://www.computerworld.com>, April 2008.
- [12] E. Cooke, F. Jahanian, and D. McPherson, "The zombie roundup: Understanding, detecting, and disrupting botnets," in *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet SRUT'05*, July 2005.
- [13] P. Wang, S. Sparks, and C. C. Zou, "An advanced hybrid peer-to-peer botnet," in *Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets (HotBots 2007)*, April 2007.
- [14] R. Vogt, J. Aycok, and J. M. J. Jacobson, "Army of botnets," in *Proceedings of the 14th annual Network and Distributed System Security Symposium (NDSS 2007)*, March 2007.
- [15] G. Keizer, "Microsoft: We took out storm botnet," <http://www.computerworld.com>, April 2008.
- [16] 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 *To appear in the 13th European Symposium on Research in Computer Security (ESORICS'08)*, 2008.
- [17] A. L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509–512, 1999.
- [18] "Gnutella," <http://www.gnutella.com>, March 2001.
- [19] P. Erdős and A. Rényi, "On random graphs I," *Publ. Math.*, vol. 15, pp. 290–297, 1959.
- [20] K. Kutznet and T. Fuhrmann, "Measuring large overlay networks the overnet example," in *KiVS*, May 2006.
- [21] J. Douceur, "The sybil attack," in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems*, March 2002, pp. 251–260.
- [22] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the XOR metric," in *Revised Papers from the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, March 2002.
- [23] F. Boldewin, "Peacomm.c - cracking the nutshell," <http://www.reconstructor.org>, September 2007.
- [24] P. Porras, H. Saidi, and V. Yegneswaran, "A multi-perspective analysis of the storm (peacomm) worm," Technical report, Computer Science Laboratory, SRI International, October 2007.
- [25] "Know your enemy: Fast-flux service networks," HoneyNet Project, www.honeynet.org/papers/honeynet, July 2007.
- [26] 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*, vol. 33, pp. 38–49, June 2005.
- [27] N. Christin, A. Weigend, and J. Chuang, "Content availability, pollution and poisoning in file sharing peer-to-peer networks," in *Proceedings of the 6th ACM conference on Electronic commerce*, June 2005, pp. 68–77.
- [28] J. Liang, N. Naoumov, and K. Ross, "The index poisoning attack in p2p file sharing systems," in *Proceedings of 25th IEEE International Conference on Computer Communications (INFOCOM 2006)*, April 2006.
- [29] A. Singh, T.-W. Ngan, P. Druschel, and D. Wallach, "Eclipse attacks on overlay networks: Threats and defenses," in *Proceedings of 25th IEEE International Conference on Computer Communications (INFOCOM 2006)*, April 2006, pp. 1–12.
- [30] N. Naoumov and K. Ross, "Exploiting p2p systems for DDoS attacks," in *Proceedings of the 1st international conference on Scalable information systems (INFOSCALE 2006)*, May 2006.
- [31] M. Steiner, T. En-Najjary, and E. Biersack, "Exploiting kad: possible uses and misuses," *ACM SIGCOMM Computer Communication Review*, vol. 37, pp. 65–70, October 2007.
- [32] G. Csárdi, "The igraph library," <http://cneurocv.s.rmk.kfki.hu/igraph>, 2005.