
Managing Large Botnets

Joint Written Project
John Brozycki, Kevin Bong
April 23, 2007

Managing Large Botnets - © 2007 SANS

Aspects of large botnet management

- Rallying/Recruitment
- Command and Control
- Staying undetected
- Remaining Anonymous
- Using your botnet
- Protecting your Botnet

Managing Large Botnets - © 2007 SANS

In today's environment, there are hundreds of small bot-herders managing thousands of small botnets. However, the largest of the botnets are controlled by a small, elite group. This presentation focuses on the current state of management of the large botnets (over 50,000 bots.) The same aspects come into play for large botnets and for small botnets. In some areas things are the same, but there are some differences in how the bot herders of large botnets manage their bots. The aspects we will focus on for large botnet management include:

Rallying/recruitment – Once a victim host is compromised (generally a home PC with a broadband connection), the bot needs to join to the botnet that the bot herder is creating so that it may be controlled. Frequently, this is a URL (often to a system running an IRC service) that the bot attempts to connect with to register for further instructions. This process is known as rallying.

Command and Control – This is the mechanism by which the bot herder manages and directs the activities of his bots.

Staying Undetected – These are the techniques the bot herder uses to keep his bots from being discovered by the computer's owner, the ISP, authorities, etc.

Remaining Anonymous – These are the techniques the bot herder can employ to reduce the risk of divulging his real identity.

Use of Botnets – Once the bot herder has control, what does he do with it?

Protection of Botnets – How does the bot herder keep control?

Rally/Recruitment

- **Static IPs**
Botnet unlikely to achieve growth using static addresses.
- **Dynamic DNS**
Allows bot herders to easily change IPs of command and control systems.
- **Distributed DNS**
Ultimate control for bot herder. Requires more skill.

Managing Large Botnets - © 2007 SANS

Static IPs

Because every new copy of the malware needs to connect back to a rally point to register with the botnet, and the malware code itself usually needs to be seeded with the rally point addresses, using static IPs means that the rally point(s) will be easy for authorities to find but difficult or impossible for the bot herder to change. This results in rally mechanisms using static IPs usually being the easiest to block or take down. For this reason, it is unlikely that large botnets will utilize hard coded IP addresses as a rally mechanism since the botnet can be detected and the command and control at the IP address(es) taken down or blocked before the botnet can become large.

Dynamic DNS

Dynamic DNS services can also be used for rallying. Dynamic DNS is a service commonly available on the internet that allows clients to quickly and independently change the IP address associated with a controlled name at any time. In place of the hard coded IP addresses, the malware contains DNS names that are hosted by dynamic DNS providers. The bot herder can easily update the IP address references for the dynamic DNS entries to overcome a disabled command and control server. Because most dynamic DNS services operate within the bounds of the law, given time dynamic DNS rally mechanisms can be taken down. Dynamic DNS can also scale to large botnets contingent upon the Dynamic DNS service being able to respond to queries of all of the bots plus normal traffic, or the botnet may use multiple domain names across more

than one dynamic DNS service. Presumably, any service used by the bot herder will have been tested to ensure it is up to the load. If the bot-herder selects a dynamic DNS service that is generally uncooperative with takedown requests and uses a pre-existing domain name provided for free by the dynamic DNS service to multiple customers, not only will it make removal of the DNS records difficult, but requests to take down the entire domain name are less likely to be acceptable since there will be legitimate relying on the same domain name. Finally, dynamic DNS is often free and very easy to use.

Distributed DNS services

Distributed DNS refers to multiple systems that can provide name resolution for a given name. Distributed DNS can be run on compromised systems or systems running in locations where cooperation with local authorities for deactivation is difficult or unlikely. Because the service hosts are uncooperative and there are multiple systems, perhaps in different jurisdictions, this rally mechanism is the most difficult to take down. Distributed DNS services can be scaled by the bot-herder to support large botnets. Distributed DNS offers the bot-herder the greatest degree of control and flexibility but at the cost of complexity as distributed DNS is more complex than using an existing dynamic DNS service.

Command & Control Models

- **Centralized model**
Only this model currently supports the ability to scale botnets to 50,000+ zombies.
- **Peer2Peer**
Maturing, but limited to a maximum botnet size of about 2,000.
- **Distributed/Random**
Too slow to scan for bots. Many infected machines behind filtered network connections and cannot be contacted.

Managing Large Botnets - © 2007 SANS

Centralized

Centralized Command and Control relies on a single host, often a bot itself, to provide command of all of the bots. In large botnets, a pyramid like model may be used where a single bot herder system may pass communications to several agent systems that in turn each have thousands of bots connecting to them[1]. The bots can point to multiple servers for redundancy and improved survivability. Centralized was used by the first botnets and has matured over the years. It is still the most commonly implemented model. Advantages of the centralized model are that it is easy to implement, scales to support large botnets (seen as large as 1.5 million systems with the Toxbot trojan botnet[2] and, unofficially reported to have been significantly larger than this number) and allows for low latency communication between the bot herder and his botnet. The main disadvantage is that, by being in one place, it is more vulnerable to being taken down. Removing the command and control system removes the botnet. Because of the significant advantages of scalability, maturity of the technology, and low latency (bot-herders can push out commands to their botnets relatively quickly) Centralized is by far the preferred and most widely employed model, and the one model that currently supports large botnets.

Peer2Peer

Peer2Peer Command and Control distributes functionality within the botnet itself, not relying on a single system for administration duties. Advantages of the P2P model are that there is no single host that can be removed to bring down C&C, and that detection may be more difficult

since there isn't a single destination in communications. Disadvantages include scalability, since only small quantities of zombies can currently be utilized in a group, and there is no way currently to ensure message delivery or low latency communications. The botnets created by the SpamThru Trojan contain a professional quality P2P command and control, but currently only scale to about 2,000 zombies[3]. While improvements over time may make P2P more viable in the future, right now it isn't capable of supporting large botnets.

Distributed/Random

In the distributed or random model, infected hosts never attempt to contact the command and control. Instead, they sit and wait for communication from the bot herder. To find active bots, the bot herder must scan large blocks of the Internet. This model has not yet been observed in the wild. Advantages include being nearly impossible to detect and taken down as you won't observe infected machines initiating communication in the rallying process, but must wait until they are contacted and instructed by the bot-herder. Disadvantages include latency and scalability. It is very slow and time consuming to scan for, find, and send messages to individual bots. Another disadvantage is the inability to contact successfully infected bots behind NAT routers and firewalls. Because of these disadvantages, botnets based on distributed/random command and control cannot become large botnets.

[1] Poor, Mike. Personal Interview. 15 April 2007.

[2] Keizer, Gregg. *Dutch Botnet Bigger Than Expected*. 21 October 2005. 14 April 2007.

<<http://www.informationweek.com/story/showArticle.jhtml?articleID=172303265>>

[3] Higgins, Kelly Jackson. *Spammers Turn the Tables Again*. 20 October 2006. 14 April 2007.

<http://www.darkreading.com/document.asp?doc_id=107951>

Command & Control Protocols

- IRC
Mature, robust, and efficient. It works and continues to be used.
- HTTP
Very useful when IRC ports are blocked.
- Other protocols
Don't offer scalability or maturity needed for a large botnet.

Managing Large Botnets - © 2007 SANS

IRC

Internet Relay Chat was used by the first piece of botnet malware, the Pretty Park Worm in 1999[1]. It is the most commonly used and most mature protocol used by botnets today. Because of the simplicity and low overhead of the IRC format, it is highly scalable (an IRC botnet of 1.5 million bots was observed) and offers low latency, so the bot herder can get quick response to his orders. IRC can be set up on existing IRC servers or simply run as a service or daemon on compromised machines. Since IRC supports passwords and private chats, botnets can be somewhat protected (requiring a password to access the channel) as well as divided up into different tasks (via private conversations within a channel or through different channels.) Some bot herders have customized their IRC daemons to function specifically for their purposes. Finally, IRC can scale extremely well by supporting a pyramid like structure, as mentioned in the previous slide. The main disadvantage to IRC is that, once known, the channel can be taken down, effectively ending command of the botnet. The other disadvantage is that the ports for IRC use are often blocked by firewalls. Given all of the advantages, IRC is the predominant protocol used for command and control and supports large botnets. IRC is still used in more than 90% of all botnets. Because it is still effective there is little incentive to devote resources towards new development[2].

HTTP

HTTP is one of the most predominant protocols on the Internet and has been referred to as the “universal firewall traversal” protocol. Because port 80 is usually allowed out of most firewalls (whereas IRC ports are often blocked) botnets can take advantage of this. HTTP is the other main protocol used for large botnets. Currently, in bots such as Bobax, command and control is accomplished via HTTP variables and gets. Bobax botnets have been seen at least as big as 100,000 zombies[3], proving it is capable of supporting large botnets. While the HTTP protocol is utilized, packets don’t look like normal HTTP traffic and may be detected.

Other Protocols

Botnets have been observed to use other protocols, such as peer-to-peer protocols. One example is Phatbot which makes use of Gnutella and Waste, but only scales to about 50 clients[4]. To date, development and execution is not mature and these implementations do not support large botnets.

[1] Canavan, John. *The Evolution of Malicious IRC Bots*, p6. Symantec Corporation. 2005. 17 April 2007.

<<http://www.symantec.com/avcenter/reference/the.evolution.of.malicious.irc.bots.pdf>>

[2] McAfee Avert Labs Blog. *Hello from HotBots '07*. 11 April 2007. 20 April 2007.

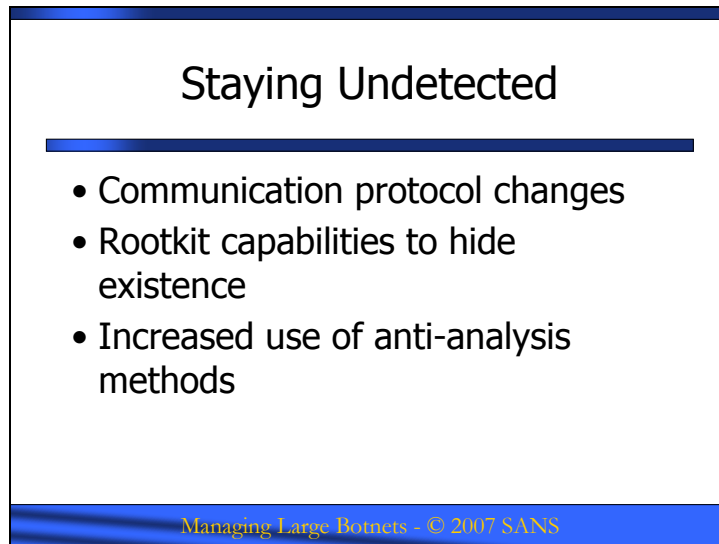
<<http://www.avertlabs.com/research/blog/?cat=3>>

[3] Ramachandran, Anirudh, and Feamster, Nick. *Understanding the Network-*

Level Behavior of Spammers, p18. 2005. 13 April 2007. <<http://nanog.org/mtg-0606/pdf/nick-feamster.pdf>>

[4] LURHQ Threat Intelligence Group. *Phatbot Trojan Analysis*. 15 March 2004. 13 April 2007.

<<http://www.lurhq.com/phatbot.html>>



Staying Undetected

- Communication protocol changes
- Rootkit capabilities to hide existence
- Increased use of anti-analysis methods

Managing Large Botnets - © 2007 SANS

Once a host is compromised and is participating as part of a botnet, there are a number of techniques a bot herder uses to keep his presence on the host or the network unnoticed. This is especially true of the more sophisticated owners of the larger botnets. On many networks, the existence of IRC traffic can be a tip-off to botnet activity. Many of the newer botnets are starting to transition away from IRC and instead use HTTP or “proprietary” protocols for communication[1].

Botnet command and control traffic itself is difficult to detect with traditional IDS techniques. There are no simple characteristics of the command and control channel, such as length of time of connections, quantity of data, or the nature of the data, that themselves can be used for detection. Bot herders have also resorted to encrypting traffic, masking behavior with random noise, and switch communication topologies to evade detection [2]. By switching their command and control protocols from IRC traffic to ubiquitous HTTP or Peer to Peer protocols the botnet commands are more likely to blend in with regular Internet traffic. Using HTTPS goes even farther to hide activity, because the content of the botnet traffic is hidden from filters and IDS sensors[3].

Current bots use NTFS alternate data streams as well as Rootkit capabilities to hide their existence on compromised hosts[4]. The Rbot bot, for example, has features copied and pasted from the open-source rootkit FU. These functions allow Rbot to hide its files as well as hide its

Remaining Anonymous

- Multiple hops between the bot and the bot herder's terminal
- Frequent changes of ISP's, addresses, handles, domain names, and IRC channels
- Located in a non-cooperative country

Managing Large Botnets - © 2007 SANS

The bot herders of very large botnets have the most “visible” botnets, with each bot having the possibility to get traced back to the bot herder. These large bot herders and also often have the most to lose if they get caught. These more elite bot herders have proven to be highly elusive, while the neophytes tend to be sloppy about hiding their tracks[1]. There are a number of techniques bot herders use to remain anonymous. The main way bot herders are protected is the number of levels they build between themselves and the bots in their herd. A bot herder will often issue commands to the botnet by connecting through a chain of compromised hosts or with anonymizing networks such as TOR (The Onion Router.) A bot can be traced back a number of levels, but the bot herder watches their control network and clears out before they can get discovered[2].

Many bot herders reduce the likelihood of being caught by frequently “moving” – changing Internet Service Providers, IP Addresses used, handles, nicknames, or IRC channels[3]. A third way bot herders protect their identity is by residing in or working through non-cooperative countries. The top gangs, most agree, are in Russia, Eastern Europe and Brazil, although there also are a few up-and-coming cybercrime syndicates in Asia[4].

A lot of the script kiddies and other small bot herders tend to be sloppier than their counterparts managing large botnets about protecting their identities. They have fewer resources to move around (physically or virtually), advertise their bots, or to launder money from their activities.

These less-seasoned bot herders are often caught when authorities find them trying to sell their services, or by following the “money trail” back to the bot herder.

[1,4] Acohido, Byron and Swartz, Jon. *Computer crime chronicles*. USA Today. 25 April 2006. 13 April 2007.
<<http://www.crime-research.org/analytics/1959/>>

[2] Information Security Officer for a large Online Banking provider (Anonymous). Personal Interview. 21 April 2007.

[3] Gage, Deborah and Nash, Kim. *Security Alert: When Bots Attack*. 6 April 2006. 13 April 2007.
<http://www.baselinemag.com/print_article2/0,1217,a=175062,00.asp>

Using your botnet

- Used to be lone hackers looking for fame
- Now a lucrative endeavor with a maturing business model
 - Used for SPAM, Extortion, Phishing, DDoS, Adware distribution
 - Botnet “crews” – coder, launcher, miner, washer
 - Management of resources – partitioning
 - Advertising

Managing Large Botnets - © 2007 SANS

Originally, botnets were the realm of lone hackers. They grew and used botnets to gain credibility with other hackers, demonstrating their skills. This is still the case for many of the smaller botnets.

Generally, for the largest botnets, the motive has turned from credibility to profit. Large botnet management is now a lucrative business model often funded by and closely affiliated with organized crime. There are a number of ways large botnets can be used for financial gain:

Relaying SPAM. Routing SPAM through botnets is now the standard practice because it hides the identity of the spammer, and it spreads the SPAM across multiple IPs to bypass blacklists and delivery quotas[1].

Using Denial of Service attacks to extort money from an organization. Online banks and offshore gambling sites are favorite targets for this type of attack, due to the significant losses these organizations can take from even short Internet outages[2].

Phishing, keystroke logging, and other information gathering for identity theft[3].

Distributing adware. In 2006 a California man was sentenced to 57 months in prison for profiting from his 30,000 strong botnet by installing adware and perpetrating click fraud[4].

Renting or selling bots out to others to be used for all these purposes[5].

The lone-hacker has turned into a crew of folks with different roles – The coder (who customizes the bot program), the launcher (who distributes the software and commands the bot herd), the miner (who analyzes the data gathered by the bots), and the washer (who manages and launders the money generated by the botnet use[6].)

The largest of botnets are very carefully managed. Bots are partitioned on different servers based on bandwidth or location. Bots are put to use based on their connection and configuration. For example, a bot on a dialup machine doesn't have much use, so it is segmented off and pushed with Spyware to get the bot herder paid for the installation[7].

Bot herders are also beginning to provide botnets made to order. For example, a botnet could be composed of specialized systems, such as all government or education hosts[8]. Bot herders work to market and advertise their services. In February 2007, a massive Denial of Service attack impacted three of the Internet's root domain servers. It is believed the attacker was demonstrating the power of his botnet to a potential client.

[1] St Sauver, Joe. *Spam Zombies and Inbound Flows to Compromised Customer Systems* 1 March 2005 <<http://darkwing.uoregon.edu/~joe/zombies.pdf>>.]

[2] [Gross, Grant. *Investigator urges companies to report cybercrime* 24 August 2006 <<http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9002710>>]

[3,7] [Naraine, Ryan *Is the Botnet Battle Already Lost?* 16 October 2006 <http://www.channelinsider.com/print_article/Is+the+Botnet+Battle+Already+Lost/191629.aspx>]

[4] *Major Prison Time for Botmaster* 9 May 2006 <<http://www.securityfocus.com/brief/205>>

[5] Evers, Joris. *'Bot herders' may have controlled 1.5 million PCs.* 25 October 2005. 13 April 2007. <http://news.com.com/Bot+herders+may+have+controlled+1.5+million+PCs/2100-7350_3-5906896.html>

[6] Kellermann, Tom. *BOTs: Cyber-zombies.* July 2004. 13 April 2007. <[http://wbln0018.worldbank.org/html/FinancialSectorWeb.nsf/\(attachmentweb\)/Bots/\\$FILE/Bots.pdf](http://wbln0018.worldbank.org/html/FinancialSectorWeb.nsf/(attachmentweb)/Bots/$FILE/Bots.pdf)>

[8] Cooke, Evan and Jahanian, Farnam and McPherson, Danny. *The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets.* Arbor Networks. 2005. 13 April 2007. <http://www.arbornetworks.com/downloads/research130/sruti05_final.pdf>

Protecting your botnet

- From White-hats, and from their competition – other bot herders
- Hardening of zombies
- Redundancy and alternate control channels
- Getting rid of “narc” bots

Managing Large Botnets - © 2007 SANS

Bot herders need to protect their herds not only from the activities of law enforcement and the information security community, but also from other bot herders. The “hacker pride” of having the largest, most powerful botnet has been fueled by the competitive business of using botnets for profit. As a result, bot herders continually work to steal bots from other bot herders or disable other herds. One of the first examples of taking over a system compromised by another piece of malware was the “Doom Juice” worm which took control of PCs infected with the “MyDoom” worm through a backdoor left by the latter.[1] This set the stage for taking another group’s “owned” machines and also demonstrated the risks of leaving back doors. Ironically, bot herders must also concern themselves with the security of their code. For example, many web-based botnet controllers are written in PHP and have vulnerabilities that can be attacked[2].

The war has escalated to a level where bot herders must jealously guard their hijacked computers. In October, a yet-to-be-named Russian gang released a program called SpamThru that infected machines worldwide and quickly amassed an army of zombies nearly 100,000 strong, capable of sending out 1 billion messages each day. To protect the investment, the malicious program actually included a stolen copy of the Kaspersky antivirus program, modified to stop all attacks but its own[3]. The Storm worm includes functions to attack competing bot herds. Computers infected with the “Storm” bot were instructed to attack Web sites run by the rival Russian WarezoV gang, hitting sites with cryptic names like esunhuitionkdefunhsadwa.com. These target sites were the controllers for the WarezoV bot herds. The purpose for these attacks

was to disable the rival's spam network. Without them, the bots could not get commands to distribute spam or perform DDoS attacks[4].

Sophisticated bot herders recognize the weaknesses of centralized control and other single points of failure. They are often building in redundancy and other alternate control channels to allow their herds to survive and regroup when command and control is interrupted. Bot herders are often using free dynamic DNS services to quickly move machines around to avoid detection. There is also evidence of botnets operating like offline terrorist cells, where botnets control each other in a tree-like structure. These tactics have advanced botnets to a point where there is no command and control to take down[5].

Bot herders are also increasingly working to identify when hosts imitating one of their bots is present in the herd. These imitation bots are frequently used by security professionals to monitor or analyze a botnet. To fight these imitation bots, bot herders have started using custom communication mechanisms wildly different from IRC, and have become more particular about their clients and will actively blacklist any host that does not speak the protocol correctly[6]. Anti-analysis techniques, such as changing behavior if virtual environments or debuggers are found, are also used to prevent bots from divulging information about the herd. Bot herders of smaller botnets can generally run a bit more under the radar. They are unlikely to be attacked by the large botnet owners because they aren't perceived as competition or a threat. They are, however, still susceptible to the more seasoned botnet owners compromising and stealing their bots. This is because the bot software developed by the newer and smaller botnet owners will generally be buggier and less sophisticated than the tools the large bot owners have had the time and resources to develop.

[1] Liu, Yana. *W32.HLLW.Doomjuice*. Symantec Corporation. 9 February 2004. 13 April 2007.
<http://www.symantec.com/security_response/writeup.jsp?docid=2004-020909-2916-99&tabid=2>

[2] Poor, Mike. Personal Interview. 15 April 2007.

[3,4] Sullivan, Bob. *Virus Gang Warfare Spills Onto the Net*. 3 April 2007. 13 April 2007.
<http://redtape.msnbc.com/2007/04/virus_gang_warf.html>

[5] Naraine, Ryan. *Is the Botnet Battle Already Lost?*. 16 October 2006. 13 April 2007.
<http://www.channelinsider.com/article/Is+the+Botnet+Battle+Already+Lost/191391_2.aspx>

[6] Nazario, Dr. Jose. *Botnet Tracking: Tools, Techniques, and Lessons Learned*. 2007. 13 April 2007.
<<https://www.blackhat.com/presentations/bh-dc-07/Nazario/Paper/bh-dc-07-Nazario-WP.pdf>>

Weaknesses of Large Botnets

- Increased Visibility – more opportunities to be caught with a large botnet.
- Infrastructure Limitations –
(Rally points / Command & Control):
 - DNS: Providers can disable botnet domain names or dynamic DNS.
 - Centralization: Authorities have a limited number of systems to take down to remove the botnet.
 - IRC/HTTP: IRC often blocked. HTTP usage doesn't look normal so it can be detected.
- Observability - Bot herder can't tell a true bot from a whitehat who is monitoring an IRC channel.
- Financial transactions – Follow the money trail.

Managing Large Botnets - © 2007 SANS

The biggest weakness is that the large botnet owners are more visible than their counterparts who control smaller botnets. The likelihood is higher that one of their bots will infect a computer on a highly monitored and protected network, such as a honeynet or a government or corporate network. This could facilitate quick discovery of the bot and prompt for analysis of its nature and command structure.

Rally points are a significant weakness. Large botnets will likely use DNS. The botnet can be taken down by preventing or altering the resolution of the involved DNS names or suspending the domain name(s) involved. To survive, large botnets must provide redundancy (with multiple rally points) and seek to remain as hidden as possible, but currently they still remain vulnerable.

Command and control also presents a significant weakness to large botnets. Because of limitations of the methodologies and protocols available, today's large botnets will likely be using a centralized command and control over IRC or HTTP. It is the most scalable and responsive method of botnet control. Centralized command and control also creates a single point of failure, and if the bot herder's control infrastructure is compromised the bot herder can lose the entire herd. The IRC protocol is often blocked and current botnet usage of the HTTP protocol does not look like normal HTTP traffic if monitoring is performed. Overall, there are currently limited options available for large botnets.

In primarily using existing, open standards such as IRC, it is possible (and currently being done) that white hats and authorities can monitor botnets, glean information, and take the botnets down. An example of this is Arbor Network's "Blade Runner" custom IRC client, which was specifically designed for monitoring botnets[1]. With a large botnet, bot herders pay less attention to individual bots. They are less likely to notice or detect a "bot" in their herd that is running in a honeynet or is otherwise controlled by one of their foes. This gives law enforcement and security professionals more opportunity to analyze the bot software and monitor the activities and commands sent to the herd. However, as discussed earlier, bot herders are starting to address botnet spying with encryption and detection techniques.

With the shift to a profit motive, perhaps the biggest weakness of managing a large botnet is the money trail. It is much easier to trace payments back to the recipient than it is to track botnet commands back to the terminal they initiated from.

[1] Nazario, Dr. Jose. *Botnet Tracking: Tools, Techniques, and Lessons Learned*. 2007. 13 April 2007.
<<https://www.blackhat.com/presentations/bh-dc-07/Nazario/Paper/bh-dc-07-Nazario-WP.pdf>>

Trends and Newest Innovations

- Continue using the same basic tools and techniques (they work)
- Sophistication of bot software growing
 - Bundling, more features, more automation, redundancy
 - Adaptability – monitor your response and modify the attack in real time
 - More evasion techniques, encryption and digital signatures
- Continued refinement of P2P command and control
- Use of social networking or other public websites for command and control

Managing Large Botnets - © 2007 SANS

As long as the current methods and protocols work, there is little motivation for bot herders to innovate[1]. Bot herders continue experimenting with alternate methods. Steady improvements in alternate protocols and methods and improvements and incorporation of encryption will allow large bots that will be difficult to detect and take down. As long as current methodologies work, it is unlikely that the more advanced groups will reveal new tools and techniques. Why reveal new techniques and give the security professionals a chance to come up with countermeasures while the existing technology still works? Innovation/evolution will come when it is needed, and will probably be driven not by pressure for more protection from law enforcement, but will be driven by competition with other bot herders and desire to increase profitability. While the bot herders will continue to use the same basic tools and techniques, they will continue to enhance those tools:

- * More capabilities and features will be bundled into the current bots.
- * The activities to spread, rally, and control bots will continue to become more automated, allowing the bot herder more anonymity and ability to control larger herds.
- * More redundancy and resiliency will be built in.

One of the key trends as bot software matures is the increased adaptability that is seen. One large online banking provider who was hit by numerous large DoS attacks saw that the attackers were able to monitor their response and modify the attack in real time to circumvent the protection mechanisms that were put in place. This creates a dynamic problem, leaving an

organization unable to build its DoS protection infrastructure based on specific protocols, even if they were used in prior attacks[2].

Large botnets are mostly used for bragging rights and demonstrating the bot herder's mastery. Is a single botnet of 100,000 zombies more valuable than five botnets of 20,000 zombies? The five botnets can be given similar commands to combine their strength. However, by keeping the botnets separate, the bot herder has less to lose if one is discovered and may then employ alternate forms of command and control to help keep his botnets hidden. Another potential capability is the use of digital signatures to sign botnet code and commands. This could prevent botnet takeover by confirming to the bot that the commands came from a legitimate controller[3]. A major current trend is a gradual move away from IRC to web and P2P protocols. HTTP and SSL protocols are used so widely on the Internet now that bot commands using these protocols would be difficult to block or detect within all the other web traffic. Bot networks are increasingly using peer-to-peer systems, designed to look like file and music swapping systems like eDonkey. This prevents Internet service providers from picking out bot communications from regular Web traffic. This also removes the single centralized server that could be shut down, so networks can only be dismantled one infected computer at a time[4]. The question remains if peer-to-peer will scale for the large botnets.

Bots in the future could use commands hidden within more normal looking pages, or even real web pages of large websites that accept content from anyone. For example, analysts have seen some social networking sites that have user profiles that contain bot commands. The bots know to search the website for the given profile, and fetch their commands that way. If the profile is deleted, the bots know to search for another profile which is updated with commands[5]. A similar technique could use One Time Passwords (OTP) and general Internet search engines. If the bot and the controller could generate the same One Time Password, the controller could use the password in a page indexed by Google, or in an eBay listing or Craigslist posting. The bot would search for the OTP, and find the command in the page associated with the OTP[6].

[1] McAfee Avert Labs Blog. *Hello from HotBots '07*. 11 April 2007. 20 April 2007. <<http://www.avertlabs.com/research/blog/?cat=3>>

[2] Information Security Officer for a large Online Banking provider (Anonymous). Personal Interview. 21 April 2007.

[3,6] Fucs, Andre. *New botnet trends and threats*. 17 April 2007. 21 April 2007.<<https://www.blackhat.com/presentations/bh-eu-07/Fucs-Paes-de-Barros-Pereira/Presentation/bh-eu-07-barros.pdf>>

[4] Carney, Kim. *Is Your Computer a Criminal*. 27 March 2007. 21 April 2007. <http://redtape.msnbc.com/2007/03/bots_story.html>

[5] Skoudis, Ed. Personal Interview. 17 April 2007.

The Storm Worm

- Uses many of the new techniques
 - Spreads through multiple channels
 - Different functions are modules that can be added or changed
 - Multiple C&C mechanisms
 - Uses P2P networking
 - Uses rootkit techniques

Managing Large Botnets - © 2007 SANS

The recent Storm bot variants, also known as W32/Small.DAM and Trojan.Peacomm, are examples of a mature, sophisticated bot system that was designed to create, control, and protect a very large bot herd. It is a great example to use to demonstrate many of the areas covered by this presentation.

Most Storm worm infections are accomplished through email spam or website downloads. When initially run, the program links up to other infected hosts via Overnet P2P networking. The remote P2P host provides the bot a URL that points to additional executables to be downloaded and installed on the infected system[1].

The master component is run from a kernel rootkit driver that inserts its code into services.exe. In recent versions, the rootkit driver has a randomly generated name and the injected module is XOR encrypted[2]. It uses this component to participate in the Overnet P2P network. The P2P component also has a hard-coded list of the IP addresses of over 100 peers - this allows it to know about enough of the bot herd to stay in touch, but not disclose or endanger the whole herd.

The P2P protocols are used to direct the bots to websites where additional bot code can be downloaded. Many different components are downloaded via directions from the P2P network. These include an SMTP relay, an Email address stealer, an email virus spreader, a DDoS attack tool, and updated version of the core bot code itself[3]. One of the modules commonly installed

over the bot's P2P network is Trojan.MESPAM. The Trojan.MESPAM component is used to spread the worm by monitoring network traffic for message board posts and injecting the spam text into the middle of a legitimate message board post[4].

The P2P network provides the bot the web addresses to receive code and commands, but DDoS Attack commands are received using a website download. This is likely because the P2P protocols may not be responsive enough for time sensitive activities like a DDoS attack. The DDoS command configuration file has been seen containing IP addresses for websites that appear to be associated with the command and control for a competing spam group - possibly these targets are attacked to hinder the competition[5].

A very recent version, Trojan.Peacomm.B, has been coded with the ability to detect virtual environments such as VMWare and VirtualPC. When a virtual environment is detected, instead of just stopping execution, the program actually attempts to shut down the machine. This appears to be an attempt to shut down honeypot machines to make analysis even more difficult[6].

[1,3,5] Stewart, Joe. *Storm Worm DDoS Attack*. 8 February 2007. 20 April 2007.

<<http://www.secureworks.com/research/threats/storm-worm>>

[2,6] Florio, Elia. *The Evolution of Peacomm to "all-in-one" Trojan*. 17 April 2007. 20 April 2007.

<http://www.symantec.com/enterprise/security_response/weblog/2007/04/the_evolution_of_peacomm_to_al.html>

[4] LeClaire, Jennifer. *Storm Worm Rears Its Ugly Head*. 1 March 2007. 24 April 2007.

<http://business.newsfactor.com/news/Storm-Worm-Rears-Its-Ugly-Head/story.xhtml?story_id=0210002G5EO9>