

Chapter | 4

Denial of Service Attacks

Chapter Objectives

After reading this chapter and completing the exercises, you will be able to do the following:

- Understand how denial of service attacks are accomplished
- Know how certain denial of service attacks work, such as SYN flood, Smurf, and distributed denial of service
- Take specific measures to protect against denial of service attacks
- Know how to defend against specific denial of service attacks

Introduction

By now you are aware, in a general way, of the dangers of the Internet, and you have explored a few basic rules for protection on the Internet. In Chapter 3, “Cyber Stalking, Fraud, and Abuse,” you were introduced to some fraud, stalking, and related crimes. It is now time to become more specific about how attacks on systems are conducted. In this chapter, we will examine one category of attack that might be used to cause harm to a target computer system. This chapter will describe for you, in depth, the workings of the *denial of service (DoS)* attack. This threat is one of the most common attacks on the Internet, so it is prudent for you to understand how it works and how to defend yourself against it. Further, in the exercises at the end of the book, you will practice stopping a DoS attack. In information security, the old adage that “knowledge is power” is not only good advice, but also an axiom upon which to build your entire security outlook.

DoS

As was said in the Introduction, one of the most common and simplest forms of attack on a system is a DoS. This attack does not even attempt to intrude on your system or to obtain sensitive information; it simply aims to prevent legitimate users from accessing the system. This type of attack is fairly easy to execute. The basic concept requires a minimum of technical skill. It is based on the fact that any device has operational limits. For example, a truck can only carry a finite load or travel a finite distance. Computers are no different than any other machine; they, too, have limits. Any computer system, web server, or network can only handle a finite load. A workload for a computer system may be defined by the number of simultaneous users, the size of files, the speed of data transmission, or the amount of data stored. If you exceed any of those limits, the excess load will stop the system from responding. For example, if you can flood a web server with more requests than it can process, it will be overloaded and will no longer be able to respond to further requests (Webopedia, 2004). That is just as true today as it was in 2004. Every technology has limits; if you can exceed those limits, then you can take the system offline. This reality underlies the DoS attack. Simply overload the system with requests, and it will no longer be able to respond to legitimate users attempting to access the web server.

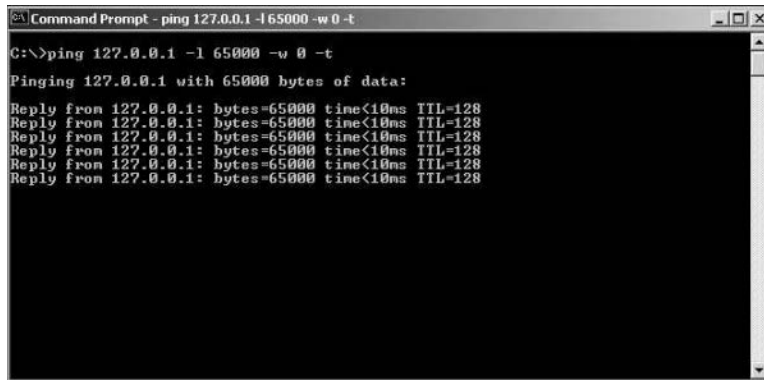
Illustrating an Attack

One simple way to illustrate this attack, especially in a classroom setting, involves the use of the `ping` command discussed in Chapter 2, “Networks and the Internet”:

1. Start a web server service running on one machine. (You can use Apache, IIS, or any web server.)
2. Ask several people to open their browsers and key the IP address of that machine in the address bar. They should then be viewing the default website for that web server.

Now you can do a rather primitive denial of service attack on the system. Recall from Chapter 2 that typing in `ping /?` will show you all the options for the `ping` command. The `-l` option changes the size of the packet you can send. Remember that a TCP packet can be only of a finite size. Thus, you are going to set these packets to be almost as large as you can send. The `-w` option determines how many milliseconds the `ping` utility will wait for a response from the target. You are going to use `-0` so that the `ping` utility does not wait at all. Then the `-t` instructs the `ping` utility to keep sending packets until explicitly told to stop.

1. Open the command prompt in Windows 7/8/8.1/10. (That is the shell in UNIX/Linux.)
2. Type in `ping <address of target machine goes here>-l 65000 -w 0 -t`. You will then see something very much like what is shown in Figure 4.1. Note that, in the figure, I am ping-
ing the loop-back address for my own machine. You will want to substitute the address of the machine on which you are running the web server.



```
Command Prompt - ping 127.0.0.1 -l 65000 -w 0 -t
C:\>ping 127.0.0.1 -l 65000 -w 0 -t
Pinging 127.0.0.1 with 65000 bytes of data:
Reply from 127.0.0.1: bytes=65000 time<10ms TTL=128
Reply from 127.0.0.1: bytes=65000 time<10ms TTL=128
Reply from 127.0.0.1: bytes=65000 time<10ms TTL=128
Reply from 127.0.0.1: bytes=65000 time<10ms TTL=128
Reply from 127.0.0.1: bytes=65000 time<10ms TTL=128
Reply from 127.0.0.1: bytes=65000 time<10ms TTL=128
```

FIGURE 4.1 Ping from the command prompt.

What is happening at this point is that this single machine is continually pinging away at the target machine. Of course, just one machine in your classroom or lab that is simply pinging on your web server is not going to adversely affect the web server. However, you can now, one by one, get other machines in the classroom pinging the server in the same way. After each batch of three or four machines you add, try to go to the web server's default web page. After a certain threshold (certain number of machines pinging the server), it will stop responding to requests, and you will no longer be able to see the web page.

How many machines it will take to deny service depends on the web server you are using. In order to see this denial happen with as few machines involved as possible, you could use a very low-capacity PC as your web server. For example, running an Apache web server on a simple laptop running Windows 7 Home Edition, it can take about 15 machines each running about 10 different command windows simultaneously pinging to cause a web server to stop responding to legitimate requests. This strategy is, of course, counter to what you would normally select for a web server—no real web server would be running on a simple laptop with Windows 7 Home Edition (or even Windows 10). Likewise, actual DoS attacks use much more sophisticated methods. This simple exercise, however, should demonstrate for you the basic principle behind the DoS attack: Simply flood the target machine with so many packets that it can no longer respond to legitimate requests. It is important to be aware that this is just an illustration. With modern servers, and many servers actually being hosted in clusters or server farms, this exact illustration would not work against a modern target.

Generally, the methods used for DoS attacks are significantly more sophisticated than the illustration. For example, a hacker might develop a small virus whose sole purpose is to infect as many computers as possible and then get each of the infected computers to initiate a DoS attack on the target. Once the virus has spread, the various machines that are infected with that virus then begin their flood of the target system. This sort of DoS is easy to do, and it can be hard to stop. A DoS that is launched from several different machines is called a distributed denial of service (DDoS).

Regardless of the methods or the tools (many of which we will describe in this chapter), DoS and DDoS attacks are becoming even more prevalent. According to Akamai research, "DDoS attacks grew

seven percent since the last quarter and a staggering 132 percent compared to this time last year. In the quarter there were also 12 attacks that were categorized as “mega attacks,” peaking at more than 1,000 gigabits per second (Gbps) and 50 million packets per second (Mpps).”¹

Common Tools Used for DoS

As with any of the security issues discussed in this book, you will find that hackers have at their disposal a vast array of tools with which to work. The DoS arena is no different. While it is certainly well beyond the scope of this book to begin to categorize or discuss all of these tools, a brief introduction to just a few of them will prove useful. The two tools discussed here, TFN and Stacheldraht, are typical of the types of tools that someone wishing to perform a DoS attack would utilize.

Low Orbit Ion Cannon

This is one of the most widely known DoS tools available. It has a very easy to use graphical user interface, shown in Figure 4.2.

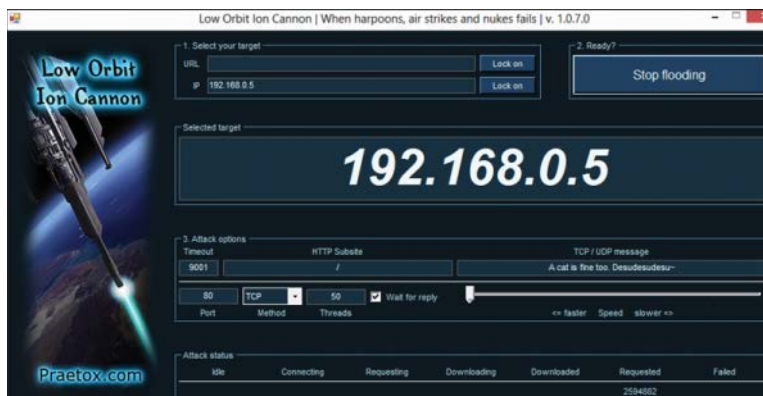


FIGURE 4.2 LOIC.

This tool is very easy to use. As you can see in Figure 4.2, it simply requires the user to put in the target URL or IP address and then begin the attack. Fortunately, this tool also does nothing to hide the attacker’s address and thus makes it relatively easy to trace the attack back to its source.

XOIC

This tool is similar to LOIC. It has three modes. You can send a message, execute a brief test, or start a DoS attack. You can see these options in Figure 4.3.

1. <http://www.digitaltrends.com/computing/ddos-attacks-hit-record-numbers-in-q2-2015/>

- Both master-to-agent communications and the attacks themselves can be sent via randomized TCP, UDP, and ICMP packets.
- The master can falsify its IP address (spoof).

Stacheldraht

This tool is not as widely known as the previously mentioned DoS tools. Stacheldraht, which is German for “barbed wire,” is a DDoS attack tool that combines features of the Trinoo DDoS tool (another common tool) with the source code from the TFN DDoS attack tool. Like TFN2K, it adds encryption of communication between the attacker and the Stacheldraht masters. It also adds an automatic updating of the agents.

Stacheldraht can perform a variety of attacks including UDP flood, ICMP flood, TCP SYN flood, and Smurf attacks. It also detects and automatically enables source address forgery.

DoS Weaknesses

The weakness in any DoS attack, from the attacker’s point of view, is that the flood of packets must be sustained. As soon as the packets stop sending, the target system is back up. A DoS/DDoS attack, however, is very often used in conjunction with another form of attack, such as disabling one side of a connection in TCP hijacking or preventing authentication or logging between servers.

If the hacker is using a distributed attack, as soon as the administrators or owners of the infected machines realize their machine is infected, they will take steps to remove the virus and thus stop the attack. If a hacker attempts to launch an attack from her own machine, she must be aware that each packet has the potential to be traced back to its source. This fact means that a single hacker using a DoS will almost certainly be caught by the authorities. For this reason, the DDoS is quickly becoming the most common type of DoS attack. The specifics of DDoS attacks will be discussed later in this chapter.

Specific DoS Attacks

The basic concept for perpetrating a DoS attack is not complicated. The real problem for the attacker is performing the attack without being caught. The next few sections of this chapter will examine some specific types of DoS attacks and look at specific case studies. This information should help you gain a deeper understanding of this particular Internet threat.

TCP SYN Flood Attack

One popular version of the DoS attack is the SYN flood. This particular attack depends on the hacker’s knowledge of how connections are made to a server. When a session is initiated between the client and server in a network using the TCP protocol, a packet is sent to the server with a 1-bit flag called a SYN flag set. SYN is short for synchronize. And this packet is asking the target server to please synchronize communications. The server will then allocate appropriate resources and then send to

the client a packet with both the SYN (synchronize) and the ACK (acknowledge) flags set. The client machine is then supposed to respond with an ACK flag set. This is called the three-way handshake and is summarized as follows:

1. The client sends a packet with the SYN flag set.
2. The server allocates resources for the client and then responds with the SYN and ACK flags set.
3. The client responds with the ACK flag set.

There have been a number of well-known SYN flood attacks on web servers. The reason for the popularity of this attack type is that any machine that engages in TCP communication is vulnerable to it—and all machines connected to the Internet engage in TCP communications. Such communication is obviously the entire reason for web servers. The easiest way to block DoS attacks is via firewall rules. We will discuss firewalls in detail in Chapter 9, “Computer Security Technology.” A properly configured firewall can prevent the SYN flood attack. There are, however, several methods and techniques you can implement on individual servers to protect against these attacks. The basic defensive techniques are as follows:

- Micro blocks
- SYN cookies
- RST cookies
- Stack tweaking

Some of these methods require more technical sophistication than others. These methods will be discussed in general here. When you are entrusted with defending a system against these forms of attacks, you can select the methods most appropriate for your network environment and your level of expertise and examine it further at that time. The specifics of how to implement any of these methods will depend on the operating system that your web server is using. You will need to consult your operating system’s documentation, or appropriate websites, in order to find explicit instruction on how to implement methods.

Micro Blocks

A *micro block* works by simply allocating a micro-record instead of allocating a complete connection object (an entire buffer segment) to the SYN object. In this way, an incoming SYN object can allocate as little as 16 bytes of space, making it significantly more difficult to flood a system. This method is a bit more obscure and not as widely used today as it once was. It also does not actually prevent a DoS attack; it merely mitigates the effects.

SYN Cookies

This is another mitigation method, just like micro blocks. It should also be noted that many network administrators simply depend on their firewall to block DoS attacks and don’t take any remediation

steps on individual servers. I suggest you at least consider combining both approaches. Yes, have a well-configured firewall to block many DoS attacks, but also consider mitigating steps that can be taken on individual servers.

As the name *SYN cookies* suggests, this method uses cookies, not unlike the standard cookies used on many websites. With this method, the system does not immediately create a buffer space in memory for the handshaking process. Rather, it first sends a *SYN+ACK* (the acknowledgment signal that begins the handshaking process). The *SYN+ACK* contains a carefully constructed cookie, generated as a hash that contains the IP address, port number, and other information from the client machine requesting the connection. When the client responds with a normal *ACK* (acknowledgment), the information from that cookie will be included, which the server then verifies. Thus, the system does not fully allocate any memory until the third stage of the handshaking process. This enables the system to continue to operate normally; typically, the only effect seen is the disabling of large windows. However, the cryptographic hashing used in *SYN cookies* is fairly resource intensive, so system administrators that expect a great deal of incoming connections may choose not to use this defensive technique.

FYI: Hashing

A hash value is a number generated from a string of text. The hash is significantly smaller than the text itself and is generated by a formula in such a way that it is extremely unlikely that some other text will produce the same hash value. Hashing plays a role in security when it is used to ensure that transmitted messages have not been tampered with. To do this, the sender generates a hash of the message, encrypts it, and sends it with the message itself. The recipient then decrypts both the message and the hash, produces another hash from the received message, and compares the two hashes. If they are the same, there is a very high probability that the message was transmitted intact. We will discuss hashing in more detail in Chapter 8, “Encryption.”

RST Cookies

Another cookie method that is easier to implement than *SYN cookies* is the *RST cookie*. In this method, the server sends a wrong *SYN+ACK* back to the client. The client should then generate an *RST* packet telling the server that something is wrong. Because the client sent back a packet notifying the server of the error, the server now knows the client request is legitimate and can now accept incoming connections from that client in the normal fashion. This method has two disadvantages. It might cause problems with older Windows machines or machines that are communicating from behind firewalls.

Stack Tweaking

The method of *stack tweaking* involves altering the TCP stack on the server so that it will take less time to time out when a *SYN* connection is left incomplete. Unfortunately, this protective method will just make executing a *SYN* flood against that target more difficult; to a determined hacker, the attack is still possible.

In Practice**Stack Tweaking**

The process of stack tweaking is often quite complicated, depending on the operating system. Some operating systems' documentation provides no help on this subject. For these reasons, this method is usually only used by very advanced network administrators and is not recommended unless you have a very solid knowledge of the operating system with which you are working.

Smurf IP Attack

The Smurf attack is a very popular version of the DoS attack. An ICMP (Internet Control Message Protocol) packet is sent out to the broadcast address of the network. Since it is broadcast, it responds by echoing the packet out to the network hosts, who then send it to the spoofed source address. Also, the spoofed source address can be anywhere on the Internet, not just on the local subnet. If the hacker can continually send such packets, she will cause the network itself to perform a DoS attack on one or more of its member servers. This attack is clever and rather simple. The only problem for the hacker is getting the packets started on the target network. This task can be accomplished via some software, such as a virus or Trojan horse, that will begin sending the packets.

In a Smurf attack, there are three people/systems involved: the attacker, the intermediary (who can also be a victim), and the victim. The attacker first sends an ICMP echo request packet to the intermediary's IP broadcast addresses. Since this is sent to the IP broadcast address, many of the machines on the intermediary's network will receive this request packet and will send an ICMP echo reply packet back. If all the machines on a network are responding to this request, the network becomes congested and there can be outages.

The attacker impacts the third party—the intended victim—by creating forged packets that contain the spoofed source address of the victim. Therefore, when all the machines on the intermediary's network start replying to the echo request, those replies will flood the victim's network. Thus, another network becomes congested and could become unusable. This attack is illustrated in Figure 4.4.

The Smurf attack is an example of the creativity that some malicious parties can employ. It is sometimes viewed as the digital equivalent of the biological process in an auto-immune disorder. With such disorders, the immune system attacks the patient's own body. In a Smurf attack, the network performs a DoS attack on one of its own systems. This method's cleverness illustrates why it is important that you attempt to work creatively and in a forward-thinking manner if you are responsible for system security in your network. The perpetrators of computer attacks are inventive and always coming up with new techniques. If your defense is less creative and clever than the attackers' defense, then it is simply a matter of time before your system is compromised.

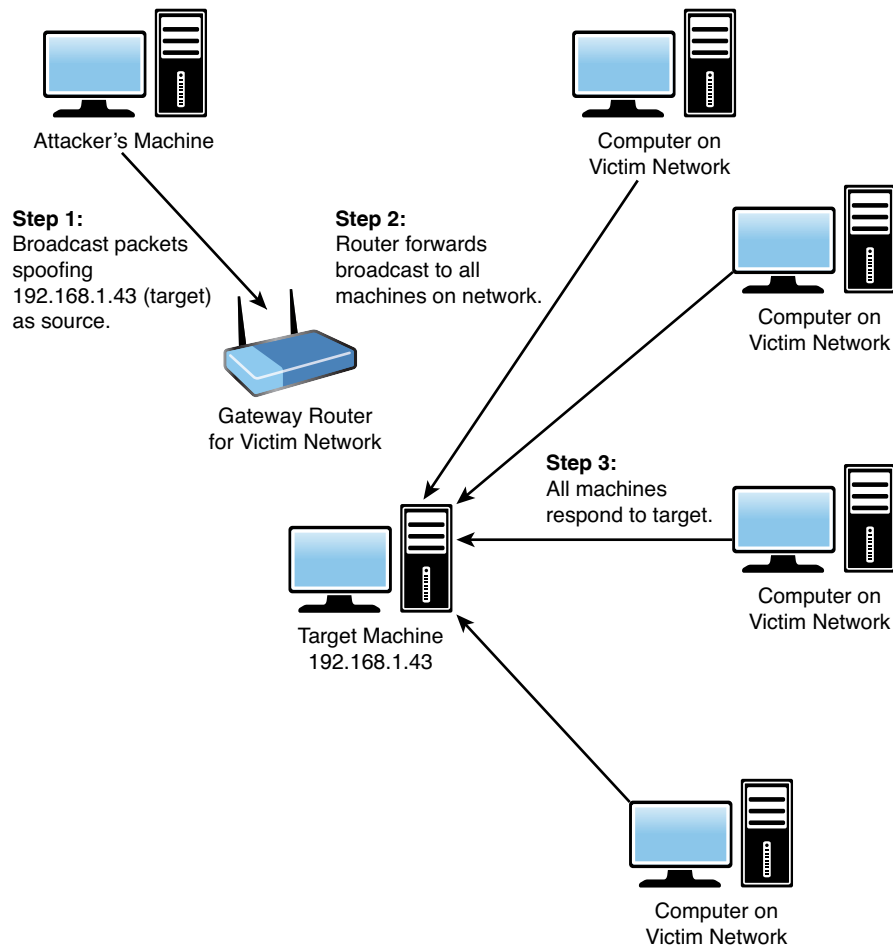


FIGURE 4.4 Smurf

There are several ways to protect your system against this problem. One is to guard against Trojan horses. More will be said about the Trojan horse attack in later chapters; however, having policies prohibiting employees from downloading applications will help. Also, having adequate virus scanners can go a long way in protecting your system from a Trojan horse and, thus, a Smurf attack. It is also imperative that you use a proxy server, which was discussed in Chapter 2. If the internal IP addresses of your network are not known, then it is more difficult to target one in a Smurf attack. And, of course, the most obvious mitigation step you can take is to block all inbound broadcast packets at the firewall. Probably the best way to protect your system is to combine these defenses along with prohibiting directed broadcasts and patching the hosts to refuse to reply to any directed broadcasts.

UDP Flood Attack

UDP, as you will recall from Chapter 2, is a connectionless protocol that does not require a connection setup procedure prior to transferring data. In a *UDP flood attack*, the attacker sends a UDP packet to a random port on a target system. When the target system receives a UDP packet, it automatically determines what application is waiting on the destination port. In this case, since there is no application waiting on the port, the target system will generate an ICMP packet of “destination unreachable” and attempt to send it back to the forged source address. If enough UDP packets are delivered to ports on the target, the system will become overloaded trying to determine awaiting applications (which do not exist) and then generating and sending packets back.

ICMP Flood Attack

There are two basic types of *ICMP flood attacks*: floods and nukes. An ICMP flood is usually accomplished by broadcasting a large number of either pings or UDP packets. Like other flood attacks, the idea is to send so much data to the target system that it slows down. If it can be forced to slow down enough, the target will time out (not send replies fast enough) and be disconnected from the Internet. ICMP nukes exploit known bugs in specific operating systems. The attacker sends a packet of information that he knows the operating system on the target system cannot handle. In many cases, this will cause the target system to lock up completely.

This attack is far less effective against modern computers. Even a low-end desktop PC now will have 4 gigabytes (or more) of RAM and a dual core processor. That makes it difficult to generate enough pings to knock the machine offline. However, at one time this was a very common form of DoS attack.

The Ping of Death

Recall from Chapter 2 that TCP packets are of limited size. In some cases, simply sending a packet that is too large can shut down a target machine. This action is referred to as the *ping of death (PoD)*. It works simply by overloading the target system. The hacker sends merely a single ping, but he does so with a very large packet and thus can shut down some machines.

This attack is quite similar to the classroom example discussed earlier in this chapter. The aim in both cases is to overload the target system and cause it to quit responding. PoD works to compromise systems that cannot deal with extremely large packet sizes. If successful, the server will actually shut down completely. It can, of course, be rebooted.

The only real safeguard against PoD is to ensure that all operating systems and software are routinely patched. This attack relies on vulnerabilities in the way a particular operating system (or application) handles abnormally large TCP packets. When such vulnerabilities are discovered, it is customary for the vendor to release a patch. The possibility of PoD is one reason, among many, why you must keep patches updated on all of your systems.

Teardrop Attack

In a *teardrop attack*, the attacker sends a fragmented message. The two fragments overlap in ways that make it impossible to reassemble them properly without destroying the individual packet headers.

Therefore, when the victim attempts to reconstruct the message, the message is destroyed. This causes the target system to halt or crash. There are a number of variations on the basic teardrop attack that are available, such as TearDrop2, Boink, targa, Nestea Boink, NewTear, and SYNdrop.

Land Attack

A *land attack* is probably the simplest in concept. The attacker sends a forged packet with the same source IP address and destination IP address (the target's IP address). The method is to drive the target system "crazy" by having it attempt to send messages to and from itself. The victim system will often be confused and will crash or reboot. More modern computers are not susceptible to this attack, but it is presented here just for historical purposes.

DDoS

Perhaps the most common form of DoS today is the *DDoS*. This is accomplished by getting various machines to attack the target. A typical way this is done is by sending out a Trojan horse that will cause infected computers to attack a specified target at a particular date and time. This is a very effective way to execute a DDoS on any target. In this form of DDoS, the attacker does not have direct control of the various machines used in the attack. These machines are simply infected by some malware that causes them to participate in the attack on a particular date and time.

Another method is to use a botnet to orchestrate the attack. *Botnets* are networks of computers that have been compromised by the attacker, giving said attacker control of the infected system. This is often accomplished via delivery of a Trojan horse. However, unlike the previous DDoS example, the attacker will have direct control of the attacking machines in the botnet.

Real-World Examples

A good deal of time has been spent discussing the basics of how various DoS attacks are conducted. By now, you should have a firm grasp of what a DoS attack is and have a basic understanding of how it works. It is now time to begin discussing specific, real-world examples of such attacks. This section will take the theoretical knowledge you have gained and give you real-world examples of its application.

MyDoom

One of the most well-publicized DoS attacks was the MyDoom attack. While this attack is a few years old, it is an excellent one to study as it is a classic example of a virus being used to propagate a DDoS attack. This threat was a classically distributed DoS attack. The virus/worm would email itself to everyone in your address book and then, at a preset time, all infected machines would begin a coordinated attack on www.sco.com (Network World, 2004). Estimates put the number of infected machines between 500,000 and 1 million. This attack was successful and promptly shut down the SCO website. It should be noted that well before the day that the DoS attack was actually executed, network administrators and home users were well aware of what MyDoom would do. There were also several tools

available free of charge on the Internet for removing the virus/worm. However, it appears that many people did not take the steps necessary to clean their machines of this virus/worm.

What is interesting is that MyDoom was still causing problems many years after its discovery. As late as July 2009, there was a DoS that utilized MyDoom as a backdoor to launch a DoS attack against South Korean and United States web servers.

FYI: Virus or Worm?

Definitions of the terms *virus* and *worm* are widely debated among the experts. And, depending upon the definition, what some would call a virus, others would call a worm. One general distinction that is accepted by many is that worms do not require direct human interaction to propagate, whereas viruses do. If you accept this definition, then both MyDoom and Slammer are worms. To avoid confusion on this issue, however, the term *virus/worm* will be used.

One thing that makes this attack so interesting is that it is clearly an example of domestic cyber terrorism (although it is certain that the creators of MyDoom would probably see it differently). (Cyber terrorism will be discussed further in Chapter 12, “Cyber Terrorism and Information Warfare.”) For those readers who do not know the story, it will be examined here briefly. Santa Cruz Operations (SCO) makes a version of the UNIX operating system. Like most UNIX versions, their version is copyright protected. Several months before this attack, SCO began accusing certain Linux distributions of containing segments of SCO UNIX code. SCO sent demand letters to many Linux users demanding license fees. Many people in the Linux community viewed this request as simply an attempt to undermine the growing popularity of Linux, an open-source operating system. SCO went even further and filed suit against major companies that were distributing Linux (Software Patent Workgroup, 2003). This claim by SCO seemed unfounded to many legal and technology analysts. It was also viewed with great suspicion because SCO had close ties to Microsoft, which had been trying desperately to stop the growing popularity of Linux.

Many analysts feel that the MyDoom virus/worm was created by some individual (or group of individuals) who felt that the Santa Cruz Operations tactics were unacceptable. The hackers wished to cause economic harm to SCO and damage its public image. This probable motive makes this case clearly one of domestic economic terrorism: One group attacks the technological assets of another group based on an ideological difference. Prior to this virus/worm, there were numerous website defacements and other small-scale attacks that were part of ideological conflicts. However, this virus/worm was the first such attack to be so widespread and successful. This incident began a new trend in information warfare. As technology becomes less expensive and the tactics more readily available, you can expect to see an increase in this sort of attack in the coming years.

Anonymous Uses DoS

On December 8, 2010, the infamous hacker group Anonymous launched multiple DDoS attacks on various financial companies, including Mastercard.com, PayPal, Visa.com, and PostFinance. These sites

were brought down for over 16 hours. The attacks were launched because these sites refused to process donations for Wikileaks. The group Anonymous is a supporter of Wikileaks founder Julian Assange.

DDoS Blackmail

In November 2015, the Australian company FastMail was the victim of a DDoS attack. First the system was attacked and knocked offline. After the second attack, the victim received a ransom demand. The attackers demanded 20 bit coins to call off the attack. A similar attack had been previously launched against Protonmail, also demanding ransom to stop the attacks.

How to Defend Against DoS Attacks

There is no guaranteed way to prevent all DoS attacks, just as there is no sure way to prevent a hacking attack. However, there are steps you can take to minimize the danger. Some methodologies, such as SYN cookies and RST cookies, have already been mentioned. In this section, a few of the steps you can take to make your system less susceptible to a DoS attack will be examined.

One of the first things for you to consider is how these attacks are perpetrated. They may be executed via ICMP packets that are used to send error messages on the Internet or are sent by the `ping` and `tracert` utilities. If you have a firewall (and you absolutely should have one), then simply configuring it to refuse ICMP packets from outside your network will be a major step in protecting your network from DoS attacks. Since DoS/DDoS attacks can be executed via a wide variety of protocols, you can also configure your firewall to disallow any incoming traffic at all, regardless of what protocol or port it occurs on. This step may seem radical, but it is certainly a secure one.

In Practice

Blocking ICMP Packets

There are very few legitimate reasons (and, some would argue, no good reasons) for an ICMP packet from outside your network to enter your network. Thus, blocking such packets is very often used as one part of the strategy to defend against DoS attacks. Incidentally, that will also make it more difficult for an attacker to scan your network (as we will see in Chapter 12).

It is also possible to detect some threats from certain DoS tools, such as TFN2K, by using information tools like NetStat. Many of these tools can be configured to look for the `SYN_RECEIVED` state, which could indicate a SYN flood attack.

If your network is large enough to have internal routers, then you can configure those routers to disallow any traffic that does not originate with your network. In that way, should packets make it past your firewall, they will not be propagated throughout the network. You should also consider disabling directed IP broadcasts on all routers. This strategy will prevent the router from sending broadcast packets to all machines on the network, thus stopping many DoS attacks. Additionally, you can install

a filter on the router to verify that external packets actually have external IP addresses and that internal IPs have internal IP addresses.

Because many distributed DoS attacks depend on “unwitting” computers being used as launch points, one way to reduce such attacks is to protect your computer against virus attacks and Trojan horses. This problem will be discussed in more detail in a later chapter, but for now, it is important that you remember three things:

- Always use virus-scanning software and keep it updated.
- Always keep operating system and software patches updated.
- Have an organizational policy stating that employees cannot download anything onto their machines unless the download has been cleared by the IT staff.

As previously stated, none of these steps will make your network totally secure from either being the victim of a DoS attack or being the launch point for one, but they will help reduce the chances of either occurring. A good resource for this topic is the SANS Institute website, at www.sans.org/dosstep/. This site has some good tips on how to prevent DoS attacks.