Peer-to-Peer Botnet Investigation: A Review

Mark Scanlon and Tahar Kechadi

School of Computer Science and Informatics,
University College Dublin,
Belfield, Dublin 4, Ireland.
{mark.scanlon, tahar.kechadi}@ucd.ie

Abstract. Botnets have become the tool of choice to conduct a number of online attacks, e.g., distributed denial of service (DDoS), malware distribution, email spamming, phishing, advertisement click fraud, brute-force password attacks, etc. Criminals involved in conducting their craft online all share one common goal; not to get caught. Botnet design, as a result, has moved away from the traditional, more traceable and easily blocked client/server paradigm towards a decentralized Peer-to-Peer (P2P) based communication system. P2P Internet communication technologies lend themselves well to be used in the world of botnet propagation and control due to the level of anonymity they award to the botmaster. For the cybercrime investigator, identifying the perpetrator of these P2P controlled crimes has become significantly more difficult. This paper outlines the state-of-the-art in P2P botnet investigation.

1 Introduction

In the past, cyberattackers required high-end computer equipment coupled with high bandwidth Internet connections to accomplish their goals. In recent years, high bandwidth home and workplace broadband Internet connections have become commonplace. This has resulted in these computers being targeted by criminals to attempting to create large, global distributed systems, i.e., botnets, to perform their bidding. The software robots, or bots, which form these distributed systems are controlled remotely by the criminal attacker, or botmaster.

Investigation of botnets branches into three main areas [1]:

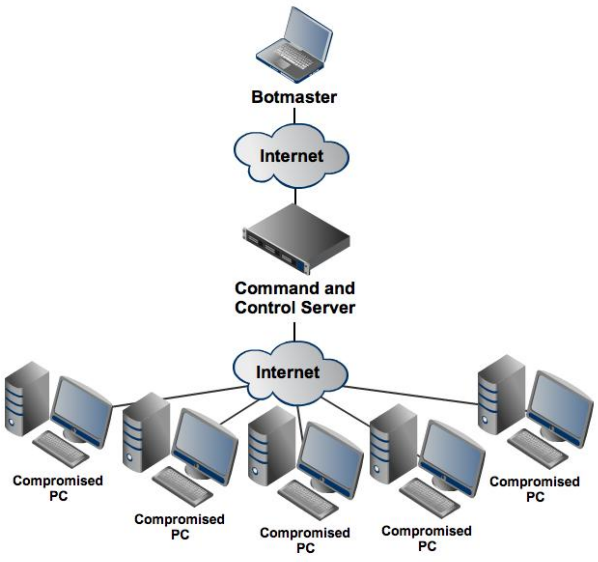
1. Botnet Anatomy - Investigating the anatomy of a particular botnet includes both analysis of the reverse engineering of the binary content and analysis of the network communication behaviours.
2. Wide-area Measurement - This concentrates on attempting to enumerate the population of the botnet, the bandwidth and computational overhead and their usage. Gathering the population of a botnet is a non-trivial task as the number of nodes ever connecting to a Command and Control (C&C) server may only count for a small proportion of the infected nodes [2].

3. Botnet Modelling and Prediction - This includes the theoretical modelling of future botnet designs, along with attempting to design best practice in countermeasures against them.

1.1 Traditional Client/Server Botnets

Traditional botnet design was centred on a client/server paradigm, as can be seen in Figure 1 below. Using this model, the botmaster issues requests to the "Command and Control" (C&C) server. The client-side bot software, which runs on the infected nodes, is pre-programmed to frequently "check-in" with the C&C server in order to get its latest commands. The C&C server eliminates the need for the botmaster's computer to remain online in order to distribute the latest orders to the entire botnet, while awarding the botmaster an added level of anonymity. The C&C server is generally either IRC or HTTP based serving commands to the nodes.

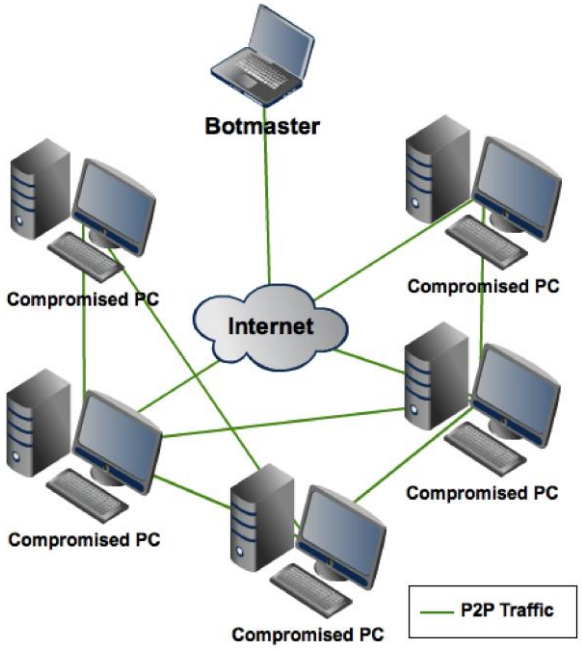
Fig. 1. Typical Client/Server Botnet Command and Control Topology.



The main issue with the client/server model is that it leaves the botnet vulnerable to a single point of failure. To counteract this, multiple C&C servers may be used optionally in conjunction with a dynamic DNS service, such as DynDNS [3] or No IP [4]. The dynamic hostnames are hard-coded into the bot software, quickly and easily enabling the botmaster to swap in a new command and control server by updating the IP addresses associated with the dynamic DNS provider, without any disruption of the botnet's operation.

1.2 Peer-to-Peer Botnets

Fig. 2. Typical Peer-to-Peer Botnet Command and Control Topology



In the P2P botnet topology, each peer, in effect, acts as both the client and the server. When a peer receives a command, that command is executed and the command itself is then forwarded on to any other peers it is aware of, as can be seen in Figure 2. Using this model, the botmaster can briefly connect to the network, issue a command to a single infected node, and immediately disconnect in the knowledge that the command will propagate across the network to each compromised node. As new nodes come online, they bootstrap onto a distributed hash table (DHT). This newly connected node will then receive the latest command from another node in the DHT. Each node in the P2P botnet maintains the DHT, similar to the methods employed in the maintenance of active nodes in the BitTorrent DHT [5]. By design, each bot has a limited view of the entire network imposed by a maximum number of addressed stored or because of a limitation of the network due to firewalls or NAT [6].

The intra-bot communication is generally conducted on a private, closed sourced P2P networks though some use existing public P2P networks. Commonly, P2P botnets also take advantage of the popularity of P2P file-sharing services to help to spread copies of the bot to new machines [7]. By posing as a desirable file, e.g., “Microsoft Office.exe”, on these file-sharing networks, users will, in effect, infect their own machine with the malware while attempting to download something entirely different.

2 Infection

The infection or “recruitment” phase of the botnet malware consists of it attempting to compromise a host through any means possible, e.g., taking advantage of an exploit, social engineering, email attachments or the mimicking of desirable content on download sites or on P2P file-sharing networks [6]. A traditional bot, once installed on a new machine, will immediately attempt to phone home through an IRC network or contacting a HTTP C&C server. P2P bots ship with a bootstrapping method to connect to the DHT. Once connected, the newly compromised machine will ask one of its peers for the latest command. Some of the P2P bots require that a specific port is open for the peers to be able to communicate with each other [7]. Through the deployment of a firewall, many of the unnecessarily open ports on any given machine will be blocked. Any new application that attempts to access the network for any reason can also be flagged to the user, e.g., immediately after a recent infection of the botnet malware.

3 Measurement

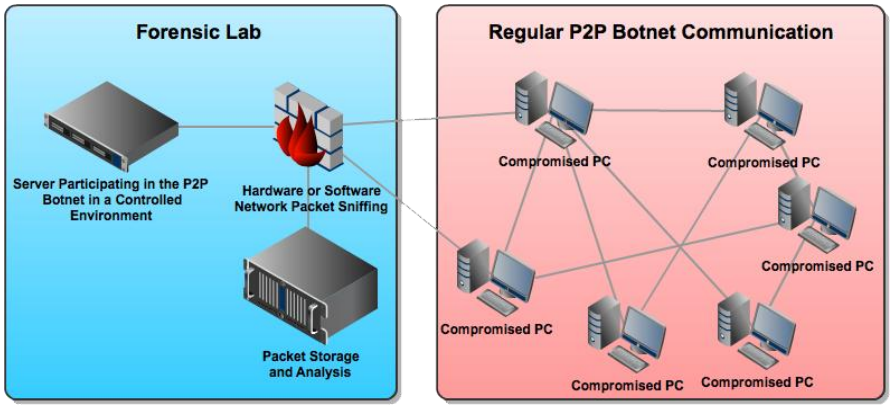
A straightforward method for measuring the size of a botnet is to run a bot on a deliberately infected machine and monitor the resultant traffic. The number of IP addresses the infected node is in communication with can be easily counted having eliminated all non-botnet related network traffic. While it would be unsafe to assume that a single node will ultimately communicate with every other node over time, increasing the number of infected machines (physically or virtually) and amalgamating the results should lead to a more accurate representation.

Byung et al. proposed in 2009 a methodology for improving botnet size estimates through the implementation of a botnet crawler, called Passive P2P Monitor (PPM) [8]. PPM acts as though it were the same as any other node on the network by implementing the "Overnet Protocol", as explained below. This method involves mimicking the functionality of a regular bot with regards to maintaining the DHT. For each peer the crawler connects to, it can ask for a list of all known peers. In this manner, a list of all known peers on the network can be compiled.

4 P2P Botnet Investigation

As quickly as botnet technology is evolving, so too is the methodology for trying to investigate the latest botnet advancements. The objective of any botnet investigation is to attempt to decipher the methods of communication used in order to eavesdrop on the botnet chatter in an attempt to record the manner with which the botnet propagates itself, what commands the botnet is executing, what systems are at risk and how many machines are infected. There are three main approaches to P2P botnet investigation [6]:

Fig. 3. Typical Investigation Network Topology



1. Deliberately infect a host and participate in the botnet. This is the most realistic scenario insofar as a real machine is infected and, as a result, no flags should be raised to either the bot client or any other peers that an investigation is taking place. The network traffic of the machine can be monitored and analysed.
2. Deliberately infect a virtual host (or multiples thereof). This allows multiple bot clients to run on the same physical machine allowing much more network traffic to be gathered in a shorter period of time. However, many modern bots have the ability to detect if their host is a virtual machine and may adjust their behaviour accordingly.
3. Create a crawler and mimic the protocol used by the botnet. In order for a crawler to be built, the bot itself will need to be completely reversed engineered. The crawler can then act as though it were a regular bot on the network to every other peer. This method awards the investigator much control over the network, from enumeration to forwarding bogus commands.

Irrespective of the investigation method used, the topology of the investigation will appear similar to that outlined in Fig. 3 above. A client machine in a controlled forensically sound environment will attempt to partake in the botnet. In order not to raise any flags to any built in counter-forensic measures to either the botnet client or any other peers on the network, this client machine must appear as any other regular infected machine. All network communication from that client machine can then be monitored, recorded and analysed.

4.1 Obstacles in P2P Botnet Investigation

Many of the obstacles facing an investigation on P2P botnets are shared by the investigation of any P2P network, documented or undocumented [9]:

1. Dynamic Host Configuration Protocol (DHCP) – Due to a typical lease from an Internet service provider lasting in the order of 2-7 days, dynamic reallocation of the same IP address may result in two or more infected machines participating in the network appearing as a single peer.

2. Proxy servers - Similar to the issue caused by DHCP, any bots that access the Internet through a transparent or anonymous proxy server will also appear as a single bot.
3. Network Address Translation - Numerous machines behind a shared router may appear to the outside world as a single machine as they share a single IP address.
4. Encrypted Communication – Should the bot employ encrypted communication, the only method available for investigation is to attempt to reverse engineer the bot. The decryption key for any incoming commands must be stored within the bot's client.
5. Difficulty in Take Down - Fighting back against botnets is often a matter of discovering their weak spot. Traditionally this has meant attempting to take down their centralized C&C server [7]. However, with the popularity of employing a fully decentralized network design, the ability to take down a botnet has been made considerably more difficult. Should the bot be reverse engineered, it's possible that the botnet could be "imploded", i.e., through the issuing of an uninstall command to each infected node.

5 Case Studies

5.1 Nugache Botnet

Nugache uses a list of 22 hardcoded IP addresses which each newly infected host attempted to connect to [6]. These 22 hosts maintain a list of active nodes, which they share with each new node. The list of active nodes that any given peer maintains always contains the initial 22 hosts, along with any newly shared active IP addresses. The weakness of this design is that once these 22 hardcoded nodes are taken down, no newly connecting peer will be able to gather its initial list of other peers to communicate with. The Nugache botnet communicates across its own bespoke network protocol. The communication between each node is not encrypted, but there is a degree of obfuscation employed [7].

5.2 Storm Worm

The "Storm" botnet, first discovered in January 2007 [10], is the first botnet discovered that utilised a P2P protocol. It spread through a mixture of social engineering and exploiting vulnerabilities in Windows XP and Windows 2000. The social engineering aspect of the worm was realised through the sending of topical newsworthy email with attachments or links to videos and pictures, which were in fact executables to infect the user's machine. When it infected any given machine, it would disable the Windows firewall and open a number of TCP and UDP ports. Communication in the Storm botnet relies on the "Overnet Protocol". Once the malware was installed and the host machine was configured, it would then bootstrap onto the Overnet network and start listening for commands. The worm was also engineered to aggressively attack anyone who attempted to reverse engineer it [11].

The Overnet Protocol utilises a Distributed Hash Table (DHT) storing the IP addresses and unique IDs of each active peer in the network [8]. It is based on the Kademlia algorithm, similarly to BitTorrent [11]. Kademlia assigns a 160-bit hash ID to each participating peer on the network. Each peer maintains a local routing table consisting of the binding values for other peers that are "close" to their own ID.

5.3 Waledec Botnet

The Waledec botnet has striking similarities to the Storm botnet, while simultaneously exhibiting unique refinements that make it more robust and in part more vulnerable to attack. Waledec follows a hierarchical architecture design. The lowest level were the spammer nodes, which, as their name implies, were responsible for sending spam emails. These spammer nodes communicated exclusively with repeater nodes or super-nodes. These super-nodes, in turn, were in control of the communication with the spammer nodes and would receive their commands from the next level up, known as the sub-controllers [12]. The highest level in the hierarchy, the C&C server, only communicated directly with these sub-controllers.

Similarly to the Storm botnet, the Waledec binary contains a list of hardcoded nodes to use to bootstrap onto the network. In the event of all of these hardcoded nodes being offline, a dynamic URL is also included in the binary to fall back on HTTP to receive commands. Due to this HTTP fall-back, this category of botnet is sometimes referred to as a “HTTP2P” botnet [13]. Communication between nodes was encrypted, initially using a constant key for all nodes, which later evolved into a frequently changing key, which would be created at the C&C server and passed down the hierarchy [12].

6 Conclusion

The P2P botnet topology is a desirable one to choose for botmasters and it affords them an additional level of anonymity when conducting their crimes. The ideal design for a P2P botnet is one that is completely decentralised, utilises unique encryption methods and operates on a bespoke network protocol for communication. Investigation of such a botnet may prove particularly difficult. However, a combination of research, network monitoring, deep packet inspection and network crawling should result in successful, albeit more labour intensive, investigations. The requirement for any newly infected node to have a starting point to bootstrap onto the network as well as seek out other active nodes on the network will always leave an approach for detection and monitoring.

7 Acknowledgement

This project is co-funded by the Irish Research Council for Science, Engineering and Technology and Intel Ireland Ltd. through the Enterprise Partnership Scheme.

Bibliography

1. Zhu, Z., Lu, B., Liao, P., Liu, C., Cui, X.: A hierarchical hybrid structure for botnet control and command. In : Proceedings of 32nd Annual IEEE International Conference on Computer Software and Applications, pp.967-972 (2008)
2. Rajab, M., Zarfoss, J., Monrose, F., Terzis, A.: My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging. In : Proceedings of the First USENIX Workshop on Hot Topics in Understanding Botnets (HotBots'07), pp.5-5 (2007)
3. DynDNS. Available at: http://dyn.com/dns
4. No IP. Available at: http://www.no-ip.com
5. Jimenez, R., Osmani, F., Knutsson, B.: Towards automated detection of peer-to-peer botnets: on the limits of local approaches. In : Proceedings of the 2nd USENIX conference on large-scale exploits and emergent threats: botnets, spyware, worms, and more, p.3 (2009)
6. Dittrich, D., Dietrich, S.: Discovery techniques for P2P botnets. CS Technical Report 2008--4, Stevens Institute of Technology (2008)
7. Schoof, R., Koning, R.: Detecting peer-to-peer botnets. Unpublished Paper http://staff.science.uva.nl/~delaat/sne-2006-2007/p17/report.pdf, University of Amsterdam (2007)
8. Byung, B., Kang, H., Chan-Tin, E., Lee, C., Tyra, J., Kang, J., Nunnery, C., Walder, Z., Sinclair, G., Hopper, N., Dagon, D., Kim, Y.: Towards complete node enumeration in a peer-to-peer botnet. In : Proceedings of the 4th International Symposium on Information, Computer, and Communications Security (ASIACCS '09), pp.23-34 (2009)
9. Scanlon, M., Hannaway, A., Kechadi, M.-T.: A Week in the Life of the Most Popular BitTorrent Swarms. In : Proceedings of the 5th Annual Symposium on Information Assurance (ASIA '10), pp.32-26 (2010)
10. Grizzard, J., Sharma, V., Nunnery, C., Byung, B., Dagon, D.: Peer-to-Peer Botnets: Overview and Case Study. In : Proceedigns of First USENIX Workshop on Hot Topics in Understanding Botnets (HotBots'07)
11. Mukamurenzi, N. M.: Storm Worm: A P2P Botnet. Master of Science Thesis in Communication Technology, Department of Telematics, Norwegian University of Science and Technology (2008)
12. Sinclair, G., Nunnery, C., Kang, B. B.-H.: The waledac protocol: The how and why. In : Proceedings of 4th International Conference on Malicious and Unwanted Software (MALWARE), pp.69-77 (2009)
13. Jang, D., Kim, M., Jung, H., Noh, B.: Analysis of HTTP2P botnet: case study waledac. In : Proceedings of IEEE 9th Malaysia International Conference on Communications, pp.409-412 (2009)