Understanding and Blocking the New Botnets

April 2008

Researched by Corey Nachreiner, CISSP, Network Security Analyst, LiveSecurity
Written by Scott Pinzon, CISSP, Information Security Analyst, LiveSecurity

Abstract

Botnets are widely regarded as the top threat to network security. This paper explains how botnets have traditionally worked, then contrasts the established model with startling botnet innovations that emerged in 2007 and are continuing to evolve. Examples describe botnets known as Zunker, Gozi, Storm, MayDay, and a kit known as Mpack that helps botmasters easily create malicious web sites. After explaining why the latest innovations make bots a more serious threat than ever, the paper concludes by suggesting defensive countermeasures.

Introduction

Botnets have existed as a known threat since at least 1999, when the first high-profile Distributed Denial of Service (DDoS) attack occurred.[1] By the end of 2007, almost the entire security industry classified botnets at or near the top of any list of threats. Botnets figured prominently in "Top Threats of 2008" prediction pieces, including the *Radio Free Security*[2] podcast from WatchGuard, SANS,[3] McAfee's AVERT Labs,[4]

[1] Most experts cite Trinoo or the Tribal Flood Network as the first DDoS attack platforms, as described here: http://www.cert.org/incident_notes/IN-99-07.html. These were Unix-based attacks; the first Windows DDoS attack is widely regarded as powered by WinNuke in April 1999. (http://en.wikipedia.org/wiki/WinNuke). Denial of Service attacks rose to the mainstream in February 2000, when some of the Internet's most reliable sites were rendered nearly unreachable by Distributed Denial of Service (DDoS) attacks. Yahoo took the first hit on February 7, 2000. In the next few days, Buy.com, eBay, CNN, Amazon.com, ZDNet.com, E*Trade, and Excite were taken down by DDoS. Though damage estimates vary widely, the FBI estimates that the companies suffered $1.7 billion in lost business and other damages. For more, see http://www.auscert.org.au/render.html?it=2259.
[2] http://www.watchguard.com/education/radiofreesecurity.asp
[3] "Top Ten Cyber Security Menaces for 2008," http://www.sans.org/2008menaces/?utm_source=web-sans&utm_medium=text-ad&utm_content=text-link_2008menaces_homepage&utm_campaign=Top_10__Cyber_Security_Menaces_-_2008&ref=22218
[4] "McAfee's Top 10 Security Threats for 2008," http://www.macdailynews.com/index.php/weblog/comments/mcafees_top_10_security_threats_for_2008/

Symantec,[5] Computer Associates,[6] BestSecurityTips.com,[7] Arbor Networks survey of ISPs,[8] and others, including BusinessWeek.[9]

However, in LiveSecurity® Service training efforts and in the LiveSecurity analysts' speaking engagements around the world, we frequently find IT administrators who know the word "botnet" but have only vague concepts of what a botnet is and what it does. Thus, we begin with a description of how classic botnets have worked. We say "have worked," because, until late 2006, botnets relied almost exclusively on a typical architecture and familiar protocols. Then, in 2007 (as we detail), an explosion of botnet innovations created what we dub "Botnets 2.0," threatening us in previously unforeseen ways.

Network administrators who rely on an outdated understanding of bots may not spot the activity of newer bots on their networks, and may have a faulty understanding of how to mitigate the botnet threat. We offer the following description of botnet techniques and architecture in hopes that sysadmins will keep their grasp of the threat up to date, and thus wield solid understanding as the foundation for their botnet defenses.

Botnets 1.0: The Classic Architecture

For more than a decade, malicious hackers have found ways to take control of networked computers that do not belong to them. Commonly, attackers fashion an automated way to exploit a vulnerability in a widely popular program (for example, Internet Explorer), and use the exploit to sneak code onto victim machines en masse.

A typical (but uninformed) computer user probably believes that if attack code resides on a computer, he or she will see symptoms of it. But such is not always the case. Once an attacker establishes control over a victim computer, the attacker is not obliged to exercise that control immediately. In addition, most bot activity does not show up on an infected machine's screen. So, unless the victim uses network traffic capture tools such as a packet sniffer, the victim will not be aware of participating in a botnet.

In a botnet, an attacker establishes a Command and Control Center (hereafter abbreviated C&C). Then he sends onto the Internet small bits of code that have a tiny amount of scripted intelligence. These are the *bots*, short for "robots" because this bit of code can be reproduced infinitely, each iteration acting as an agent for the botmaster. Bots can search entire ranges of IP addresses for computers vulnerable to the exploits that the bot "knows." Once a bot finds a vulnerable machine and successfully infects it, the bot sends a request from the freshly victimized machine back to the C&C, announcing its presence on a new victim and requesting further instructions. The bot code has now become a *bot client* of the C&C. The infected machine is now a *zombie* or *slave* in the bot's network. (See Figure 1 for a diagram of this architecture.) Typically, the botmaster or *bot herder* lets the zombie lie dormant (from an attack perspective) until such time as he needs the resources on the infected machine.[10]

[5] "Top 5 security-menace predictions for 2008," http://www.networkworld.com/news/2007/111307-top-security-menace-2008.html
[6] "CA Internet Security Report Forecasts Top Online Threats for 2008," http://ca.com/press/release.aspx?cid=163385
[7] "10 High Impact Cyber Security Threats in 2008," http://www.bestsecuritytips.com/news+article.storyid+452.htm
[8] "Survey: DoS attacks, bots top security threats," http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9003209
[9] http://images.businessweek.com/ss/07/11/1111_digital_threats/index_01.htm
[10] Many people first coming to grips with Internet security believe their computer will never be hacked because it does not contain data of high value. Botmasters don't care what's on the victim machine. They can utilize its free hard drive space to host illegal files; they can use its ability to send email as part of a spamming network; they can use its address as a relay to help anonymize some other nefarious activity. We'll get into examples later, but the point here is that every networked computer, however humble, looks desirable to a bot herder simply because it *is* a networked computer. All computers have, or are, resources.

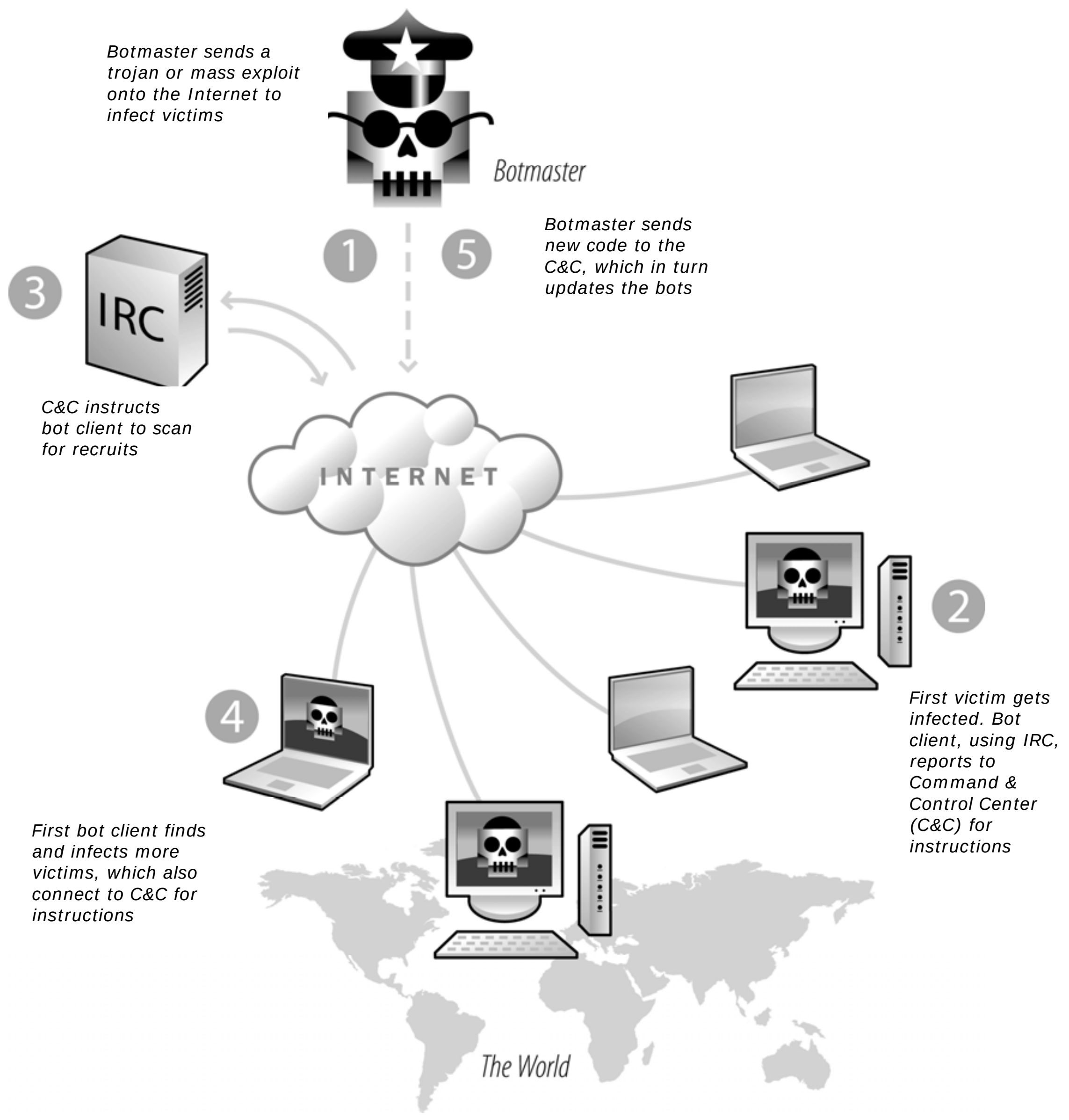
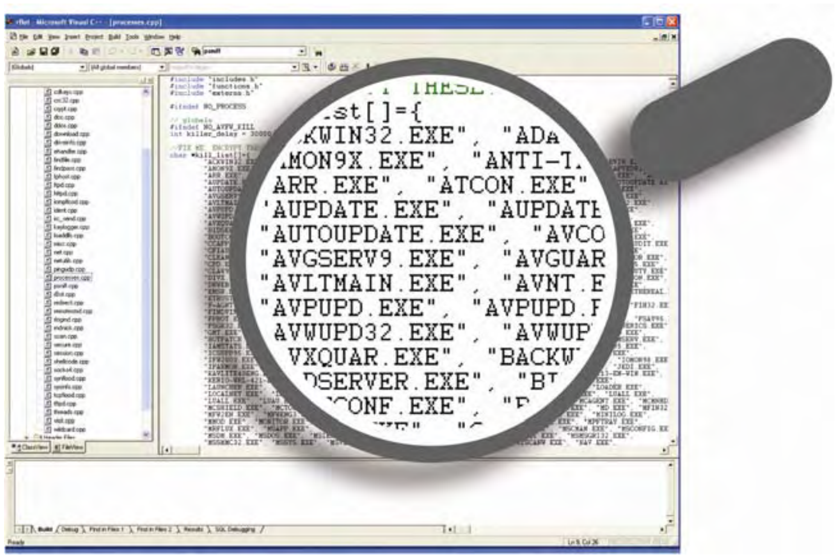

Botmaster sends a trojan or mass exploit onto the Internet to infect victims
Botmaster
1
5
Botmaster sends new code to the C&C, which in turn updates the bots
3
IRC
C&C instructs bot client to scan for recruits
INTERNET
2
First victim gets infected. Bot client, using IRC, reports to Command & Control Center (C&C) for instructions
4
First bot client finds and infects more victims, which also connect to C&C for instructions
The World

Figure 1. Classic Botnet Architecture

However, if he is still building his army of zombies, he might instruct the bot code to scan for more potential victims. If the attacker has programmed his bot clumsily, the victim might sense that her computer has slowed down (due to the high volume of scanning the machine is doing on the bot herder's behalf). However, if the botmaster has throttled his bot code competently, the scanning can carry on with no visible sign to the victim.

In classic botnet architecture, the bot code is built modularly (Figure 2). This approach allows the attacker to move subcomponents of code into or out of his bots readily. The modular code allows him to activate or deactivate exploits based on what vulnerabilities are extant on the Internet, or to pursue a particular project (e.g., a phishing campaign). Because the botmaster can command any and all of his bots to connect to the C&C for instructions, he can update his bot network at will. For example, one week his bots might be optimized for spamming; then, with an update that removes the spamming module and inserts other modules, a week later the same bots might be optimized to seek out financial activity, record login credentials, and report them to a secret server controlled by the botmaster.[11]



```
#include "includes.h"
#include "functions.h"
#include "externs.h"

#ifndef NO_PROCESS

// globals
#ifndef NO_AVFW_KILL
int killer_delay = 30000

.st[]={
KWIN32.EXE", "ADA
MON9X.EXE", "ANTI-T
ARR.EXE", "ATCON.EXE"
'AUPDATE.EXE", "AUPDATE
"AUTOUPDATE.EXE", "AVCO
"AVGSERV9.EXE", "AVGUAR
"AVLTMAIN.EXE", "AVNT.E
"AVPUPD.EXE", "AVPUPD.F
AVWUPD32.EXE", "AVWUP
VXQUAR.EXE", "BACKW
DSERVER.EXE", "BT
CONF.EXE", "F
```

This screen capture comes from Rxbot, commonly found in the wild. In the left panel, each file is another exploit. The magnified file, processes.cpp, tells the bot which processes on the victim's machine to seek and kill (thus disabling anti-virus). The botmaster simply adds or removes files to change what this bot can do.

Figure 2. Bot Code Is Modular

Classic Bot Protocol

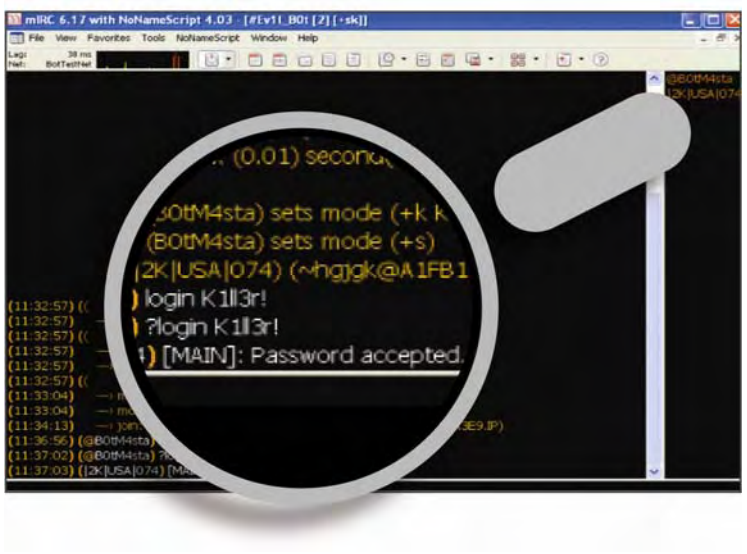
In classic bot architecture, communications between the bot clients and the C&C are accomplished using Internet Relay Chat (IRC) protocol. IRC makes it easy to establish a dedicated "chat channel" for bots to use when reporting to the C&C. As a botmaster's network grows into thousands of zombies, using IRC, the attacker can easily set up separate channels for various subgroupings of his bot army.

[11] For a first-hand look at how the C&C works, or how the bot herder configures his bot clients, refer to the educational videos that the authors made for WatchGuard LiveSecurity Service. These videos demonstrate the use of Rxbot, a popular bot commonly used by attackers. You can download the *Malware Analysis* series at http://www.watchguard.com/education/videos.asp. The parts especially relevant to this white paper are "Botnets, Part 1" and "Botnet Source Code for Overachievers."

Early bot networks were set up on public IRC servers, counting on "security by obscurity" – that is, the bots were relatively easy to find if someone looked for them, but initially ISPs and law enforcement did not know or did not bother to look. However, as time passed, botmasters learned that not only could officials find them and take them out of business with relative ease; but also, in the dog-eat-dog bot underground, their bot armies were too susceptible to being taken over by rival bot herders. Today, it is rare to find a bot network on public IRC servers. The bot underground has moved to private servers.

In addition to setting up private chat servers, bot herders protect their assets by configuring their IRC bots to ignore all commands, unless the command is preceded by a character or string known only to the bot herder (Figure 3). To further obscure the existence of the C&C, the botmaster typically sets it up on one of the zombie computers in his botnet. This also keeps the botmaster safely at arm's length if the C&C is discovered. Whoever legitimately owns the zombie machine would appear to be the botmaster.

To recap the classic "Botnet 1.0" approach, then: a botmaster creates a Command and Control Center, and also creates a bot client. He sends the bot client out onto the Internet looking for its first victim, which it will infect by exploiting a vulnerability in the victim machine's defenses. Once the first bot client infects a victim, it notifies the C&C via IRC and, typically, starts the victim computer – now a zombie under the botmaster's control -- scanning for new victims to add to the botnet. As more and more victim computers become infected, the zombies announce their presence to the C&C and seek yet more victims. When the botmaster decides he has enough new bots, he uses IRC to command the bots to stop scanning. He can also command the bots to visit some Internet location where the bots download update code, effectively repurposing the bots.


mIRC 6.17 with NoNameScript 4.03 - [#Ev1l_B0t [2] [+sk]]
File View Favorites Tools NoNameScript Window Help
Lag: 38 ms
Net: BotTestNet
@B0tM4sta
|2K|USA|074

```
.. (0.01) second
B0tM4sta) sets mode (+k k
(B0tM4sta) sets mode (+s)
|2K|USA|074) (~hgjgk@A1FB1
) login K1ll3r!
) ?login K1ll3r!
4) [MAIN]: Password accepted.
```

This bot herder has programmed his bots to ignore any command unless it begins with a question mark. His attempt to log in using the password K1ll3r! failed until he used the command ?login.

Figure 3. Protecting Bot Assets

Worm, Bot, Trojan: What's the Difference?

Because bots blend many malware technologies, describing bots in words quickly gets confusing. Attackers now overlap technologies in ways that cross traditional boundaries, making classification difficult. Botnets are the ultimate "blended threat." For clarity in this paper, however, we want to state explicitly how LiveSecurity uses the terms "worm," "bot," and "trojan."

Worm. The primary distinguishing characteristic of a worm is that it is an executable program (usually malicious) that can spread itself. The term "worm" does not specify what the malicious payload is. Confusingly enough, a worm could spread a payload that is bot code.

Botnet. Executables must have these capabilities and architecture, at minimum, to constitute a botnet:

- Can run on a victim's computer
- Can connect back to a remote control center
- Can receive and respond to commands from the control center
- Is part of a system automated to control many agents simultaneously
- Deploys without the victim's knowledge or permission

Like worms, many bots include ways to spread themselves. Unlike worms, the ability to self-spread is not inherent in the definition of bot. Though many bots spread themselves, as you'll see in our description of the newer wave of bots, the function of spreading and recruiting can be provided by separate components that the attacker selects.

Trojan. Trojans resemble bots in the sense that they install surreptitiously, can run on a victim's computer, and usually send data back to their creator. The biggest distinguishing feature between a trojan and a bot is that traditionally, an attacker manages trojans manually, one at a time. The industry refers often to "an army of bots" but never "an army of trojans."

Because the term "trojan" derives from the Trojan horse of legend, the term formerly connoted malware that misrepresented itself deceptively. In this older definition, a trojan claimed to have a legitimate purpose but in actuality did something sinister. Over time, this aspect of the definition has faded. For example, the commonly-used term RAT[12] refers to malware that has little visible component at all to the victim. As a further distinction between trojans and bots, the typical trojan can't respond to commands from the control center with the kind of modular flexibility and wholesale updating that bots can. However, though that distinction has been true until today, we suspect it might change in the near future.

In short: a worm could deliver a bot, a worm could deliver a RAT, a RAT could open the door for a bot, but a bot is neither a worm nor a RAT.

Strengths of Classic Botnet Architecture

IRC does not dominate bot architecture by accident. It offers some great advantages to attackers:

- **Relatively easy to set up**. Because it was first invented in the late 1980s,[13] IRC is a mature, well-understood protocol. As protocols go, it is straightforward, and much less complex than many other communication protocols. (This might explain why so often, when law enforcement agencies capture a botmaster, he turns out to be a bright teenaged boy.)

[12] Remote Access Trojans, which typically install a back door on a victim computer so the attacker can enter repeatedly at will
[13] See RFC 1459, or the account at http://www.livinginternet.com/r/ri.htm

- **Lots of free software support**. Because of its long history, and its early use supporting electronic bulletin board services, loads of open source tools, free software, and pre-made scripts and skins support IRC. Almost any tool a botmaster might need exists, and is probably available for free.
- **Plethora of mentors and models**. Because IRC played a key role in the botnet underground almost from its start, the majority of the black hat community has played with it. Some malicious actors have extensive experience with IRC. Others, seeking to establish themselves as "leet hax0rs" (elite hackers), gladly share tips and tricks with new beginners. In short, if a botmaster finds himself stumped by a technical issue, he can find plenty of interactive help, code examples, and debugging advice – after all, IRC is a *chat* channel first.
- **Two-way, real time, immediate**. The precursors of botnets traveled the Internet via the User Datagram Protocol (UDP), notorious for returning to its sender little or no information about the success or failure of its connection attempt (and thus dubbed by some, the "Unreliable Damn Protocol"). In contrast, IRC provides immediate feedback in a two-way session. In effect, the botmaster can "converse" with his bots and maintain a strong sense of what is happening throughout his bot network.

In retrospect, IRC was an obvious protocol for hackers to turn to, since the earliest BBS administrators were already using it. But with the 1990s well behind us, IRC looks relatively creaky as a communication protocol. Botmasters who want to maintain both stealth and uptime find that IRC was not designed with their goals in mind.

Weaknesses of Classic Botnet Architecture

For years, botnet activity was motivated primarily by grudges or mischief. Rival botmasters could try to take each other out by flooding one another's C&C with bot traffic. A hacktivist could try to annoy the CIA by making the CIA's home page inaccessible for an hour or two. But after 2002, as organized crime became more apparently the agent behind many cyber attacks, botnets transitioned into profit centers. Today's botnets are designed to make money, stay up for long periods, and hide the identities of their administrators. IRC is not the best protocol to fulfill those requirements, for these reasons:

- **Centralized Command and Control Center**. In classic botnet architecture, if agents of good can find or disable the C&C, the bot herder is out of business. Because all the bots are programmed to check in with a single command station, the botnet itself is susceptible to a single point of failure: lop off its head, and the beast's tentacles lie inert.
- **Easy to spot**. The classic IRC botnet generates high volumes of traffic. (The downside of a two-way communication protocol is that it generates twice as much traffic as a one-way protocol.) And IRC traffic is obviously IRC traffic, a form of communication that stands out from the tides of HTTP (Web) and SMTP (email) traffic surging across the Internet. IRC natively uses port 6667, a high-order port that most business networks close. All these facets of IRC combine to make the protocol far too easy to notice and monitor for a criminal enterprise's comfort. As of this writing, if a classic IRC botnet goes highly active (for example, spamming out millions of emails, or commanding thousands of bots to hit a link in order to commit *click fraud*[14]), the C&C can only remain up for two to four hours before it is spotted and shut down. This becomes an important driver for understanding why Botnet 2.0 architecture exists: the lifetime of conventional botnets is measured in hours; the lifetime of new botnets is measured in weeks, even months.
- **Lack of automated tools for mining victims**. Though we wrote above that many software tools and mentors support IRC, for most of its life IRC was not intended to accomplish the goals that modern botmasters want to accomplish. A highly successful botnet can amass hundreds of

[14] Any reader unfamiliar with "click fraud" can find a definition at http://en.wikipedia.org/wiki/Click_fraud.

> thousands of victims in its zombie army – at which point, an IRC botnet becomes a victim of its own success. How do you organize a list of 250,000 bots when they're all sending update status as they find new machines, boot up or down, send stolen login credentials, and so on? How do you keep track of every web site your bots have hacked? If you have bot victims all over the world, the chaos of polyglot traffic your bots generate turns into a modern-day Babel. In short, manually managing an IRC botnet is inefficient and confusing at best; impossible at worst.

And so in late 2006, the bot underground started evolving a more effective way.

Botnets 2.0: The First Wave

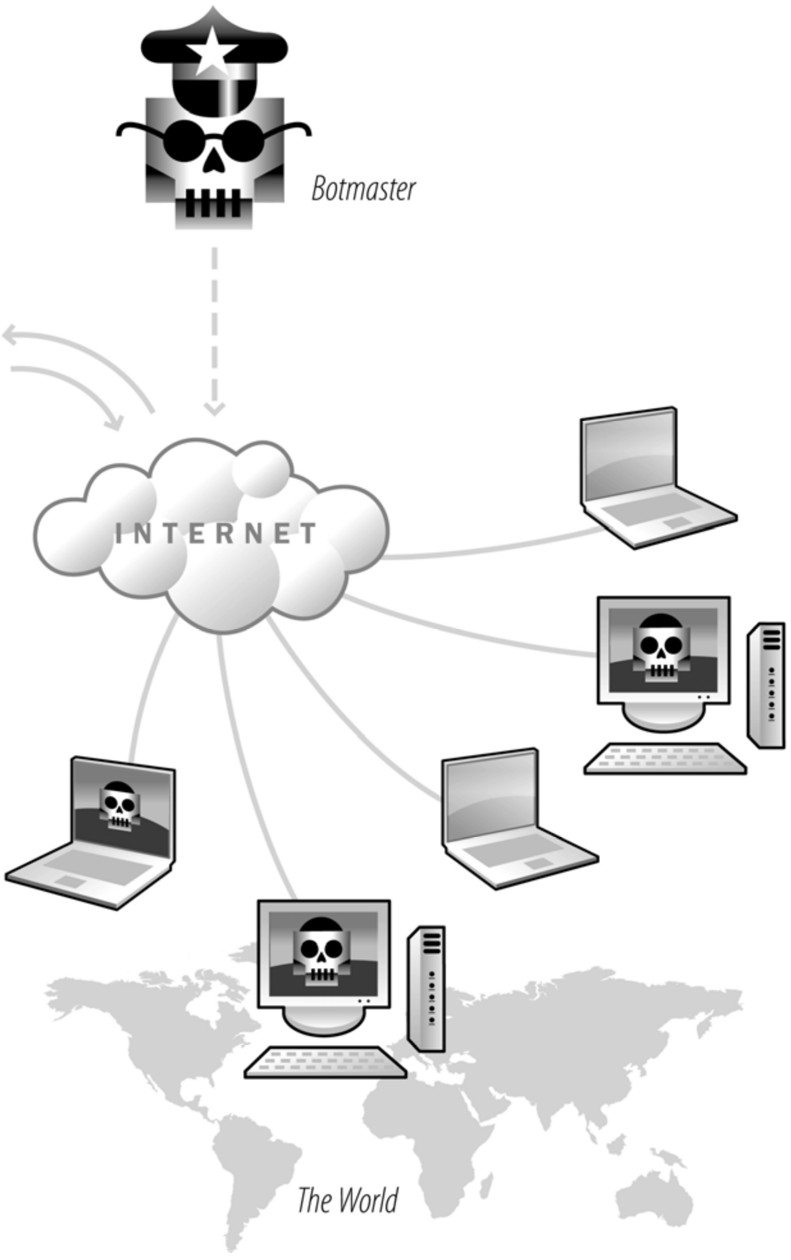
By January 2007, white hat security researchers were finding innovative new bot traffic in the wild.[15] Because one particular bot network (which we'll get to, below) innovated the most, it dominated nearly all of the tech press's coverage of botnets. As a result, several other innovative botnets and helper components flew under the radar. In truth, the amount of technical innovations in botnets during 2007 sends us scrambling for anything to compare it to. The next few passages describe botnet "firsts" that you might or might not have heard of.

Zunker: the Command and Control Center goes pro

In April 2007, Panda Software was the first to describe "Zunker," explaining it as code that provided a backdoor onto victim machines. A victim opened himself for exploitation by running an email attachment or visiting a web site that installed Zunker. You could also (unwittingly) install Zunker's backdoor on your computer by visiting a peer-to-peer (P2P) file sharing site and downloading an enticingly named file that, in reality, was Zunker's backdoor code. With Zunker installed, hackers could depend upon remote access to your computer, and decide later what to do about it.

Zunker departed from Botnet 1.0 architecture, in that instead of relying on IRC to communicate with its victims, it used HTTP (Figure 4). This minor innovation brought a major benefit to Zunker's creators. With an old-fashioned IRC bot, you might stop all its communication by closing port 6667. But in today's web-enabled business environment, who can afford to close HTTP port 80? Doing so would silence Zunker's commands to victim machines – but it would also stop a business LAN and all the employees who use it from accessing the World Wide Web.

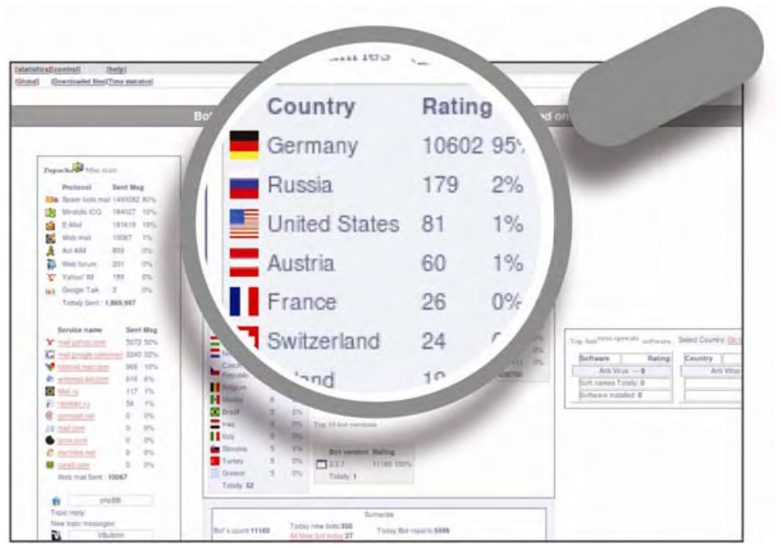
[15] The security industry uses the term "in the wild" when referring to malware found traveling on the Internet. The label "wild" distinguishes such malware from threats that are merely theoretical, and from experimental malware found only in research laboratories. Malware that exists only in research labs, but has never been set loose on the Internet, is often dubbed "zoo" malware.



By running botnet communications over HTTP/TCP port 80, rather than via IRC, botmasters make it harder for administrators to notice and block bot traffic.

Figure 4. Innovations to Bot Architecture

Zunker consisted of a group of well-written, professionally coded PHP scripts[16] and CGI scripts[17] running on a web server. Besides its innovation of running on HTTP, Zunker also introduced a solution to the organizational burdens of running a successful botnet. Instead of the chaotic, assorted list of victims that an IRC bot typically presented to its botmaster, Zunker featured a slick, graphical, well-organized display. It could even display the flag of origin of each victim machine. If Zunker determined that a zombie machine's IP address was located in France, Zunker displayed a tiny French flag next to that entry (Figure 5).



Country	Rating	
Germany	10602	95
Russia	179	2%
United States	81	1%
Austria	60	1%
France	26	0%
Switzerland	24	

Zunker's graphical interface even shows the country where each victim IP address resides.

Figure 5. Zunker's Polished Command and Control Center

What did Zunker's scripts do? They provided a plethora of tools for mining a database of victims. Victim computers could easily be parsed, queried, searched, and sorted. Need to know which zombies in your botnet offer the most open hard drive space? Zunker could list them in descending order. Want to send a spam email, but only from victim machines in Germany? Zunker could provide a list of IP addresses. Zunker was basically the C&C gone totally professional. The influence of organized crime was stamped all over it. Instead of a ragtag, text-based list of victims, a Zunker botmaster had a management package that enabled the efficient exploitation of every resource his victims might yield.

Zunker was cool and slick. But it had one glaring drawback: it was just a management interface that tracked victims. On its own, it couldn't scan machines or recruit victims; it could only lie in wait for victims to stumble into it.

But Russian innovations would solve that problem.

[16] PHP is a recursive acronym for PHP Hypertext Pre-processor. In short, PHP is a scripting language highly optimized to work well with HTML. Web developers use it commonly to create dynamic web pages that can interact with a database.
[17] CGI stands for Common Gateway Interface. It refers to the program or script on a server that will process user input. CGI input is often (but not always) data that a web page user has sent by using a fill-in form.

Gozi: from Russia, with stealth

Around the same time the world was first learning of Zunker, Don Jackson of SecureWorks was blogging about a trojan named Gozi. Jackson had a friend who complained that when he visited certain web sites, his site accounts had been hijacked. An analysis of the friend's computer revealed a malware executable that did not match any previous classifications. It had been on his computer since the end of 2006, and was discovered at the beginning of February 2007.

Thus began an investigative journey that Jackson chronicled in detail.[18] To jump to the ending, what he found was:

- Malware that no anti-virus engine correctly identified for at least 54 days in the wild
- It copied itself into DLLs and registry keys on the victim computer so that it would restart every time the computer rebooted
- The code included kernel rootkit technology so that it could hide from discovery, both visually and programmatically
- The malware could connect to a web-based management system, sending data over HTTP, preformatted to work with MySQL queries
- It could "sniff" web sessions on the victim computer
- The malware installed itself at such a low level that it could read the victim's encrypted traffic before it was encrypted,[19] outwitting browser protections that Microsoft installed to defeat this specific kind of attack

The punch line: the executable, which Jackson named Gozi, had been written specifically to beat the security used in banking transactions. It stealthily harvested confidential login credentials and sent them to a remote server, in clear text, already formatted to respond to database queries. In short, Gozi provided a perfect complement to Zunker. It had the data harvesting capabilities that Zunker lacked, formatted to respond to the types of queries Zunker excelled at.

And before anyone spotted Gozi and figured out what it did, it had been loose in the wild for nearly two months.

Armed with his new discoveries, Jackson and other investigators tried to find the source of Gozi. They ultimately were able to track back to two different servers in Russia in February 2007. In a write-up posted in March 2007, Jackson stated,

> …we were able to retrieve the names of companies and organizations whose customers were affected. We found that over 5,200 home PC users, with 10,000 account records, were compromised and account and login information for applications offered by over 300 organizations was stolen through these infected home PCs. The information stolen contained everything from bank, retail, and payment services account numbers, as well as social security numbers and other personal information. The records retrieved included account numbers and passwords from clients of many of the top global banks and financial services companies (over 30 banks and credit unions were represented), the top US retailers, and the leading online retailers.[20]

Jackson also discovered that Gozi was for sale on Russian web sites, at prices ranging from US$1000 to $2000.

---

[18] Jackson has excellent write-ups on the SecureWorks site, including this one: http://www.secureworks.com/research/threats/gozi/

[19] For the technically-minded reader: Gozi used advanced Winsock2 techniques to intercept SSL/TLS traffic that the victim's browser was about to send. SSL packets were being skimmed and wrapped in a correlated request to the malware's home server via insecure HTTP, thus sending login credentials in the clear to Gozi's author even as the encrypted credentials traveled to their legitimate destination.

[20] "Gozi Trojan," Don Jackson, March 21, 2007, http://www.secureworks.com/research/threats/gozi/

Mpack: Exploit framework on steroids

Throughout this discussion, we've mentioned that botmasters could take over computers that their bots have compromised. What would they do with these zombie machines?

During May and June of 2007, researchers stumbled upon a startling phenomenon. Thousands of legitimate web sites, mostly in Italy, had unwittingly been delivering malware to anyone who visited the sites.[21] Websense soon reported that at least ten thousand hacked web sites had been redirecting victims to a server hosting a "hacking kit" called Mpack.

Mpack provided a Botnets 2.0 example of the kind of code a botmaster might place on a victim machine. Mpack itself was neither a bot nor a scanning tool; in essence, it was an easily customized collection of scripts that could turn a zombie into a malicious web site. This malicious web site, using drive-by download techniques, could in turn automatically infect a second generation of victims.

Mpack represents a sea change in how bots recruit more bots. Classic botnets resembled a gang of thieves sneaking through an apartment complex, testing every door and window to see what homes they could slip into. Mpack inverted the paradigm. Now one of the respectable apartments masks the activities of a drug pusher, whose minions paper the complex with bulletins that entice fresh victims to visit. Where the Botnet 1.0 architecture recruited new victims primarily by having each bot use its own scanning and infection engine, Mpack didn't scan, but instead changed each victim into another booby trap. The botmaster's task was to find ways to drive or lure more victims to the Mpack machines; Mpack would handle the rest.[22]

The Mpack kit contained scripts that could exploit various browser vulnerabilities in Firefox, Opera, and Internet Explorer. Mpack's PHP scripts could detect a visitor's operating system, browser, and geographic location; then, from a library of exploits, select the exploit that was most likely to work on that target. Mpack would run the exploit on the visitor's machine, and if it successfully converted the visitor to a Generation 2 victim, the Generation 1 Mpack computer would record the results in a MySQL-format database.

Because Mpack was highly customizable, what Generation 1 victims delivered to Generation 2 victims was entirely up to the attacker's whim. This made Mpack a perfect tool (for example) for delivering Zunker, which couldn't deliver itself.

In the wild, many botmasters chose to use Mpack with a program called Dream Downloader. Dream Downloader added stealth aspects to whatever payload Mpack was delivering to Generation 2 victims. If a visitor was enticed to visit the malicious web server Mpack had set up, in most cases Dream Downloader could disable the anti-virus software and personal firewall running on the Generation 2 victim. Mpack's scripts would execute on the Gen 2 victim, while Dream Downloader obscured the exploit code so that if it were noticed, debuggers would have difficulty figuring out what the code did and how it did it. Dream Downloader made whatever Mpack was delivering more evasive.

When Panda Software first discovered Mpack, the Russian team who coded it (the self-styled "Dream Coder Team") had assigned a version number of 0.33. Within weeks, Mpack was up to version 0.90. Its

[21] "Europe under intense cyber attack," Gregg Keizer, June 19, 2007, http://www.computerworlduk.com/management/security/cybercrime/news/index.cfm?newsid=3561&pagtype=allchandate&tsb=print

[22] Mpack's ability to infect a victim, which in turn hopes to infect more victims, creates unique communication challenges, because Mpack does not deliver itself. Mpack is like a payload within a payload. Thus: an attacker uses some exploit to infect victim machines, and that exploit delivers Mpack. We'll call the machines infected with Mpack the "Generation 1" victims. Then the attacker consults his database to find who and where the Gen 1 victims are. Armed with this knowledge, he creates spam or uses other methods (such as social engineering) to lure visitors to the now booby trapped Generation 1 computers. Mpack then infects these visitors, which become "Generation 2" victims. Since Mpack can be customized to deliver whatever the attacker wants, the Gen 2 victims need not resemble nor function like the Gen 1 victims at all.

basic package sold on the underground for US$700 to $1000, accompanied by assurances that buyers who paid full price would later receive version updates with new features. Exploits coded to run on Mpack's framework were also available *à la carte*, at prices ranging from as little as US$30 to $150.

Recap

Within the first half of 2007, botmasters had unleashed the first graphically oriented, HTTP-based botnet Command and Control Center; the first bot client code that could read and report on encrypted banking transactions; and distribution components that enabled bot code to seed itself on thousands of legitimate sites before being detected.

Though IRC bots still made up the vast majority of active botnets, suddenly the Botnets 1.0 approach looked antiquated. A new model had emerged, utilizing sophisticated technology formerly associated only with business enterprises. Zunker provided state-of-the-art back-end accounting and management of bot victims. Gozi sent data that any enterprising botmaster, with a simple script, could use to fill Zunker's databases. Mpack scaled up the operation by turning legitimate computers into drive-by download booby-traps. Dream Downloader helped hide all the illicit activity. Botnet architecture had been revolutionized.

And the first wave of Botnet 2.0 innovations had barely begun. Much as the Beatles album *Sgt. Pepper's Lonely Hearts Club Band* shocked the music world of 1967 with its vision, execution, and technical innovations, in 2007, the "Sgt. Pepper" of botnets arrived.

Taking the World by Storm

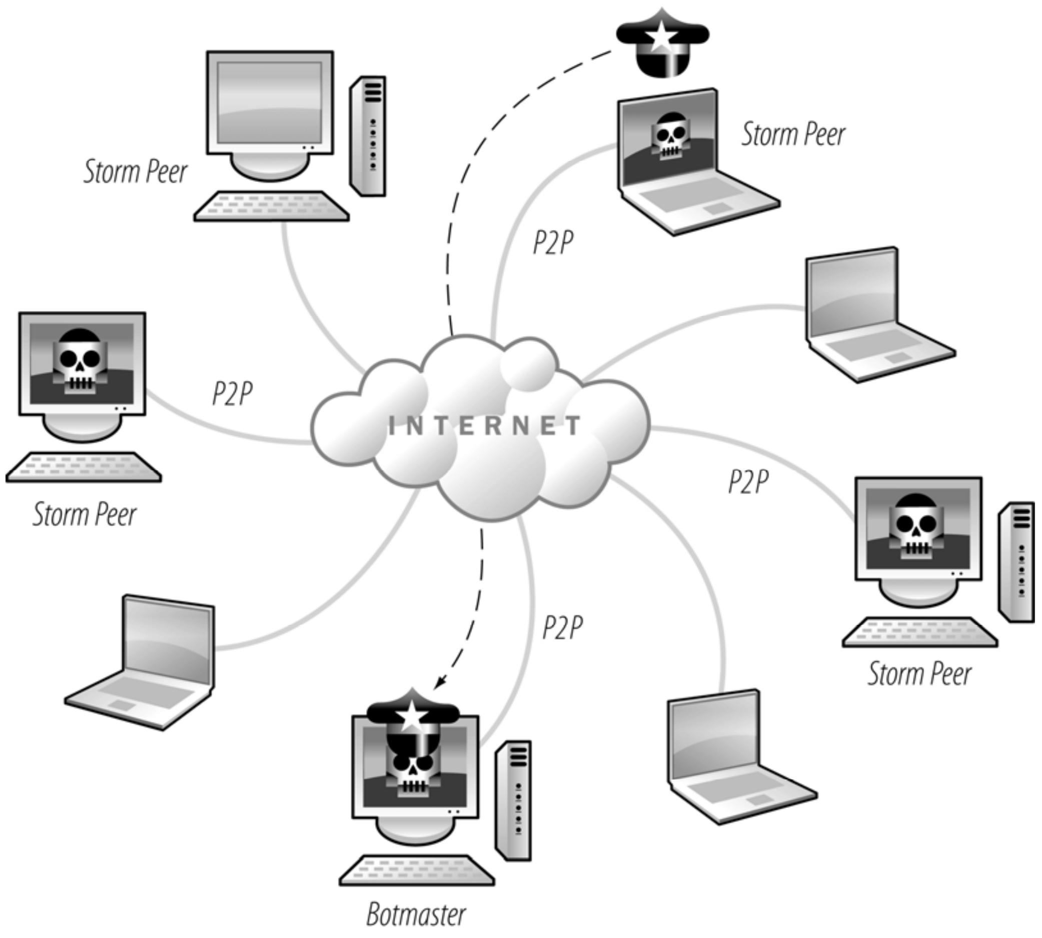
In January 2007, security researchers noticed an onslaught of spam having subject lines related to extreme weather (for example, "230 dead as storms batter Europe"). An attachment usually named "ReadMore.exe" accompanied the spam. If an intrigued recipient tried to open the attachment, the attachment infected the victim's computer with trojan code that could do several things, including:

- Send spam from the victim's computer
- Spread itself (primarily by more email social engineering)
- Cooperate with other infected zombies in Distributed Denial of Service attacks (TCP SYN flood or ICMP Echo flood)
- Download and execute files upon the infected victim's machine
- Receive, install, and execute updates
- Use rootkit technology to hide on the victim machine

Because of the weather-related subject line, F-Secure dubbed this malware the "Storm worm"[23] (even though this malware also used five other subject lines, and wasn't a worm). McAfee at first rated the malware "low profile," and Symantec, who named the malware "Peacomm," rated its threat as "low."[24] Nonetheless, the media jumped on F-Secure's more shrill treatment[25] and waxed hysterical. While the inaccurate reporting and overblown estimations of the power of this malware helped no one, the tech press was right about one thing: Storm was the harbinger of something never seen before. From its humble beginning, Storm evolved throughout 2007 to become the most formidable, highest-traffic botnet that security researchers had ever seen.

[23] F-Secure press release, January 19, 2007. http://www.f-secure.com/f-secure/pressroom/news/fs_news_20070119_1_eng.html
[24] http://www.symantec.com/business/security_response/writeup.jsp?docid=2007-011917-1403-99
[25] Besides the press release cited above, F-Secure released a number of blog entries and quotations from their head of research, Mikko Hypponen. Many of F-Secure's predictions proved accurate over time, but since their main intent was to promote F-Secure's BlackLight "Rootkit Eliminator," some of their worst-case predictions preceded hard evidence. This led us to write a Storm anti-FUD piece on *WatchGuard Wire* in January 2007.

One of Storm's innovations was to abandon IRC, and instead use peer-to-peer (P2P) technology for its command and control communications. Most readers will be familiar with P2P file sharing services such as Kazaa, Napster, Morpheus, Limewire, Bearshare, and BitTorrent. In a P2P network, all member machines really are peers – that is, there is no one central C&C. The list of active peers, and the relevant content they contain, is distributed in many small pieces in the memory of all the nodes participating in the network. Any machine in the network can act in the role normally associated with a centralized, authoritative server, by drawing on resources from other peers. By using P2P protocols (specifically, Overnet[26]), Storm distributed command authority (Figure 6).



The Storm botnet abandoned IRC in favor of P2P protocols such as Overnet. This diagram shows command authority passing from one node to another - any infected Storm machine can act as the command and control center (C&C) at the botmaster's will.

Figure 6. Storm's P2P Architecture

[26] Overnet is a Distributed Hash Table network based on the Kademlia algorithm, first brought to popularity by older versions of eDonkey. Without going into technical details, suffice it to say that this type of network enables the peers to rapidly calculate which fellow peer hosting the desired resource is "closest" logically (requires the fewest router hops to reach). Readers who want to delve further into Overnet's actual workings should read the excellent presentation by Brandon Enright, whose research we helped ourselves to generously and gratefully acknowledge: http://noh.ucsd.edu/~bmenrigh/exposing_storm.ppt#271,16, Slide 16

If law enforcement or other authorities discovered a Storm-infected machine and shut it down, it was merely one node in a network of thousands of peers. This single innovation of using P2P for communications eliminated the typical Botnet 1.0 weakness of having a single point of failure, and contributed strongly to Storm's amazing resilience. In effect, you couldn't shut down Storm unless you shut down *all* of Storm.

As 2007 wore on, Storm evolved like an ever-shifting malware kaleidoscope. At first it spread "pump and dump" stock market spam. Much of its spam was not text-based, but rather, spelled-out messages in bitmapped image files; in essence, a photo of a message. Humans could read the message, but anti-spam software that filtered text strings could neither read nor block the pictures. Over time, Storm spam utilized PDF files, Excel spreadsheets with embedded images, even audio files (MP3) using text-to-speech technology to deliver spam messages. Storm also sent phishing emails; performed Distributed Denials of Service (DDoS) against select groups and organizations; and basically gave every indication of being leased out, in subsections, to paying clients in the Internet's underground.

Malicious activity was possible on this scale because Storm also innovated in evasion. Its creators continually repacked Storm bot clients using different compression and encryption schemes, each time making obsolete anti-virus signatures that were based on previous iterations of Storm. Storm's creators also utilized DNS fast flux techniques to avoid discovery.[27] In using P2P Overnet protocols, the creators changed how Storm generated its Overnet IDs; switched from UDP to TCP; and evolved ways to detect and work around NAT.[28] Besides these innovations, Storm also used all the old-school evasion techniques, such as giving its client files names that sounded like they belonged on the average Windows computer (for example, wincom32.sys and spooldr.sys).

Most startling to security researchers, Storm even developed "strike-back" technology. Josh Corman, a researcher at ISS, reported that when IBM's security team probed a known Storm machine, it would sense their IP address and send back a withering flood of Denial of Service traffic.[29] "As you try to investigate [Storm], it knows, and it punishes," Korman said. "It fights back." In some cases, Storm knocked researchers off the Internet for days – another innovation that helped Storm endure.

All the way until October 2007, Storm kept growing, recruiting ever larger numbers of victims. At its zenith, it seemed everyone in the security profession had written about Storm or made wild estimates about its powers. Storm was credited with having "50 million bots,"[30] was called "the world's most powerful super computer,"[31] and was credited with sending "tens of billions" of email messages *per day*.

While Storm was indeed fearsome, such hyperbolic claims could not be proven. Network security analyst Brandon Enright, a researcher at the University of California in San Diego, attempted to separate hype from reality. He built a crawler that traced Storm traffic from June until October 2007. His crawler beat Storm's strike-back capability by using the eDonkey/Kedmelia algorithm to successfully pose as a peer in Storm's botnet. He then discarded obviously fake Storm traffic, extremely buggy copies of Storm, peers that responded once but never responded again, and Storm-like traffic that was created by other researchers. After those basic adjustments, he concluded that Storm possessed 200,000 "live, reachable" bots at its peak,

---

[27] DNS fast flux and double-flux techniques are complicated ways of frustrating attempts to trace traffic. Fast flux techniques are foundational to the evasiveness of Botnet 2.0 networks, so we encourage the reader to read the excellent white paper from the Honeynet Project's "Know Your Enemy" series, "Fast-Flux Service Networks: An Ever Changing Enemy." http://www.honeynet.org/papers/ff/fast-flux.html Dave Piscitello's shorter recap of the concepts can be heard in the July 2007 episode of our podcast, Radio Free Security, available at https://www.watchguard.com/education/radiofreesecurity.asp.
[28] Network Address Translation, a networking protocol that allows an administrator to hide his network's private IP addresses by having their outbound traffic "translated" to look as if they came from one single public IP address.
[29] Cited at the Interop New York conference, October 2007: http://www.networkworld.com/news/2007/102407-storm-worm-security.html
[30] "Storm Worm Botnet More Powerful than Top Supercomputers," Information Week, September 6, 2007, http://www.informationweek.com/news/showArticle.jhtml?articleID=201804528
[31] "The Internet's Public Enemy Number One," *PC World*, December 2007, http://www.pcworld.ca/news/column/9196fcf50a01040800129dda48843378/pg0.htm

declining to 160,000 bots by October. Not every researcher agreed with Enright, but his methodology was more empirical than that of his naysayers.

Whatever the specific numbers really were, compared to any previous measure, Storm generated gargantuan amounts of traffic. CommTouch, with stations scanning Internet traffic all over the world, reported that in the fourth quarter of 2007, Storm accounted for as much as 7 percent of all the spam in the world.[32] MessageLabs put the figure as high as 20 percent.

Despite excessive interest from researchers and coverage by the popular technical press,[33] as of October 2007, leading security commentator Bruce Schneier could comment,

> We have no idea who controls Storm, although there's some speculation that they're Russian. The programmers are obviously very skilled, and they're continuing to work on their creation… Personally, I'm worried about what Storm's creators are planning for Phase II.[34]

Despite a seemingly unbroken record of innovation and success, Storm's activity slacked off noticeably at the beginning of 2008, subsiding to 85,000 infected machines. Some have speculated that Storm's creators ceased development on it because they indeed moved on to Phase II. In our view, Phase II might be an emerging botnet named MayDay.

MayDay

It took white hat researchers awhile to piece together what was happening, but by late January/early February 2008, the security community realized that a new P2P botnet had arrived, even sneakier and more sophisticated than Storm.[35] Anti-botnet startup company Damballa made most of the breakthrough discoveries, naming the new bot MayDay. According to Damballa researcher Trip Cox, some of MayDay's characteristics and capabilities include (Figure 7):

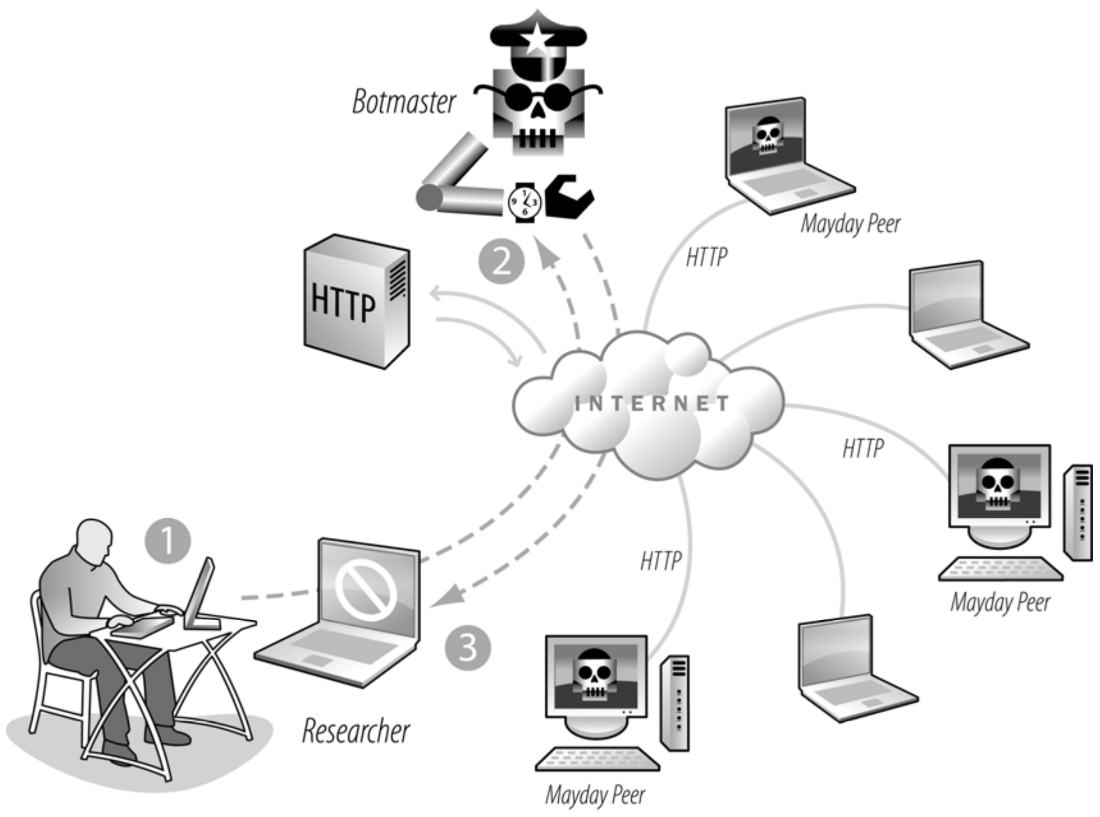
- **Communicates between bot clients and the command center using standard HTTP**. Other bots have used HTTP port 80 so that network administrators cannot block the bot's communication port unless they are also willing to block their network users from accessing the World Wide Web. MayDay takes the trick one step further with the ability of working around *web proxies*.[36]

- **Can fall back to P2P communication methods**. If the main HTTP communication channel isn't working, MayDay bots can alternatively try two other protocols. One method uses a TCP protocol that Damballa has not disclosed. The second method is more sneaky: MayDay can communicate using encrypted Internet Control Message Protocol (ICMP), a protocol admins routinely rely on for its troubleshooting functions, most notably ping and ping echo. As with HTTP, MayDay's ICMP messages travel over a port that admins dare not block. This also makes MayDay the first bot we've heard of that can use three communication protocols.

---

[32] CommTouch Year-End Email Threat Report, http://www.commtouch.com/Site/News_Events/pr_content.asp?news_id=983&cat_id=1
[33] For a long list of some of the articles describing Storm, see the 60 citations in the references section of http://en.wikipedia.org/wiki/Storm_botnet.
[34] "Gathering Storm Superworm Poses Grave Threat to PC Nets," Wired, October 04, 2007, http://www.wired.com/politics/security/commentary/securitymatters/2007/10/securitymatters_1004
[35] "MayDay! Sneakier, More Powerful Botnet On the Loose," DarkReading, Feb. 4, 2008, http://www.darkreading.com/document.asp?doc_id=144919
[36] To control end user web access, some administrators set up a proxy web server and configure every user's browser to send all HTTP requests to the proxy server. The proxy server then forwards the request to its intended destination. This technique gives the administrator one "choke point" for monitoring and, if necessary, curtailing user web surfing. If one of the administrator's users is infected by a Botnet 1.0 client, when the bot client tries to connect back to the C&C, it can't, because the bot's creator didn't plan for the web proxy being in the way. MayDay is smart enough to discover and use the victim's web proxy settings and work through them.

- **Strategically limited C&C traffic**. MayDay contains two new stealth innovations. First, it carefully throttles how much traffic passes between the control center and bot clients so that there is not a telltale, large volume of converging traffic. Second, it enforces a short window wherein communication must happen. When a machine gets infected by MayDay, the C&C starts a timer. If the new bot client does not connect to the C&C within a certain period of time, the new client is not allowed to connect at all. Since researchers and law enforcement would typically try to connect to the C&C on their own arbitrary schedule, the timer halts their efforts. Finally, as of this writing, MayDay was maintaining a mere 1,000 infected clients, which we perceive as an attempt to avoid notice.



MayDay can use any of three different protocols to communicate with peers. But a new member gets only a limited window of time to join the network. If a researcher (1) tries to impersonate a MayDay peer on his own timing, the C&C notes the improper timing (2) and denies the connection (3).

Figure 7. MayDay's P2P Architecture

Trip Cox said that he found MayDay running in the wild, on the networks of at least one Fortune 50 company (whose name he has withheld), some educational institutions, and some Internet Service Providers (ISPs). The infection seems to arrive via email with an attached PDF file.

Coincidentally (or perhaps not), weeks before MayDay was spotted, industry observers noted a wide spam attack of *spear phishing*[37] directed at C-level executives (CEO, CIO, CISO, CFO, etc.). These spam messages purported to be from the Internal Revenue Service informing the executive that his or her corporate taxes had been improperly handled. If the executive opened the PDF to read the supposed "details," instead, a trojan installed itself on the executive's computer. It is sheer speculation to link this PDF attack to the subsequent emergence of MayDay, but such a scenario is too logical to dismiss outright.

What MayDay's creators intend to do with their powerful new tool remains to be seen. So far MayDay has perpetrated a few spam campaigns, which is exactly how Storm began before growing into a behemoth. We'll be keeping an eye on MayDay as 2008 progresses.

Botnets 2.0: No Signs of Slowing

The pace of innovation among bot coders defies description. We had difficulty finding a cutoff point for this white paper because notable innovations appear nearly every week; for example, during the review process for this paper, a bot called Mega-D replaced Storm as the largest spammer in the world. The creativity of today's botmaster is why it is urgent for every network administrator to be familiar with current bot practices.

However, we need not despair. In the face of such relentless and original advances, you might feel that the bad guys are winning. Such is not the case. In the next and final section, we suggest countermeasures that greatly mitigate the likelihood of a bot infection operating from your network.

Defeating the Botnets of the Future

As noted earlier, botnets now embody the ultimate blended threat. Botnet code carries almost every conceivable form of malware, from spyware to downloaders, rootkits, spam engines, and more. To answer like with like, defenders must employ multiple layers of security. The good news is that time-honored techniques are still surprisingly effective against botnets.

Remember that in layered defenses, no one layer needs to be one hundred percent effective. As Fred Avolio pointed out in a LiveSecurity article,

> Defense in depth is powerful. If I have a single security control that is only 50% effective, that sets the stage for disaster down the road. But if I line up two different controls in series, each only 50% effective, I get to 75% effectiveness (the first control catches half the bad stuff, leaving 50%; the second control catches half of that half, leaving only 25% of the bad stuff). If I line up five controls, each just 50% effective, I get to nearly 97% efficiency. To get over 99% we need 4 controls, each 70% effective.[38]

[37] "Phishing" describes techniques where attackers use electronic means to trick victims into divulging confidential information. The classic phishing attack would be an email apparently to you, apparently from PayPal, asking you to verify your account credentials. The email is not really from PayPal, and the link you follow to "sign in" captures your login information for attackers. "Spear phishing" is the term for phishing attacks that are targeted to a specific person or group, usually by customizing the bait so it seems like personal correspondence.

[38] "Defense in Depth," by Fred Avolio, March 2001. LiveSecurity subscription required to access: https://www.watchguard.com/archive/showhtml.asp?pack=1755

By "thinking in 3D" and lining up several controls, administrators can drastically mitigate bot effectiveness. Here are the controls we recommend, and why.

Patch promptly

As described earlier in this paper, bots can draw upon a wide variety of exploits in order to infect victims. However, the biggest and most successful bots have relied upon exploiting vulnerabilities *that the vendor patched six to eighteen months earlier*. In the most extreme cases, we've seen bots attempting exploits against vulnerabilities that were patched as long as four years earlier. We can't account for why bot communications and back-end systems innovate at a breathtaking pace, while the bot uses exploits that are known and old. Our best guess is that botmasters find exploits by waiting for vendors to patch a vulnerability, then reverse-engineering the patch to find out what the flaw was.

We expect that exploiting more recent flaws will be one of the next areas for botmasters to improve upon. But for now, it is good news for the average network administrator. If you patch promptly when vendors release fixes for software you run on your network, you can move faster than the botmasters and resist their exploits.

Block JavaScript

When a bot leveraging web-based exploits attacks a victim computer, it invariably does so by executing JavaScript. Setting browsers to prompt before executing JavaScripts will eliminate a huge swath of bot infection vectors. We highly recommend having users rely on Firefox as their primary browser, using the NoScript plug-in[39] to prompt whenever a script tries to execute.

Watch Those Ports

This is a two-part recommendation.

1) Even though the latest bots can communicate over ports every administrator must leave open, the vast majority of bots still communicate using IRC (port 6667) and other odd, high-numbered ports (such as 31337 and 54321). All ports above 1024 should be set to block both inbound *and outbound* unless your organization has a custom application or special need to open a given port. Even then, you can open a port carefully, implementing policies such as "open only during business hours" or "deny all, except traffic from the following list of trusted IP addresses." This simple measure prevents the garden variety and slow-adopter bots from reaching their C&C for instructions and updates, essentially killing such bots on arrival.

2) Botnet 2.0 traffic that travels over needed ports such as 80 or 7 often gives itself away by generating traffic when there should be none. Commonly, botmasters update their zombies between 1:00 a.m. and 5:00 a.m., when they assume no one is watching. Make a habit of checking your server logs in the morning. If you see web browsing activity when no one was there to do the browsing, that's your cue to investigate.

Administrators using WatchGuard Firebox® models will be pleased to know that the Firebox's proxies stop non-standard traffic attempting to run on standard ports. For example, the spamming botnet Mega-D runs non-standard, homebrew traffic over HTTP port 80. The Firebox's HTTP Proxy would spot and block such traffic instantly, by default.

Redouble user awareness training

Some bots perform mass scans of the Internet, find vulnerable machines, and infect them. This practice is actually diminishing, as Mpack shows. More often now, bots "social engineer" their way onto your network by enticing a victim to click a link or open a file. These bots have the same restrictions as certain legendary vampires: they can't cross your threshold unless you invite them in.

[39] Available at no cost from http://noscript.net/.

This "luring" approach has gradually constricted itself even further. Attackers used to send malicious executable code as an attachment to an email. This practice has also fallen into the minority. Most of the action now is web-based. Malicious emails that would have contained an attachment two years ago, now contain a link to a malicious site instead. Innocuous web sites, as you have read, can be infected by Mpack or other surreptitious malware to infect visitors who arrived by clicking recklessly.

That's your cue to explain to users, in terms they can understand, why they should never invite the vampire in. Tell them not to open attachments that arrive unsolicited and unexpected; why they shouldn't click links in email; and why they must think twice about *any* unusual links they click. If you need a starting point, try circulating our video that shows how drive-by downloads work, described for a non-technical audience: http://video.google.com/videoplay?docid=-4094518401580008932 . While showing this video in training classes, we have literally seen users' jaws drop and eyes bulge in shock at how rapidly malicious scripts work – and those users then changed their behavior.

Diligently applying controls such as those we have cited above have allowed networks to run free of bots for years.

Stay vigilant

This recommendation seems too obvious to mention, almost like "Try not to get infected!" Yet we keep meeting IT administrators who spend so much time putting out fires and maintaining an understaffed help desk, they never look at their system logs. They never monitor bandwidth usage. They can't tell you who is connecting to what from their network. They have devices connected to their network that they don't even know about.

If this describes you, all we can say is, you are begging for trouble. You might even have bots on your network as you read this. If you are an administrator who rarely checks your logs, you must start reading them. Today. Once you learn what "normal" looks like on your network, 30 minutes a day is all you need for a spot check.

If this describes you, the odds are you are not lazy – you are constrained by lack of personnel and resources. Explain the threat to your bosses and see whether they'll support you in blocking out a half hour each morning for checking the status of your network. This time segment should be defended against meeting requests, conference calls, and other typical interruptions. This form of insurance is dirt cheap compared to the cost of a network compromise.

Conclusion

We believe the unprecedented bot breakthroughs of 2007 merely foreshadow innovations to come. As never before in our years in Internet security, each month seems to bring newly discovered exploits *that researchers cannot fully explain*.

It turns out that botnets have been blended threats, but they have not been *ultimate* blended threats. Botmasters now freely supplement the traditional botnet architecture with added components that enhance automation, administration, and evasion. These combinations of technologies are frighteningly sophisticated and surprisingly polished. Any observer can safely predict that this trend will not only continue, but grow.

Are the bad guys winning? Obviously not, since we're still banking and buying over the Internet. But the flood of bot activity is rising, so we must push back, damming up the bot flood and revealing their masters' techniques. In this regard, we view researchers such as Don Jackson and Joe Stewart of SecureWorks, Trip Cox of Damballa, and countless other white hat researchers and teachers as the anti-botnet heroes. If you learn something about how botnets work, share it with the rest of the white hat community. Together we have resisted all attacks thus far. Together we can block the botnets of the future. ##

For information about WatchGuard security solutions and the protection they provide against botnets, visit us at www.watchguard.com, or contact your reseller.

ADDRESS:
505 Fifth Avenue South
Suite 500
Seattle, WA 98104

WEB:
www.watchguard.com

U.S. SALES:
+1.800.734.9905

INTERNATIONAL SALES:
+1.206.613.0895

ABOUT WATCHGUARD

Since 1996, WatchGuard Technologies has provided reliable, easy to manage security appliances to hundreds of thousands of businesses worldwide. Our Firebox X family of unified threat management (UTM) solutions provides the best combination of strong, reliable, multi-layered security with the best ease of use in its class. Our newest product line – the WatchGuard SSL – makes secure remote access easy and affordable, regardless of the size of your network. All products are backed by LiveSecurity Service, a ground-breaking support and maintenance program. WatchGuard is a privately owned company, headquartered in Seattle, Washington, with offices throughout North America, Europe, Asia Pacific, and Latin America. For more information, please visit www.watchguard.com.

Part. No. WGCE66525_042308