

Locating Zombie Nodes and Botmasters in Decentralized Peer-to-Peer Botnets

*Christ Nunnery and Brent ByungHoon Kang
College of Computing and Informatics
University of North Carolina at Charlotte*

1 Introduction: Peer-to-Peer Botnets

With the pervasive spread and growing presence of malicious botnets throughout the Internet, efforts to mitigate and provide countermeasures to these threats to security and resources are all the while increasing in intensity. As a result, botnets are undergoing a sort of evolutionary processes, wherein their topologies are modified to defend against incapacitation. One such example of this process is implementation of decentralized botnets with peer-to-peer architecture. This decentralized architecture marks a strong departure from traditional IRC based botnets that necessitate a central server to route traffic, consisting of botmaster commands and bot replies, throughout the botnet.

Decentralized peer-to-peer networks operate without a centralized server. Peers join the network using a method known as *bootstrapping*, where an IP address of a node already on the network is required. The joining peer contacts this node and receives information about other nodes on the network. Traffic, such as searches and file transfers, is routed through neighboring nodes on the network. Efficiency suffers to an extent, as nodes must assist in the transmission of the traffic of other nodes in the network, particularly with executing searches. Provided in a decentralized peer-to-peer (P2P) botnet, however, is the absence of a single central server. The use of a single centralized server in a botnet is a critical vulnerability which IRC based systems inherently possess.

If the central server in an IRC-based botnet is for any reason incapacitated, the entire botnet ceases to function. In decentralized peer-to-peer botnets, the only “servers” present are the peers themselves. In both examples, “servers” refer to nodes that perform traffic routing and file indexing on the network. A large number of nodes on the network would need to be disabled to incapacitate the botnet. Without a central

weak-point in the botnet topology, researches are currently attempting to discover not only how peer-to-peer botnets operate, but more critically, how they may be detected, mitigated, and possibly disabled.

2 Background

2.1 Overnet and Kademia

An example of a decentralized peer network that we believe shows potential for bonet use is Overnet, which makes use of the Kademia algorithm. Kademia uses a distributed hash table, or DHT, design in which there is no hierarchy in the topology. In Kademia, peers are used on the network to route searches from other peers and cache hash records of owned content. Essentially, the burden of file indexing is distributed throughout the network [1]. After bootstrapping onto the network, nodes generate a unique 128-bit ID [2]. They then publish information about files they own to nodes in the network that are close in the keyspace to the hash identifiers of these published file or files (“closeness” here is determined by use of the XOR operation. The XOR metric is implemented to calculate “distances” between hashes in the network). The information published consists of the file identifier hash, also known as the “key”, and a “value,” consisting of the peer’s ID, IP address and port number [2]. Each node which receives a publish packet adds the record to their own hash table. The joining client will also generate and publish records for keywords in the metadata of files. For these records, the hash of the keyword along with a version identifier is published [2]. An overlay network topology is constructed based on these actions as nodes gain awareness of other nodes. Consequently, searches in the network are possible.

For searching, Overnet peers generate hashes based on keywords and query progressively

closer nodes in the keyspace for these hashes. This is accomplished by performing a XOR operation with a search hash and the newly reached node hash for each iteration of the search. Searches continue routing to increasingly closer peers until a peer with a record for the search hash is found. This peer then responds with version identifiers for all the corresponding records it possesses in its table. The searching node could then select a version to download. To find the location of the file, the searching node would query the DHT network again, this time using the version identifier as the search key [2].

The network is surprisingly efficient despite the absence of a central server, and bandwidth between nodes is conserved. It is also quite resilient to lost peers, easily patching over “holes” in the hash tables. Note that physical proximity is not taken into account with this topology. As discussed, “close” in this instance is not referring to physical proximity, rather, the nearness of values from the XOR operation. This may result in some data transfers occurring over long distances between what, on the network, are close peer hashes, impacting performance. Regardless, we believe P2P based botnets will show a proclivity for using Kademlia, as it is a masterfully designed protocol capable of meeting the needs of the botnet with impressive resilience and concealment. Few P2P bots are known to exist as of yet, but we expect to see a surge in prominence as botmasters become frustrated with the unreliability of IRC based systems.

2.2 The Peacomm P2P Bot

Analysis has been conducted on a particular peer-to-peer bot known as Peacomm or Storm Worm [3]. Peacomm, originating in January of 2007, is one of very few known P2P bots. Peacomm is quite useful as it demonstrates a real-world implementation of a P2P botnet with techniques and methodologies for a successful deployment and full botnet functionality. Peacomm makes use of the Overnet network to locate secondary injection URLs. Peacomm only uses the first stage of searching the Overnet network and does not need to obtain the secondary injection URLs. Once the search

results are obtained, there is no further P2P traffic. The secondary injection URLs are stored encrypted with RSA [4] in a meta tag called “id” in the search result which the searching bots can decrypt using a key hard-coded in the binary, along with the second hash received in the search reply [5]. Typically in Overnet, this hash is a version identifier. It appears that innocuous peers not running the Peacomm binary are able to provide the botnet with these search results, as they obtained the records the bots are searching for simply by being near the keyspace of the search hash. The creator of this malware has designed Peacomm to work with legitimate nodes in the network engaging in non-malicious file sharing by publishing non-traditional hashes on the network that conform to the protocol specifications to a degree which is acceptable. We predict that other P2P botnets using Overnet will choose a similar strategy.

The search hashes Peacomm uses to find secondary injection URLs are not static. They are generated based on a random integer between 0 and 31 and the current system time (day) [5]. The botmaster, accordingly, would be required to publish new hashes to accommodate the changing hashes the botnet “zombie” nodes look for. This may seem unnecessary, but using rotating hashes in this type of network serves two purposes. Firstly, the botmaster may update what URLs are being used in the event of a DNS or hosting failure or simply to modify behavior. Secondly, this technique defends against hash poisoning. Hash poisoning in a DHT would involve determining what hashes are used in Peacomm for a given time frame and publishing records for these hashes with otherwise incorrect information, such as invalid meta data or version identifiers [2]. We will refer to these active hashes for a time period, respective of the time input in the hash generation algorithm, as “pertinent hashes” for the malware.

3 Our Approach:

3.1 Locating Zombie Nodes on a Peer-to-Peer DHT Network

Zombie nodes on the network, those which are compromised, could be identified by their retrieval of hashes known to be for botnet use. To determine which search hashes are pertinent, the Peacomm bot could either be actively running on a network without a true Internet connection to determine current hashes, or the hash generation algorithm could be extracted from the Peacomm binary to generate hash sets on the fly based on the limited set of random integers and the current time.

With the knowledge of pertinent hashes and the control of a large group of networked computers, botnet activity is identifiable. The theoretically collision free hashes which infected clients would search for would not arise in legitimate Overnet P2P usage. Thus, if a client within this controlled network *searches* (with Overnet type 0x14) for a pertinent hash used by the malware, it must be a so-called “zombie” node. These nodes could be blocked at their ISP, or blacklisted from other networks. By implementing numerous “vantage points” in large networks searching for this activity, we can successfully monitor this activity in inbound and outbound traffic and severely limit P2P bot activity within the network.

3.2 Locating the Botmaster

Similarly, if we discover the presence of any *publish* activity in Overnet to or from any nodes on the supervised network with known bot hashes, we can determine the IP address(es) of the computer or computers owned by the botmaster used to initially distribute hashes from these publish packets. For this, extraction of the hash generation algorithm in our Peacomm sample would be necessary, as it is likely the botmaster would publish hashes in advance, prior to the start of an active-hash time window. This would be necessary to prevent occurrences where the bots are searching for hashes that have not been fully distributed in the network. As botmasters are surely concerned with the

performance of their networks, this is highly likely. Accordingly, observing the network for hash occurrences retrieved from a running version of Peacomm would be ineffective, as the hash-publish event from the botmaster should occur *before* the monitored execution of the binary changes hashes, when we would be aware of a new hash. If we see *publish* events (suspected Overnet type 0x13) on the network for these bot hashes, we can log the IP addresses from the originating peer and notify the ISP or other networks to either disable the machine or blacklist the IP.

Success of the suggested P2P traffic discovery methods in this proposal depends on the ability to monitor large groups of computers from vantage points where we have access to both inbound and outbound traffic for all nodes within the network(s). This monitoring would be most effective if it could occur at the level of an Internet Service Provider or at the gateways, routers, or servers of organizations. Logically, with a small quantity of global P2P botnet traffic, additional monitored nodes increase our chances of discovering and mitigating these types of traffic which is otherwise difficult to detect. Further, our methods for finding zombie peers and hash upload originators would be equally effective for other Overnet based P2P botnets, provided they implement hash-based searching and publishing for botnet functionality.

In Figure 1, an example of the detection topology with sample bot activity is shown. There are two malicious nodes, a *zombie* on ISP 1 and a *botmaster* on ISP 4. The grey arrows show traffic from the botmaster **publishing** a hash known to be used in a botnet via analysis of a binary or the extraction of a hash generation algorithm. Vantage point D records this traffic directly from the botmaster. This would be a rare occurrence. Vantage points B and C also record this traffic, but as it arrives at the peers (innocuous, chosen only for their XOR closeness to the file hashes) who will store the records in their hash tables.

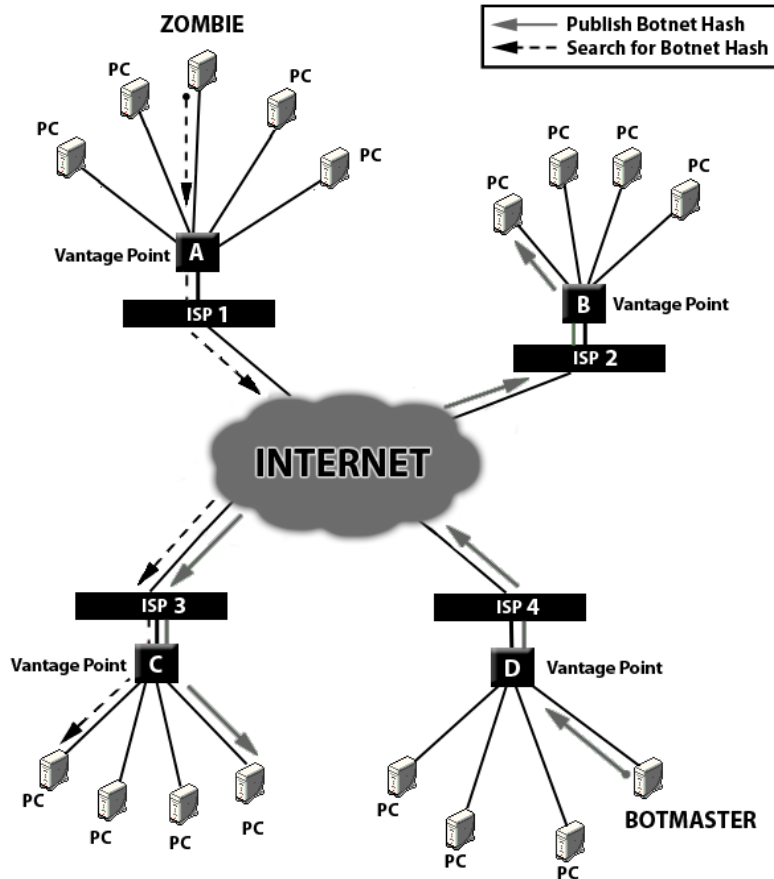


Figure 1: Detection Topology: Vantage points exist in the network at four Internet service providers. Each is capable of detecting P2P botnet traffic, e.g. instances of searching and publishing hashes known to be used by botnets.

The dashed arrows show traffic from a zombie node sending a **search** request for a hash also known to be used by a P2P bot. Vantage point A records this outbound traffic from the zombie peer itself. Vantage point C also is able to capture this data, as the search happens to query a computer in ISP 3. Our diagram implies that this computer in ISP 3 contains records for the hash the zombie node in ISP 1 is searching for. Note that the traffic from both malicious nodes in our diagram is not necessarily related in any particular sequence. This diagram, a subset of the Internet, merely portrays where vantage (observation) points could be effective given the four possible types of traffic related to a P2P botnet using Overnet. These types include: inbound botmaster hash-publish, outbound

botmaster hash-publish, inbound bot hash-search and outbound bot hash-search.

Finally, we would like to invite any interested parties to subscribe to our list of known bot hashes should they wish to block them on their own domain.

4 Related Work

Petar Maymounkov and David Mazieres proposed in [1] a peer-to-peer network using distributed hash tables known as Kademlia. This paper offers an explanation of the intricate behaviors of the system, including the use of the XOR metric, its implementation of hashes, publishing, and searching. Kademlia appears to offer all the technical needs a peer-to-peer

botnet would require in terms of protocol design. As demonstrated by its use in Peacomm, we believe other P2P botnets will incorporate the DHT design Kademlia provides.

Jian Liang et. al. in [2] delineate the process of hash poisoning P2P networks which use hashes for file indexes. This technique would be useful, but from a peer-only perspective. As shown in this proposal, with access to a monitoring scheme for a large group or group of computers where inbound and outbound traffic may be observed, further defense methods are possible for these botnets.

Grizzard et. al. outlined the history of bots, both malicious and non-malicious, peer-to-peer technologies, and the ultimate culmination of these two topics, peer-to-peer based botnets in [3]. A case study of a particular peer-to-peer bot, Peacomm, also known as Storm Worm is also included, to provide the reader with an understanding of how peer-to-peer botnet strategies may be implemented in the real world.

Elia Florino and Mircea Cibotariu also performed analysis of the Peacomm bot in [4], but have provide very detailed information about the encryption schemes the malware uses for URL cryptography. Particularly, they provide the private key the malware uses which appears to be constant among different variants of Peacomm.

Joe Stewart of SecureWorks provided the first in-depth analysis of Peacomm in [5], which includes detailed information about its hard-coded peer lists and the hash generation algorithm Peacomm employs. This research also provides information as to what the bot, at the time of writing, was used for: distributed denial of service attacks and email spamming. We draw from this that though the architecture of the botnet is drastically different from that seen in IRC based systems, P2P botnets are capable of delivering functionality which is quite familiar.

The potential threat posed by bots using peer-to-peer protocols for their command and control

(C&C) was discussed in [6]. Cooke et al. identified some of the foundational analysis techniques for handling botnets including incapacitation of the botnet itself, monitoring the C&C channels, and tracking the propagation and attack mechanisms. This work highlights the underlying difficulties in monitoring the channel that may lead back to the bot controller [6] but some of the techniques may complement the vantage point we discussed. The key challenge in detecting the bot controller in a peer-to-peer network is more difficult due to the dynamic and distributed design of the architecture. Our work focuses on the communication after infection, which can also be established via peer-to-peer networks.

- [1] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the XOR metric," in 1st International Workshop on Peer-to-Peer Systems, pp. 53-62, March 2002.
- [2] J. Liang, N. Naoumov, and K. W. Ross, "The index poisoning attack in P2P file-sharing systems," in Infocom 2006, 2006.
- [3] J.Grizzard, V.Sharma, C. Nunnery, B. Kang, D. Dagon, "Peer-to-Peer Botnets: Overview and Case Study," in Usenix Hotbots 2007, April 2007.
- [4] E.Florino, M. Cibotariu, "Peerbot: Catch me if you can," Symantec Security Response: Ireland, Virus Bulletin, March 2007
- [5] J. Stewart, "Storm worm DDoS attack." <http://www.secureworks.com/research/threats/view.html?threat=storm-worm>, February 2007.
- [6] E. Cooke, F. Jahanian, and D. McPherson, "The zombie roundup: Understanding, detecting, and disrupting botnets," in Usenix Workshop on Steps to Reducing Unwanted Traffic on the Internet, pp. 39-44, USENIX, July 2005