The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets

Evan Cooke,[*] Farnam Jahanian,[*†] Danny McPherson[†]
[*]Electrical Engineering and Computer Science Department
University of Michigan
{*emcooke, farnam*}*@umich.edu*
[†]Arbor Networks
danny@arbor.net

Abstract

Global Internet threats are undergoing a profound transformation from attacks designed solely to disable infrastructure to those that also target people and organizations. Behind these new attacks is a large pool of compromised hosts sitting in homes, schools, businesses, and governments around the world. These systems are infected with a *bot* that communicates with a bot *controller* and other bots to form what is commonly referred to as a *zombie army* or *botnet*. Botnets are a very real and quickly evolving problem that is still not well understood or studied. In this paper we outline the origins and structure of bots and botnets and use data from the operator community, the Internet Motion Sensor project, and a honeypot experiment to illustrate the botnet problem today. We then study the effectiveness of detecting botnets by directly monitoring IRC communication or other command and control activity and show a more comprehensive approach is required. We conclude by describing a system to detect botnets that utilize advanced command and control systems by correlating secondary detection data from multiple sources.

1 Introduction

Global Internet threats are undergoing a profound transformation from attacks designed solely to disable infrastructure to those that also target people and organizations. This frightening new class of attacks directly impacts the day-to-day lives of millions of people and endangers businesses around the world. For example, new attacks steal personal information that can be used to damage reputations or lead to significant financial losses. Current mitigation techniques focus on the symptoms of the problem, filtering the spam, hardening web browsers, or building applications that warn against phishing tricks. While tools such as these are important, it is also critical to disrupt and dismantle the infrastructure used to perpetrate the attacks.

At the center of these threats is a large pool of compromised hosts sitting in homes, schools, businesses, and governments around the world. These systems are infected with a *bot* that communicates with a bot *controller* and other bots to form what is commonly referred to as a *zombie army* or *botnet*. A bot can be differentiated from other threats by a communication channel to a controller. Many bots found in the the wild today are a hybrid of previous threats combined with a communication system. They can propagate like worms, hide from detection like many viruses, and include attack methods from toolkits.

The magnitude of the botnet problem is just beginning to be carefully documented. According to a recent report, the number of new bots observed each day rose from less than 2,000 to more than 30,000 over the first six months of 2004 [7]. The total number of bot infected systems has been measured to be between 800,000 to 900,000 and CERT has described botnets with more than 100,000 members [12, 6].

Botnets are a very real and quickly evolving problem that is still not well understood. In this paper, we outline the problem and investigate methods of stopping bots. We identify three approaches for handling botnets: (1) prevent systems from being infected, (2) directly detect command and control communication among bots and between bots and controllers, and, (3) detect the secondary features of a bot infection such as propagation or attacks.

The first approach is to prevent systems from being infected. There are a range of existing techniques, including anti-virus software, firewalls, and automatic patching.

The second approach is to directly detect botnet command and control traffic. Botnets today are often controlled using Internet Relay Chat (IRC) and one possible method of detecting IRC-based botnets is to monitor TCP port 6667 which is the standard port used for IRC traffic [9]. One could also look for non-human behavioral characteristics in traffic, or even build IRC server scanners to identify potential botnets [17, 19].

We argue there is also a third approach that detects botnets by identifying secondary features of a bot infection such as propagation or attack behavior. Rather than directly attempting to find command and control traffic, the key to this approach is the correlation of data from different sources to locate bots and discover command and control connections.

In this paper we investigate the second and third approach for stopping botnets. The problem with the first approach is that preventing all systems on the Internet from being infected is nearly an impossible challenge. As a result, there will be large pools of vulnerable systems connected to the Internet for many years to come.

The paper begins by describing how the botnet problem is evolving by tracing the origins of bots. We then demonstrate the size of the botnet problem using evidence from the operator community, the Internet Motion Sensor project [2], and experimental data collected from a honeypot experiment. With this information, we examine current IRC-based botnet communication and detection strategies. Next, we show how finding botnets by detecting command and control messages will become less effective as attackers move to other communication topologies and obfuscate their communications. We conclude by describing a system to identify botnets by correlating secondary detection data with host-based forensic information.

2 Bots

Studying the evolution of bots and botnets provides insight into their current capabilities. One of the original uses of computer bots was to assist in Internet Relay Chat (IRC) channel management [16]. IRC is a chat system that provides one-to-one and one-to-many instant messaging over the Internet. Users can join a named channel on an IRC network and communicate with groups of other users. Administering busy chat channels can be time consuming, and so channel operators created bots to help manage the operation of popular channels. One of the first bots was Eggdrop, which was written in 1993 to assist channel operators [1].

In time, IRC bots with more nefarious purposes emerged. The goal of these bots was to attack other IRC users and IRC servers. These attacks often involved flooding the target with packets (i.e., DoS attacks). The use of bots helped to hide the attacker because the attack packets were sent from the bot rather than directly from the attacker (assuming a non-spoofed attack). This new level of indirection also allowed multiple computers to be grouped together to perform distributed attacks (DDoS) and bring down bigger targets.

Larger targets required more bots, and so attackers looked for methods to recruit new members. Since very few users would agree to have their computers utilized for conducting packet floods, attackers used trojaned files and other surreptitious methods to infect other computers. For example, bots such as SubSeven Bot, Bionet Bot, Attack Bot, GTBot, EvilBot, and Slackbot are often simple to install remotely or hide in potentially legitimate files [10].

As the economic incentives to use bots for DoS extortion, spam, phishing and other attacks have emerged, the bot infection process has become more automated. For example, SDBot [11] (also known as rBot) can propagate using many different mechanisms such as open file shares, p2p networks, backdoors left by previous worms, and exploits of common Windows vulnerabilities such as WEBDAV [14], DCOM RPC [13], and LSASS [15]. The attack and communication capabilities of modern bots have also become extremely advanced. For example, Agobot [4] (also know as Phatbot or Gaobot) has a large range of built-in attack capabilities including denial of service attacks, a proxy for spam, GRE tunneling, and password sniffers.

In many respects, the bots found in the the wild today are a hybrid of many previous threats integrated with a command and control system. They can propagate like worms, hide from detection like many viruses, and include attack methods from toolkits. Of even greater concern, the construction of bots is now very much a cooperative effort. An example is the source code of SDBot which contains comments from many different authors. The result is a proliferation of different bot variants. As of August 2004, SDBot has been reported to have more than 4,000 variants [11].

3 Botnet Measurements

As bots have evolved, evidence has emerged suggesting the number of bot infection has grown dramatically. In this section we show the growing problem from the perspective of the operator community, using data from the Internet Motion Sensor project, and using data generated by a honeypot experiment.

3.1 Operator Experiences

As criminals update their tools for the digital age, they have turned to bots as a weapon of choice. Those that run digital networks are caught between the attackers and their targets, but also have a unique perspective on the situation.

To better understand bot behavior, we informally interviewed five major backbone operators at Tier-1 and Tier-2 providers. They indicated that the botnet problem is very real, and something they combat frequently. They also provided interesting insight into current botnet trends.

While the number of botnets appears to be increasing, the number of bots in each botnet is actually dropping. A few years ago, botnets with 80k to 140k members were observed [6]. Today, botnets with a few hundred to a few thousands hosts are common. There are several factors that may be driving this trend. First, smaller botnets are more difficult to detect and may be easier to sell or rent.

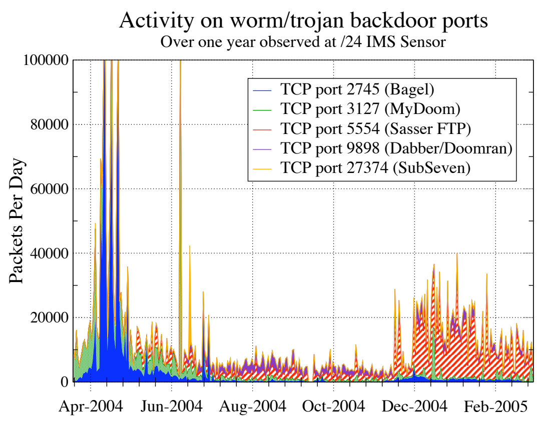

Figure 1: Backdoor activity over one year as observed by a /24 IMS sensor

Another major consideration is the additional *firepower* of to each bot due to the proliferation of DSL, cable, and other broadband access technologies. For example, only a few hundred hosts having a cable broadband Internet connection with an upstream bandwidth of 1Mbps can saturate a high-speed OC3 (155 Mbps) Internet link used by large businesses.

What is clear is that botnets have become a business opportunity, and the characteristics of botnets today often correspond to economic considerations. For example, botnet controllers have been observed building botnets-to-order. These custom botnets might consist of systems within educational networks or, more frighteningly, of systems from government networks.

3.2 Botnet Propagation

Accurately measuring the number of active botnets is very difficult because there are few overt characteristics to identify. One method of tracking bots is to look for propagation activity. The mechanisms used by bots to spread vary greatly, but one major mechanism is to scan for vulnerabilities. These weaknesses can be in production software or in other malware. For example, the Bagel and MyDoom worms left backdoors that could be used to run arbitrary code. These backdoors are known infection vectors for bot activity. Starting in April 2004, the Internet Motion Sensor project (IMS) observed a large uptick in activity on these backdoors. The IMS is a network of sensors that monitors 60 unused address blocks at 19 diverse organizations [2]. Figure 1 shows the number of packets received at well-known backdoor ports left by worms and trojans over a one year period at a /24 (256 address wide) IMS sensor. Figure 1 demonstrates the large amount of continued activity on these port.

3.3 Bot Honeypot

To get a better understanding of bots firsthand, we set up an experiment to measure botnets on a real network. The idea was very simple. If botnets are such a serious problem, then a vulnerable system should be quickly recruited into a botnet when placed on the Internet [5, 19]

The experimental setup was as follows. We placed a new system with a fresh installation of Windows 2000 and XP without any service packs (i.e. a honeypot) behind a transparent proxy device (FreeBSD bridge). The proxy performed three operations: (1) it rate limited traffic in and out to 12KB/s; (2) it disallowed access to systems on the local network; and (3) it logged all traffic to and from the vulnerable system. The proxy was carefully monitored to ensure that the honeypot did not perform any illicit activities such as sending spam or participating in attacks.

We performed 12 experimental runs in which each run was typically 12-72 hours long and traces typically topped 100MB of activity. The traces contained repeated compromises achieved using a wide range of old vulnerabilities, including the DCOM RPC [13] vulnerability and the LSASS [15] vulnerability. The honeypot was often recruited in multiple botnets at the same time.

This experiment suggests a widespread bot problem. Over the 12 runs, only 2 included an infection from a worm (i.e. malware without command and control). While this single deployment may not be representative of other networks, when combined with the experiences of the operational community and the IMS data, the evidence suggests a significant evolution in the threat landscape has already occurred.

4 Botnets Today: Detecting Command and Control

To combat the growing problem of bots, we identified two approaches for detecting botnets: detect the command and control communication, or detect the secondary features of a bot infection. In this section we study methods of detecting botnets by directly locating command and control traffic.

4.1 IRC-based Command and Control

A bot must communicate with a controller to receive commands or send back information. One method for establishing a communication channel is to connect directly to the controller. The problem is that this connection could compromise the controller's location. Instead, the bot controller can use a proxy such as a public message drop point (e.g., a well-known message board). However, because websites and other drop points can introduce significant communication latency, a more active approach is desirable. A well-known public ex-

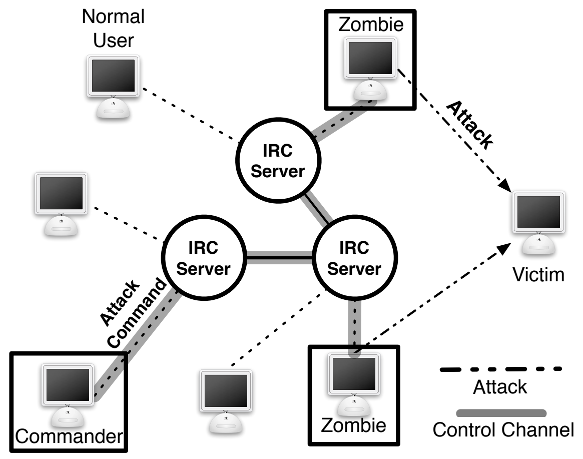

Figure 2: IRC-based botnet DDoS Attack

change point that enables virtually instant communication is IRC.

IRC provides a common protocol that is widely deployed across the Internet and has a simple text-based command syntax. There are also a large number of existing IRC networks that can be used as public exchange points. In addition, most IRC networks lack any strong authentication, and a number of tools to provide anonymity on IRC networks are available. Thus, IRC provides a simple, low-latency, widely available, and anonymous command and control channel for botnet communication.

An IRC network is composed of one or more IRC servers as depicted in Figure 2. In a typical botnet, each bot connects to a public IRC network or a hidden IRC server on another compromised system. The bot then enters a named channel and can receive commands directly from a controller or even from sequences encoded into the title of the channel. The bot and any other bots in the same channel can then be instructed to attack as shown in Figure 2.

4.2 IRC-based Botnet Detection

Today, most known bots use IRC as a communication protocol, and there are several characteristics of IRC that can be leveraged to detect bots. In this section, we describe methods of detecting IRC-based botnets.

One of the simplest methods of detecting IRC-based botnets is to offramp traffic from a live network on known IRC ports (e.g., TCP port 6667) and then inspect the payloads for strings that match known botnet commands. Unfortunately, botnets can run on non-standard ports. We detected at least three such botnets running on high-numbered ports in our honeypot experiment.

Another method is to look for behavioral characteristics of bots. One study found that bots on IRC were idle most of the time and would respond faster than a human upon receiving a command. The system they designed looked for these characteristics in Netflow traffic and attempted to tag certain connections as potential bots [17]. The approach was successful in detecting idle IRC activity but suffered from a high false positive rate.

Given problems such as false positives on live networks, another approach is to use a non-productive resource or *honeypot*. One group set up a vulnerable system and waited for it to be infected with a bot. They then located outgoing connections to IRC networks and used their own bot to connect back and profile the IRC server [19]. However, they did not take the next step and develop a detection system based on the technique.

Rather than connecting to the IRC server directly, another approach is to use a honeypot to catch the bot and then look for characteristics of command and control traffic in the outgoing connections. Using the data collected in our honeypot experiment described in Section 3, we attempted to isolate behavioral invariants in botnet communication. We located all successful outgoing TCP connections and verified that they were all directly related to command and control activity by inspecting the payloads. There were a wide range of interesting behaviors, including connections from the bot to search engines to locate and use bandwidth testers, downloading posts from popular message boards to get server addresses, and the transmission of comprehensive host profiles to other servers. These profiles included detailed information on the operating system, host bandwidth, users, passwords, file shares, filenames and permissions for all files, and a number of other minute details about the infected host.

We then analyzed all successful outgoing connections and looked for specific characteristics that could be used to identify botnet command and control traffic. The results suggested that there are no simple characteristics of the communication channels themselves that can be used for detection. For example, the length of the outgoing connections varied widely, with certain connections lasting more then 9 hours and others less than a second. The number of bytes transfered per connection also varied widely even when we separated out IRC communication from other command and control activity.

The results from our analysis nor the results from previous bot detection efforts has revealed any simple connection-based invariants useful for network detection. One might inspect every payload of every packet however this is currently very costly on high throughput networks. More importantly, attackers can make small modifications that make detection nearly impossible. For example, encrypting traffic, masking flow behavior with random noise, and even switching to different communication topologies can make detection im-

Topology	Design Complexity	Detectability	Message Latency	Survivability
Centralized	*Low*	*Medium*	*Low*	*Low*
Peer-to-Peer	*Medium*	*Low*	*Medium*	*Medium*
Random	*Low*	*High*	*High*	*High*

Table 1: Command and Control Topologies

mensely more challenging (versions of AgoBot with SSL encryption have been reported). In the end, any approach that relies on directly detecting command and control traffic can be defeated by changing the mode or behavior of the communication.

5 Botnets of Tomorrow

The difficulty in capturing bot command and control illustrates the need for a more robust detection approach. In this section we investigate possible advanced bot communication topologies and then present a detection method that gets around these difficulties by identifying other characteristics of a bot infection.

5.1 Command and Control Models

To explore the implications of various bot communication methods, we identify three possible topologies and investigate their associated benefits and weaknesses as shown in Table 1.

Centralized: A centralized topology is characterized by a central point that forwards messages between clients. Messages sent in a centralized system tend to have low latency as they only need to transit a few well-known hops. From the perspective of an attacker, centralized systems have two major weaknesses: they can to be easier to detect since many clients connect the same point, and the discovery of the central location can compromise the whole system.

P2P: Peer-to-peer (p2p) botnet communication has several important advantages over centralized networks. First, a p2p communication system is much harder to disrupt. This means that the compromise of a single bot does not necessarily mean the loss of the entire botnet. However, the design of p2p systems are more complex and there are typically no guarantees on message delivery or latency. A structured p2p location service such as Chord [18] could be used, but such a system might also reveal compromising information about other nodes. Existing p2p anonymity networks could be adapted, however there would be additional processing and latency overhead [3].

Random: A botnet communication system could also be based on the principle that no single bot knows about any more than one other bot. In this topology a bot or controller that wanted to send a message would encrypt it and then randomly scan the Internet and pass along the message when it detected another bot. The design of such a system would be relatively simple and the detection of a single bot would never compromise the full botnet. However, the message latency would be extremely high, with no guarantee of delivery. In addition, the random probing behavior could be detectable.

The three topologies described above can be viewed as a spectrum of information release. That is, they describe the maximum number of nodes any one node will know about at one time. In a central topology the server knows about all nodes, while in a random topology no node ever knows more than one other node. Each topology has specific advantages and drawbacks, and the optimal topology for a given botnet might exist somewhere between the extremes. Many existing communication systems may fall somewhere in between. IRC could be classified as a centralized system, although the server-to-server and client-to-client communication is more like peer-to-peer systems.

The implication of this analysis is that command and control communication is extremely flexible, and a bot could use any number of different channels and different topologies to communicate. Thus, it is difficult for any general botnet detection scheme to rely on specific communication characteristics.

5.2 Advanced Botnet Detection

In the end, all methods that rely on particular communication protocols or topologies like IRC will lose effectiveness as attackers modify their tools. For these reasons, we argue that long term efforts to stop botnets should focus on other methods. One approach is to combine data from existing proven detection systems to identify suspect activity.

Such a system could use data from host detectors, network detectors, or a combination of both. The detector could monitor a production resource or a non-productive resource (i.e. *honeypot*). A key requirement of this approach is the ability to aggregate and summarize data from heterogeneous sources. For example, a system could use a network detector to provide an alert on noisy behaviors such as scanning or DoS activity. The alert could then be traced back to the host that initiated the activity. Using a host-based monitor, the packets could be correlated with the sending process, and the bot program identified [8]. Finally, using the the same host monitor, other other suspected command and control channels related to that process could be identified.

There are clearly a number of challenges in realizing this approach. As a first step, we intend to identify the different available sources of detection data and then evaluate effective trace-back mechanisms on the host. Although more complicated than just identifying command and control messages, we believe a multi-detector correlational approach will provide a more robust and longer-term botnet detection system.

5.3 Challenges of Botnet Disruption

Detection is not the only step involved in stopping a botnet. Given the detection data, an action plan to disrupt the botnet must be formulated. We now describe a summary of the challenges involved in botnet disruption. When a bot is detected, there are two immediate mitigation goals: taking down the bot and taking down the botnet. A bot is different from a worm or virus because it is a member of a larger botnet and there is significant value in taking down the whole botnet rather than just a single node. The process is analogous to law enforcement efforts to capture a gang rather than a single person.

The capture of a single gang member can provide information about a whole gang, and so it may be useful to let that person operate for a time to inform on other members. For bots today, that process is straightforward because many bots communicate with a single IRC server. However, as bots evolve it is likely that it will become increasingly difficult to identify other members of a botnet.

Continuing the law enforcement analogy, gangs that cross international boundaries require the coordination of law enforcement in different countries. Similarly, botnets can be highly distributed, with nodes in hundreds of networks located in many different countries. Stopping botnets will require a significant level of cooperation among providers and some level of automation.

6 Conclusion

This paper has outlined the origins and structure of bots and botnets, and shown how they have evolved to become potent weapons. We studied methods of detecting IRC-based bots and demonstrated three general command and control topologies to illustrate the difficulty of focusing detection efforts on command and control traffic. Based on this understanding, we described an approach to detect botnets by correlating secondary detection information to pinpoint bots and botnet communication.

The threat landscape on the Internet is undergoing an important transformation, and researchers and practitioners need to adapt to the new covert, distributed, and global nature of the threat. This requires collaboration among researchers to devise hybrid data analysis techniques and collaboration between network operators to more quickly and automatically act on threats. Botnets are a global problem that affects the entire Internet community and requires a community effort to stop them.

Acknowledgments

This work was supported in part by the Advanced Research and Development Activity (ARDA). The Internet Motion Sensor project is supported by the Department of Homeland Security, Intel, and Cisco. Many thanks to David Watson, Michael Bailey, Jose Nazario, Tim Battles, Nicolas Fischbach, Chris Morrow, and Rob Thomas for helpful comments and feedback. We would also like to thank all the IMS participants.

References

[1] Eggdrop: Open source IRC bot. `http://www.eggheads.org/`, 1993.

[2] Michael Bailey, Evan Cooke, Farnam Jahanian, Jose Nazario, and David Watson. The Internet Motion Sensor: A distributed blackhole monitoring system. In *Proceedings of Network and Distributed System Security Symposium (NDSS '05)*, San Diego, CA, February 2005.

[3] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009:46+, 2001.

[4] Computer Associates. Win32.Agobot. `http://www3.ca.com/securityadvisor/virusinfo/virus.aspx?id=37776`, July 2004.

[5] Nicolas Fischbach. Router forensics DDoS/worms update. `http://www.securite.org/`, 2003.

[6] Allen Householder and Roman Danyliw. CERT Advisory CA-2003-08 Increased Activity Targeting Windows Shares. 2003.

[7] Kim Legelis, Symantec Corporation. Combating online fraud: An update. `http://information-integrity.com/article.cfm?articleid=100`, 2005.

[8] Samuel T. King, Z. Morley Mao, Dominic G. Lucchetti, and Peter M. Chen. Enriching intrusion alerts through multi-host causality. In *Proceedings of Network and Distributed System Security Symposium (NDSS '05)*, San Diego, CA, February 2005.

[9] John Kristoff. Botnets. 32nd Meeting of the North American Network Operators Group, October 2004.

[10] LockDown Corp. Bots, Drones, Zombies, Worms and other things that go bump in the night. `http://swatit.org/bots/`, 2003.

[11] McAfee. W32/Sdbot.worm. `http://vil.nai.com/vil/content/v_100454.htm`, April 2003.

[12] Laurianne McLaughlin. Bot software spreads, causes new worries. *IEEE Distributed Systems Online*, 5(6), June 2004.

[13] Microsoft. DCOM RPC vulnerability. `http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx`, July 2003.

[14] Microsoft. WEBDAV vulnerability. `http://www.microsoft.com/technet/security/bulletin/MS03-007.mspx`, March 2003.

[15] Microsoft. LSASS vulnerability. `http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx`, April 2004.

[16] J. Oikarinen and D. Reed. RFC 1459: Internet Relay Chat Protocol, 1993.

[17] Stéphane Racine. Analysis of Internet Relay Chat Usage by DDoS Zombies. Master's thesis, Swiss Federal Institute of Technology Zurich, April 2004.

[18] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160. ACM Press, 2001.

[19] The Honeynet Project. Know your enemy: Tracking botnets. `http://www.honeynet.org/papers/bots/`, March 2005.