Elias Levy, aleph1@securityfocus.com
Iván Arce, ivan.arce@coresecurity.com

A Short Visit to the Bot Zoo

bot: n [common on IRC, MUD, and among gamers; from "robot"]
1. An IRC or MUD user who is actually a program. On IRC, typically the robot provides some useful service. Examples are NickServ, which tries to prevent random users from adopting nicks already claimed by others, and MsgServ, which allows one to send asynchronous messages to be delivered when the recipient signs on.
—*The Jargon File,* version 4.4.7

This past year has seen a new attack trend emerge: bots. After a successful compromise, the attacker installs a bot (also called a zombie or drone) on the system; this small program enables a remote control mechanism to then command the victim. Attackers use this technique repeatedly to form networks of compromised machines (botnets) to further enhance the effectiveness of their attacks.

THORSTEN HOLZ
RWTH Aachen University

A short history of bots

The first bots programs were used in Internet Relay Chat (IRC) networks; they reacted to events in IRC channels and offered services to users. Inappropriate behavior started to evolve around 1993, resulting in the IRC wars that caused the first distributed denial-of-service (DDoS) attacks.

In recent years, malicious bots have become commonplace, with botnets in particular posing a severe threat to the Internet community. Attackers primarily use them for DDoS attacks, mass identity theft, or sending spam. A detailed introduction to botnets, how they work, and who uses them appears elsewhere (see http://honeynet.org/papers/bots/).

Bot characteristics

Three attributes characterize a bot: a remote control facility, the implementation of several commands, and a spreading mechanism to propagate it further. Let's look at each one in more detail.

A remote control lets an attacker manipulate infected machines. Bots currently implement several different approaches for this mechanism:

- Typically, the bots controller uses a central IRC server for command and control (C&C). All bots join a specific channel on this server and interpret all the messages they receive here as commands. This structure is usually secured with the help of passwords to connect to the server, join a specific channel, or issue commands. Several bots also use SSL-encrypted communication.
- In other situations, such as when some bots avoid IRC and use covert communication channels, the controller uses, for example, communication channels via an HTTP or DNS tunnel instead of an inappropriate IRC protocol. They can, for example, encode commands to the bots inside HTTP requests or within DNS TXT records. Another possibility is to hide commands in images (steganography).

- Some bots use peer-to-peer (P2P) communication mechanisms to avoid a central C&C server because it's a single point of failure. Expect to see more bots implement P2P communication similar to the protocol Slapper used.[1]

Typically, two types of commands are implemented over the remote control network: DDoS attacks and updates. DDoS attacks include SYN and UDP flooding or more clever ones such as spidering attacks—those that start from a given URL and follows all links in a recursive way—against Web sites. Update commands instruct the bot to download a file from the Internet and execute it. This lets the attacker issue arbitrary commands on the victim's machine and dynamically enhance the bot's features. Other commands include functions for sending spam, stealing sensitive information from the victim (such as passwords or cookies), or using the victim's computer for other nefarious purposes.

The remote control facility and the commands that can be executed from it differentiate a bot from a *worm*, a program that propagates itself by attacking other systems and copying itself to them. But similar to a worm, most bots also include a mechanism to spread further, usually by automatically scanning whole network ranges and propagating themselves via vulnerabilities. These vulnerabilities usually appear in the Windows operating system, the most common being DCOM (MS03-026, buffer overrun in RPC interface could allow code execution) and LSASS (MS04-011, security update for Microsoft Windows). Attackers also integrate recently published exploits into their bots to react quickly to new trends.

Propagation via network shares and weak passwords on other machines is another common technique. The bot uses a list of passwords and usernames to log on to remote shares and then drops its copy. Some bots propagate by using P2P file-sharing protocols, such as Kazaa and Bear Share; using interesting filenames, the bot drops copies of itself into these programs' shared folders. It generates the filename by randomly choosing from sets of strings.

An additional characteristic applies to bots that the German Honeynet Project captured in the wild: most of them have at least one *executable packer*, a small program that compresses/encrypts the actual binary. Typically, the attacker uses tools such as UPX (http://upx.sourceforge.net/) or Morphine (http://hxdef.czweb.org/download/Morphine27.zip) to pack the executable.

Examples and classification

Let's examine some specific bots in more detail. Table 1 gives a quantitative overview of the evolution of different bot types. It shows that Agobot, the bot that dominated the year 2004, is now less common. In contrast, attackers are increasingly using SDBot, and new variants appear daily.

Agobot and variants

Probably the best-known family of bots is Agobot/Gaobot, and its variants Phatbot (www.lurhq.com/phatbot.html), Forbot, and XtrmBot. The antivirus vendor Sophos currently lists more than 1,100 known different versions of Agobot, and this number is steadily increasing. Agobot's source code was published on various Web sites in April 2004, leading to new variants every week since.

A young German man using the pseudonym Ago first wrote Agobot in 2003; in May 2004, German authorities arrested and charged him with creating malicious computer code under the country's computer sabotage law. The bot is written in C++ with cross-platform capabilities, and it shows a high abstract design. It's structured in a very modular way, which makes it easy to add commands or scanners for other vulnerabilities.

Table 1. New bot variants by month.

MONTH	AGOBOT	SDBOT
May 2004	543	332
June 2004	249	654
July 2004	339	1018
August 2004	133	977
September 2004	123	818
October 2004	158	1111
November 2004	113	1156
December 2004	196	1637
January 2005	227	1539
February 2005	97	2010
March 2005	200	1689

For remote control, this family of bots typically uses a central C&C IRC server. Some variants also use P2P communication via the decentralized WASTE network (http://waste.sourceforge.net/), thus avoiding a central server.

Agobot and its variants use a packet-sniffing library (libpcap) and Perl-compatible regular expressions to sniff and sort network traffic passing through the victim's computer. This malware can use the New Technology File System (NTFS) alternate data stream and offers rootkit capabilities such as file and process hiding to hide its own presence on a compromised host. As an added complication, reverse engineering this malware is difficult because it includes functions to detect debuggers and virtual machines, and it encrypts the configuration in the binary.

On startup, the program attempts to run a speed test for Internet connectivity. By accessing several servers and sending data to them, the bot tries to estimate the victim's available bandwidth. Fortunately, this activity can help us estimate the actual number of hosts compromised by this particular bot: essentially, we look at the log files. If Agobot uses www.belwue.de as one of the domains for a speed test, for example, the domain's administrators can make an

educated guess about the bot's deployment by monitoring how often the speed test is performed. In May 2004, the University of Stuttgart's Computer Emergency Response Team (RUS-CERT) identified approximately 300,000 unique IP addresses per day in this fashion.[2]

This type of malware can also terminate the processes that belong to antivirus and monitoring applications; some variants can even modify the host file (which contains the host-name-to-IP-address mappings). The malware appends a list of Web site addresses—of antivirus vendors, for example—and redirects them to the loopback address, preventing the infected user from accessing the specified location.

SDBot and variants

At the moment, SDBot and its variants RBot, UrBot, UrXBot, and Spybot, are the most active bots in the wild. The whole family is written in C, and literally thousands of different versions exist because the source code is public. SDBot's source code isn't as well designed or written as Agobot's, but it offers similar features, although the command set isn't as large, nor the implementation as sophisticated.

We can see bot evolution through time by looking at this particular family of bots: each new version integrates new features, and each new variant results in major enhancements. Attackers integrate new vulnerabilities quickly, and once one version has a new spreading capability, all the others integrate it immediately. Moreover, small modifications that can implement specific features (such as password encryption within the malware) can be integrated in all variants.

mIRC-based bots

We subsume all mIRC-based bots into the category of GT-bots: so many different versions of them exist that giving an overview of all the forks would be close to impossible. mIRC is a popular IRC client for Windows, and GT is an abbreviation for *global threat*, which is the common name used for all mIRC-scripted bots.

GT-bots launch an instance of the mIRC chat client with a set of scripts and other binaries. One binary we usually find is a `HideWindow` executable that hides the mIRC instance from the user. The other binaries are mainly dynamic link libraries (DLLs) linked to mIRC that add some new features that the mIRC scripts can use to

control the bot. The bots can access the spreading functions in the DLLs and thus enable further propagation.

GT-bots spread by exploiting weaknesses on remote computers and uploading themselves to compromised hosts. One handicap is their large file size—they're sometimes bigger than a megabyte.

Other types of bots

Although some bots aren't as widespread as the ones we've just examined, some of them have interesting features that are worth reviewing.

Xot and its successor XT Bot implement a feature called *dynamic remote settings stub*. DRSS hides the communication flow between attacker and bots by embedding the commands in a file (for example, within an image). The attacker then uploads this file to a server, and the bot on the victim's computer downloads it, extracts the information, and interprets the commands.

The Dataspy Network X bot is written in C++ and has a convenient interface that lets attackers write scanners and spreaders as plugins and extend the bot's features. This bot has a major disadvantage—the default version doesn't come with any spreaders—but plugins are available to overcome this gap. Additional plugins also offer services such as DDoS attacks, portscan interface, or hidden HTTP server.

Bobax uses HTTP requests as its communication channel and thus implements a stealthier remote control than IRC-based C&C. It also implements mechanisms to spread further by downloading and executing arbitrary files. In contrast to other bots, Bobax's primary purpose is to send spam. A detailed analysis of it appears elsewhere (www.lurhq.com/bobax.html).

aIRCBot is very small (only 2,560 bytes); it's not a typical bot because it implements a rudimentary remote control mechanism, and it only understands raw IRC commands. It also completely lacks the functions to spread further. Likewise, Q8Bot and kaiten are small bots, consisting of only a few hundred lines of source code, but they have an additional noteworthiness: they're written for Unix/Linux systems. These programs implement all common bot features: dynamic updating via HTTP-downloads, various DDoS attack capabilities, execution of arbitrary commands, and many more. In the version we've captured, the spreaders are missing, but we assume other versions of these bots have spreaders. Many different versions of simple bots based on the programming language Perl exist, but these bots usually contain only a few hundred lines of source code and offer a rudimentary set of commands (most often just for DDoS attacks). This type of bot is typically used on Unix-based systems.

Bots are constantly evolving: attackers can integrate new vulnerabilities within an incredibly short time span, sometimes in a matter of hours or days. Furthermore, new techniques to hide the communication channel between bot and controller, new remote control mechanisms in the form of P2P communication, and other innovative ideas demonstrate that bots constitute an emerging security concern. The German Honeynet Project's current research focuses on automated ways to collect and analyze malware. We're developing techniques to observe botnets and to learn more about bots. As these threats continue to adapt and change, so too must the security community. ▢

References

1. I. Arce and E. Levy, "An Analysis of the Slapper Worm," *IEEE Security & Privacy*, vol. 1, no. 1, 2003, pp. 82–87.
2. T. Fischer, "Botnetze," *Proc. 12th DFN-CERT Workshop*, DFN-CERT Services, 2005, p. E1–E7.

Thorsten Holz *is a research student at the Laboratory for Dependable Distributed Systems at RWTH Aachen University. His research interests include the practical aspects of secure systems, but he's also interested in more theoretical considerations of dependable systems. Holz is one of the founders of the German Honeynet Project. Contact him at holz@i4.informatik.rwth-aachen.de.*