Different Types of Attacks

**Types of Attacks**

images Describe the most common network attacks, including session hacking, virus attacks, Trojan horses, denial of service, and buffer overflow.

images Explain how these attacks are executed.

images Identify basic defenses against those attacks.

images Configure a system to prevent denial of service attacks.

images Configure a system to defend against Trojan horse attacks.

images Configure a system to defend against buffer overflow attacks.

**INTRODUCTION**

[Chapter 1](https://learning.oreilly.com/library/view/network-defense-and/9780134893112/ch01.xhtml#ch01), “Introduction to Network Security,” introduced some general dangers to computer systems and provided an overview of network security. This chapter examines specific types of attacks much more closely. This chapter analyzes how systems are most commonly attacked. Particular attention will be paid to the denial of service (DoS) attack. This threat is one of the most common attack methods on the Internet, so understanding how it works and how to defend systems against them is prudent for administrators.

This chapter also describes virus attacks, Trojan horse attacks, and some less common methods of attack, such as session hacking and tunneling. In information security, the old adage “knowledge is power” is not only good advice but also an axiom upon which to build an entire security outlook.

**UNDERSTANDING DENIAL OF SERVICE ATTACKS**

The first type of attack to examine is the denial of service (DoS). Recall from [Chapter 1](https://learning.oreilly.com/library/view/network-defense-and/9780134893112/ch01.xhtml#ch01) that a denial of service attack is any attack that aims to deprive legitimate users of the use of the target system. This class of attack does not actually attempt to infiltrate a system or to obtain sensitive information. It simply aims to prevent legitimate users from accessing a given system. This type of attack is one of the most common categories of attack. Many experts feel that it is so common because most forms of denial of service attacks are fairly easy to execute. The ease with which these attacks can be executed means that even attackers with minimal technical skills can often successfully perform a denial of service.

The concept underlying the denial of service attack is based on the fact that any device has operational limits. This fact applies to all devices, not just computer systems. For example, bridges are designed to hold weight up to a certain limit, aircraft have limits on how far they can travel without refueling, and automobiles can only accelerate to a certain point. All of these various devices share a common trait: They have set limitations to their capacity to perform work. Computers are no different from these, or any other machine; they, too, have limits. Any computer system, web server, or network can only handle a finite load.

How a workload (and its limits) is defined varies from one machine to another. A workload for a computer system might be defined in a number of different ways, including by the number of simultaneous users, the size of files, the speed of data transmission, or the amount of data stored. Exceeding any of these limits 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. 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.

**DoS in Action**

The concept of a denial of services attack is simple; however, most principles are easier to grasp if one can see a concrete example. In this case you need a safe way to simulate a DoS attack within a classroom or laboratory setting. One simple way to illustrate a DoS attack, especially in a classroom setting, involves the use of the ping command along with certain parameters. (Recall that typing in ping**/h** or **ping /?** will show you all the options for the ping command.) The first step is to start a web server service running on a computer that will be used as the target for this attack. You can use any operating system and any web server you like (such as Microsoft Internet Information Services or Apache HTTP Server). Apache is a free download from [www.apache.org](http://www.apache.org/). Microsoft Windows 10 comes with Internet Information Services, so you should have no trouble finding a web server to install and run. For the purposes of this lab you would want to purposefully use a low-capacity machine. An older machine, perhaps an older laptop, would be ideal. You want to pick a machine that will be easy to overload. In essence you are looking for the exact opposite of what you look for when setting up a real web server.

Setting up a web server is actually quite simple. Because Apache is available as a free download for both Linux and Windows ([www.apache.org](http://www.apache.org/)), let’s examine it. Follow these steps to install and configure Apache on your system.

**Installing for Windows**

**1.** Download Apache for Windows from [www.apache.org](http://www.apache.org/).

**2.** Look in C:\Program Files\Apache Group\Apache2\conf for the httpd.conf file and open it.

**3.** Set the ServerName = localhost.

**4.** Save the file.

**5.** From the command prompt type **httpd start**.

You should now be able to open a browser and see the default Apache website.

**Installing for Linux**

**1.** Download Apache for Linux (or in many Linux distributions you can simply add the Apache web server package) from [www.apache.org](http://www.apache.org/).

**2.** Look in /etc/httpd/conf for the httpd.conf file. When you find it, right-click it and open it with a text editor.

**3.** Set the ServerName = localhost.

**4.** Save the file.

**5.** From a shell, type **/etc/init.d/httpd start**. The server should start and you get an OK message.

**6.** Open your browser and go to <http://localhost/>.

You should see the Apache default website.

When you are ready to make your server live (reachable from other PCs), in the /etc/httpd/conf/httpd.conf file change the following settings:

**FYI: Changing the Configuration File**

Any time you change the configuration file you must stop the Apache server and restart it. To stop Apache, use **/etc/init.d/http stop**.

images Change servername to your registered URL or to your IP:port such as 10.10.10.117:80.

images Change listen to reflect the IP and port you want (there is an example in the config file).

images Check the documentroot directory to make certain that is where you want your web pages to be served up from. The default should be /var/www/html.

images From a shell type **/etc/init.d/httpd start**.

**FYI: Conducting Labs Safely**

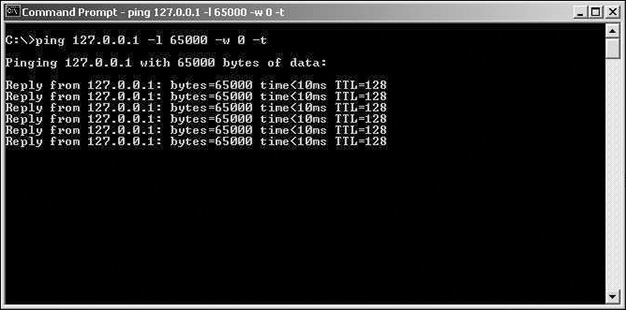
This lab is potentially dangerous to the target server. In fact, throughout this book, we examine labs that could cause significant disruption to live computer systems. You should conduct these labs only on computers that are disconnected from your main network and that do not contain any critical information. Setting up a lab specifically for security practice is best.

If you are using Windows 7 or 2008 or 2012 Server editions you can also choose to use Microsoft Internet Information Services as your web server.

The next step is to verify that the web server is actually running and that you can reach its default web page. One person in the class can open his or her browser and type the target server machine’s IP address in the address bar. He should then be viewing the default website for that web server. Now you can do a rather primitive DoS attack on it.

You make the actual attack using the ping command. If you don’t recall how to use the ping command, you should note that typing **ping /h** at the command prompt displays all the options for the ping command. The options to use in this exercise are -w and -t. The -w option determines how many milliseconds the ping utility will wait for a response from the target. In this case, set that option to -0, so it does not wait at all. The -t option instructs the ping utility to keep sending packets until explicitly told to stop. An additional option, the -l option, allows users to change the size of the packet you can send. Keep in mind that a TCP packet can only be of a finite size, so you are going to set these packets to be almost as large as you can send.

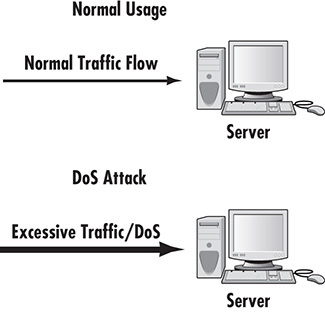
At the command prompt in Windows 10 (that’s the shell in Unix/Linux), type **ping**<*address of target machine goes here***>-l 65000 -w 0 -t**. The machine’s response should be similar to that shown in [Figure 2-1](https://learning.oreilly.com/library/view/network-defense-and/9780134893112/ch02.xhtml#ch2Fig1). Note that in the figure I am pinging the loopback address for my own machine. You will want to substitute the address of the machine on which you are running the web server.



**FIGURE 2-1** Ping from the command prompt

What is happening as this series of pings is being executed is that this single machine is continually pinging away at the target machine. At this point in the exercise, having just one machine in a classroom or lab pinging on a web server should not adversely affect the web server. This is because that level of traffic is well within the capacity of the target web server. However, after causing other machines to ping the server in the same way, you will begin to tax the target machine’s capacity. If you get enough machines pinging the target, you will eventually reach a threshold at which the target machine will stop responding to requests, and you will no longer be able to access the web page. The number of machines it will take to reach this threshold depends on the web server you are using. This author has conducted this particular experiment in classrooms. In those situations Apache web server was being run on a Pentium III laptop running Windows 7, with only 1 gigabyte. In that scenario it only took about 25 machines simultaneously pinging to cause the web server to stop responding to legitimate requests. Even if this experiment does not bring down the machine, it will at least cause it to respond more slowly.

This experiment allows you to get a feel for how a denial of service is executed. It is meant to give you a better understanding of the principle behind the DoS. You should keep in mind that actual denial of service attacks use much more sophisticated methods. Also note that no real web server would be running on a simple laptop with Windows 7. However, this exercise demonstrates 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. This basic concept is shown in [Figure 2-2](https://learning.oreilly.com/library/view/network-defense-and/9780134893112/ch02.xhtml#ch2Fig2). What we have done, in this experiment, is simply to exceed the operational limits of the laboratory web server.



**FIGURE 2-2** The DoS concept

Generally, the methods used for DoS attacks are significantly more sophisticated than the illustration. Although all DoS attacks seek to overload the target machine, a variety of ways exist to do that, and a variety of ways exist to initiate the attack itself. For example, a hacker might develop a small virus whose sole purpose is to initiate a ping flood against a predetermined target. After the virus has spread, the various machines that are infected with that virus begin their ping flood to the target system. This sort of DoS is easy to do and can be hard to stop. A few common DoS attacks are described later in this chapter.

A DoS attack that is launched from several different machines is called a distributed denial of service, or DDoS.

The DDoS is becoming more common; in fact it is now the most common sort of DoS attack. Most of the real-world examples we will examine later in this chapter are DDoS attacks. Two reasons that this form of denial of service attack is becoming more popular include:

images Overloading a target system is easier to do if you have more than one machine attacking it. With newer servers capable of handling much higher workloads, executing a DoS attack from just one machine becomes more difficult.

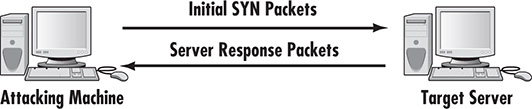
images It allows the attacker to launch the attack from other people’s machines, thus protecting his anonymity. Launching an attack from one’s own machines can be risky because each packet has the potential to be traced back to its source. This would mean an almost certainty of being caught by the authorities.

The basic concept behind a DoS attack is simple. The real problem for the attacker is avoiding being caught. The next section examines some specific types of DoS attacks and reviews specific case studies.

**SYN Flood**

Simply sending a flood of pings is the most primitive method of performing a DoS. More sophisticated methods use specific types of packets. 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 small buffer space in memory is set aside on the server to handle the “hand-shaking” exchange of messages that sets up the session. The session-establishing packets include a SYN field that identifies the sequence in the message exchange.

A SYN flood attempts to subvert this process. In this attack an attacker sends a number of connection requests very rapidly and then fails to respond to the reply that is sent back by the server. In other words, the attacker requests connections, and then never follows through with the rest of the connection sequence. This has the effect of leaving connections on the server half open, and the buffer memory allocated for them is reserved and not available to other applications. Although the packet in the buffer is dropped after a certain period of time (usually about three minutes) without a reply, the effect of many of these false connection requests is to make it difficult for legitimate requests for a session to get established, as shown in [Figure 2-3](https://learning.oreilly.com/library/view/network-defense-and/9780134893112/ch02.xhtml#ch2Fig3).



**FIGURE 2-3** The SYN flood

A number of well-known SYN flood attacks have occurred 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. The TCP packet exchange is the entire basis for web server communication. However, several methods protect against these attacks. Some of those methods require more technical sophistication than others. You can select the methods most appropriate for your network environment and your level of expertise.

**Defending with Micro Blocks**

Micro blocks seek to avoid SYN floods by changing the way the server allocates memory for any given connection request. Instead of allocating a complete connection object, the server is altered so that it only allocates a micro-record. Newer implementations of this technique allocate as little as 16 bytes for the incoming SYN object. The specifics on how to set up micro blocks are specific to a given operating system. This is a less common technique. Many network administrators are not even aware of this possibility.

**Defending with Bandwidth Throttling**

A common method to defend against DoS attacks is for the firewall or intrusion detection system to detect excessive traffic from one or more IP addresses and restrict that bandwidth. This is using bandwidth throttling to mitigate the DoS attack.

**Defending with SYN Cookies**

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 hand-shaking process. Rather, it first sends a SYNACK (the acknowledgment signal that begins the hand-shaking process). The SYNACK 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 hand-shaking process. However, the cryptographic hashing used in SYN cookies is fairly intensive, so system administrators who expect a large number of incoming connections might choose not to use this defensive technique. This is also less common, but not as uncommon as micro blocks.

This defense mechanism also illustrates the fact that most defenses require some tradeoff between performance and security. The overhead resources required by the SYN cookie might degrade performance, especially when there is a large amount of traffic; however, the SYN cookie is one of the more robust defenses against many forms of DoS. The optimal solution would be to have a very high-performance server (or server farm) that can handle the overhead and to implement SYN cookies.

**FYI: Stateful Packet Inspection (SPI)**

Implementing a firewall that examines not just individual packets but the entire “conversation” is one of the easiest ways to stop a SYN flood. Such stateful packet inspection (SPI) firewalls look at all the packets from a given source. Thus, thousands of SYN packets from a single IP address without corresponding SYNACK packets would appear suspicious and be blocked.

**Defending with 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 SYNACK back to the client. The client should then generate an RST (reset) 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 will now accept incoming connections from that client in the normal fashion. This method has two disadvantages. First, it might cause problems with some earlier Windows machines and/or machines that are communicating from behind firewalls. Also, some firewalls might block the return SYNACK packet.

**Defending with Stack Tweaking**

The method of stack tweaking involves altering the TCP stack on the server so that it will take less time to timeout 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. Stack tweaking is more complicated than the other methods and is discussed more fully in [Chapter 8](https://learning.oreilly.com/library/view/network-defense-and/9780134893112/ch08.xhtml#ch08), “Operating System Hardening.”

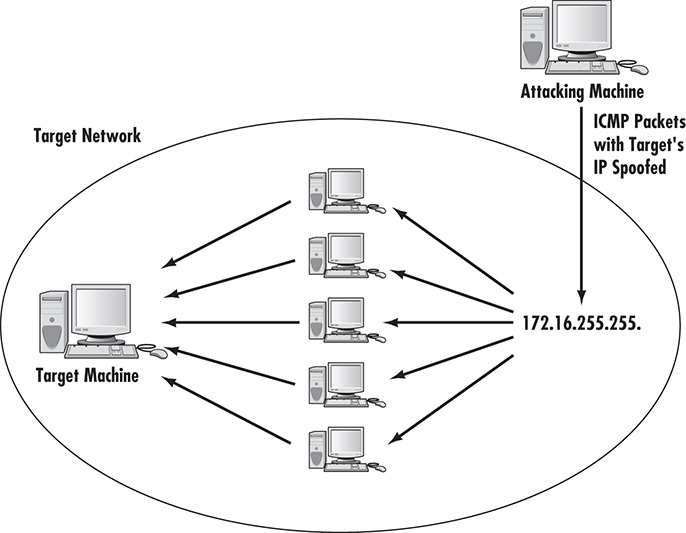
The specific implementation procedure of any of these methods depends on the operating system your web server is using. Administrators should consult your operating system’s documentation or appropriate websites in order to find explicit instructions. The most efficient way to defend against a DoS attack is a combination of these methods. The use of SYN cookies or RST cookies in conjunction with stack tweaking is a very good way to defend your web server. By combining methods, each method can overcome the others’ weaknesses. Combining these methods is rather like using both an alarm system and a security guard to protect a building. The guard can make decisions that the alarm system cannot, but the alarm system is never asleep, cannot be bribed, and is never distracted. The two methods together cover each other’s weaknesses.

**FYI: 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. Also, it only decreases the danger, but does not prevent it. For this reason it is not used as frequently as other methods.

**Smurf Attack**

The Smurf attack is a popular type of DoS attack. It was named after the application first used to execute this attack. In the Smurf attack, an ICMP packet is sent out to the broadcast address of a network, but its return address has been altered to match one of the computers on that network, most likely a key server. All the computers on the network will then respond by pinging the target computer. ICMP packets use the Internet Control Message Protocol to send error messages on the Internet. Because the address the packets are sent to is a broadcast address, that address responds by echoing the packet out to all hosts on the network, who then send it to the spoofed source address. Continually sending such packets will cause the network itself to perform a DoS attack on one or more of its member servers. This attack is both clever and simple. The greatest difficulty is getting the packets started on the target network. This can be accomplished via some software such as a virus or Trojan horse that will begin sending the packets. [Figure 2-4](https://learning.oreilly.com/library/view/network-defense-and/9780134893112/ch02.xhtml#ch2Fig4) illustrates this attack.



**FIGURE 2-4** The Smurf attack

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 autoimmune 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, continually developing new techniques. If your defense is less creative and clever than the attackers’ offense, then it is simply a matter of time before your system is compromised.

You can protect against the Smurf attack in two ways:

images The most direct method is to configure all of your routers so that they do not forward any directed broadcast packets. These packets are the cornerstone of the Smurf attack, and if routers do not forward them, then the attack is contained within one subnetwork.

images The second approach is to guard against Trojan horses (covered in depth later in this chapter). Because the Smurf attack is launched from software delivered via a Trojan horse, preventing that initial delivery will prevent the attack. Policies that prohibit employees from downloading applications and guarding a system with adequate virus scanners can go a long way to protecting the system from a Trojan horse, and thus the Smurf attack.

Using a proxy server is also imperative. Proxy servers can hide the internal IP addresses of your machine, which makes your system a lot less vulnerable to a Smurf attack. [Chapters 3](https://learning.oreilly.com/library/view/network-defense-and/9780134893112/ch03.xhtml#ch03) and [4](https://learning.oreilly.com/library/view/network-defense-and/9780134893112/ch04.xhtml#ch04) explore proxy servers and firewalls, another important tool, in detail.

**Ping of Death**

The Ping of Death (PoD), perhaps the simplest and most primitive form of DoS attack, is based on overloading the target system. TCP packets are of limited size. In some cases simply sending a packet that is too large can shut down a target machine.

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. The PoD works to compromise systems that cannot deal with extremely large packet sizes. If successful, the server will actually shut completely down. It can, of course, be rebooted.

The only real safeguard against this type of attack 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, the vendor customarily releases a patch. The possibility of PoD is one reason, among many, why you must keep patches updated on all of your systems.

This attack is becoming less common as newer versions of operating systems are better able to handle the overly large packets that Ping of Death depends on. If the operating system is properly designed it will drop any oversized packets, thus negating any possible negative effects a PoD attack might have.

**UDP Flood**

A UDP (User Datagram Protocol) flood attack is actually a variation on the experiment described earlier in this chapter. UDP is a connectionless protocol and it does not require any connection setup procedure to transfer data. TCP packets connect and wait for the recipient to acknowledge receipt before sending the next packet. Each packet is confirmed. UDP packets simply send the packets without confirmation. This allows packets to be sent much faster, making it easier to perform a DoS attack.

A UDP flood attack occurs when an attacker sends a UDP packet to a random port on the victim system. When the victim system receives a UDP packet, it will determine what application is waiting on the destination port. When it realizes that no application is waiting on the port, it will generate an ICMP packet of destination unreachable to the forged source address. If enough UDP packets are delivered to ports on the victim, the system goes down.

**ICMP Flood**

*ICMP flood* is a term you frequently encounter in security literature. In reality it is simply another name for the ping flood used in the experiment described earlier. ICMP packets are the type of packets used in the **ping** and **tracert** (this command is **tracert** in Windows and **traceroute** in Linux) utilities.

**DHCP Starvation**

DHCP starvation is another common attack. If enough requests flood onto the network, the attacker can completely exhaust the address space allocated by the DHCP servers for an indefinite period of time. There are tools such as Gobbler that will do this for you. Preventing incoming DHCP requests from outside the network will prevent this.

**HTTP Post DoS**

An HTTP Post DoS attack sends a legitimate HTTP post message. Part of the post message is the ‘content-length’. This indicates the size of the message to follow. In this attack, the attacker then sends the actual message body at an extremely slow rate. The web server is then “hung” waiting for that message to complete. For more robust servers, the attacker will need to use multiple HTTP Post attacks simultaneously.

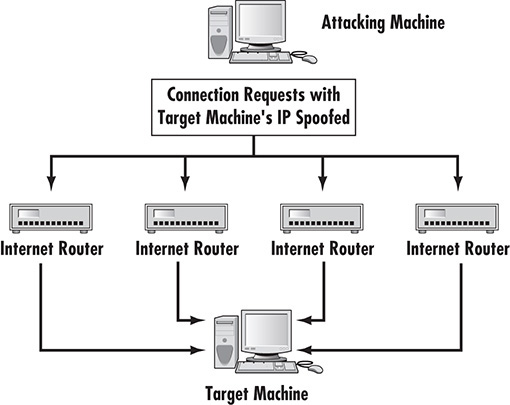
**PDoS**

A permanent denial of service (PDoS) is an attack that damages the system so badly that the victim machine needs either an operating system reinstall or even new hardware. This is sometimes called phlashing. This will usually involve a DoS attack on the device’s firmware.

**Distributed Reflection Denial of Service**

As previously stated, distributed denial of service attacks are becoming more common. Most such attacks rely on getting various machines (servers or workstations) to attack the target. The distributed reflection denial of service (DRDoS) is a special type of DoS attack. As with all such attacks, it is accomplished by the hacker getting a number of machines to attack the selected target. However, this attack works a bit differently than other DoS attacks. Rather than getting computers to attack the target, this method tricks Internet routers into attacking a target.

Many of the routers on the Internet backbone communicate on port 179. This attack exploits that communication line and gets routers to attack a target system. What makes this attack particularly wicked is that it does not require the routers in question to be compromised in any way. The attacker does not need to get any sort of software on the router to get it to participate in the attack. Instead the hacker sends a stream of packets to the various routers requesting a connection. The packets have been altered so that they appear to come from the target system’s IP address. The routers respond by initiating connections with the target system. What occurs is a flood of connections from multiple routers, all hitting the same target system. This has the effect of rendering the target system unreachable. [Figure 2-5](https://learning.oreilly.com/library/view/network-defense-and/9780134893112/ch02.xhtml#ch2Fig5) illustrates this attack.



**FIGURE 2-5** Distributed reflection denial of service

**DoS Tools**

One reason that DoS attacks are becoming so common is that a number of tools are available for executing DoS attacks. These tools are widely available on the Internet, and in most cases are free to download. This means that any prudent administrator should be aware of them. In addition to their obvious use as an attack tool, they can also be useful for testing your anti-DoS security measures.

**In Practice**

**Attacking Your Own System**

No better way exists to test your defense versus a particular sort of attack than to simulate that attack. No matter what countermeasures you implement, you won’t really know whether they are effective until you are attacked. Finding out before a real attack occurs would be better. The best way to do this is to use military-style battle exercises.

This method is not something you would try against a live machine. The preferred way to do this is to set up a machine for testing purposes. On that machine implement your various security measures, and then subject that machine to the types of attacks you hope to defend against. This can give you concrete evidence of the efficacy of your defenses.

When you conduct this sort of exercise, several guidelines should be followed:

images Always use a test system, not a live system.

images Carefully document the system’s state prior to the attack (what operating system, patches, hardware configuration, CPU usage, memory usage, what software is installed, and how the system is configured).

images Carefully document exactly what security measures you implement.

images Very specifically document each type of attack you subject the machine to.

images Document how that machine responds.

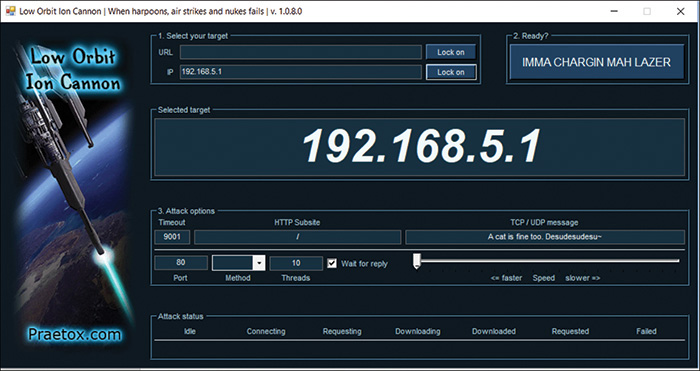
When you have completed this battle drill, you should borrow one more idea from the military, and that is the after-action review. Simply put, briefly write up how the system defenses performed and what this indicates about your system’s security.

An unfortunate fact, however, is that this particular security measure is one that is not frequently employed in industry. The primary reason is that it requires resources. You must dedicate a test machine and, more importantly, many hours to conducting the exercise. Most IT departments have a very heavy workload and simply cannot allocate the time necessary for this sort of drill. However, this is something that security consultants should definitely engage in.

**Low Orbit Ion Cannon**

This is probably the most well-known, and certainly one of the simplest, DoS tools anywhere. A simple search of the Internet will show you multiple sites you can download LOIC from.

You first put the URL or IP address into the target box. Then click the Lock On button. You can change settings regarding what method you choose, the speed, how many threads, and whether or not to wait for a reply. Then simply click the IMMA CHARGIN MAH LAZER button and the attack is underway. You can see this in [Figure 2-6](https://learning.oreilly.com/library/view/network-defense-and/9780134893112/ch02.xhtml#ch2Fig6).



**FIGURE 2-6** LOIC

**HOIC**

High Orbit Ion Cannon is a bit more advanced than LOIC, but actually simpler to run. Click the + button to add targets. A popup window will appear, where you put in the URL as well as a few settings.

**DoSHTTP**

This tool is also simple to use. You select the target, the agent (i.e., what browser type to simulate), how many sockets, the requests, then start the flood.

**Real-World Examples**

You should now have a firm grasp of what a DoS attack is and have a basic understanding of how it works. You should also have some basic ideas of how to defend your network from these attacks. It is now time to begin discussing specific, real-world examples of such attacks. The following analysis of several actual attacks illustrates the methods hackers use to launch them, their effects, their detection, and the steps administrators took to overcome them.

**FYI: Virus or Worm?**

You will see the terms *virus* and *worm* used in different books, sometimes interchangeably. However, a difference exists between the two. A worm is a specific type of virus, one that can spread without any human interaction. Traditional virus attacks come as e-mail attachments, and a human operator must open the attachment to start the infection. A worm spreads without any activity on the part of the human user. Some books use these terms rather strictly. The term *virus* accurately describes both situations.

**FakeAV**

The FakeAV virus first appeared in July 2012. It affected Windows systems ranging from Windows 95 to Windows 7 and Windows Server 2003. This was a fake antivirus (thus the name FakeAV). It would pop up fake virus warnings. This was not the first such fake antivirus malware, but it was one of the more recent ones.

**Flame**

No modern discussion of viruses would be complete without a discussion of Flame. This virus first appeared in 2012 and targeted Windows operating systems. One thing that makes this virus notable is that it was specifically designed for espionage. It was first discovered in May 2012 at several locations, including Iranian government sites. Flame is spyware that can monitor network traffic and take screenshots of the infected system.

This malware stored data in a local database that was encrypted. Flame was also able to change its behavior based on the specific antivirus running on the target machine, which indicates that this malware is highly sophisticated. Also of note is the fact that Flame was signed with a fraudulent Microsoft certificate, which meant that Windows systems would trust the software.

**MyDoom**

This is an old virus but a classic one and therefore worthy of inclusion in any discussion of viruses. In early 2004, not hearing about the MyDoom worm would have been quite difficult. This threat was a classically executed DDoS attack. The virus/worm would e-mail itself to everyone in your address book and then, at a preset time, all infected machines began a coordinated attack on [www.sco.com](http://www.sco.com/). Note that the website in question no longer exists. Estimates put the number of infected machines between 500,000 and 1 million. This attack successfully shut down the Santa Cruz Operation (SCO) website. It should be noted that well before the day that the DDoS attack was actually executed, network administrators and home users were well aware of what MyDoom would do. Several tools were available free of charge on the Internet for removing that specific virus/worm. However, apparently many people did not take the steps necessary to clean their machines of this virus/worm.

This attack is interesting to study for several reasons:

images It is a classic example of a worm. It used multiple modes to spread and could spread as an e-mail attachment or copy itself over a network.

images It was the vehicle for launching a distributed denial of service attack on a very specific target.

images It is clearly an example of domestic cyber terrorism (although certainly the creators of MyDoom would probably see it differently).

For those readers who do not know the story, it is examined here briefly. Santa Cruz Operation made a version of the Unix operating system. Like most Unix versions, their version was copyright protected. Several months before the MyDoom attack, SCO began accusing certain Linux distributions of containing segments of SCO Unix code. SCO sent letters to many Linux users demanding license fees. Many people in the Linux community viewed this as an attempt to undermine the growing popularity of Linux, an open-source operating system. SCO went even further and filed suit against major companies distributing Linux. This claim seemed unfounded to many legal and technology analysts. It was also viewed with great suspicion because SCO had close ties to Microsoft, who 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 Operation tactics were unacceptable. This hacker (or group of hackers) launched the virus to cause economic harm to SCO and to damage the company’s public image. This makes the MyDoom virus a clear case of domestic cyber terrorism: One group attacks the technological assets of another based on an ideological difference. Numerous incidents of website defacement and other small-scale attacks have arisen from ideological conflicts. However, the MyDoom attack was the first 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, there will likely be an increase in this sort of attack in the coming years.

The exact monetary damage caused by such attacks is virtually impossible to calculate. It includes the loss of service to customers, lost sales, and the impact of the negative publicity. SCO offered a $250,000 reward to anyone providing information leading to the arrest of the individuals responsible, an indication that they felt that the impact of the attack exceeded that amount.

Of particular note is the fact that variations of the MyDoom virus continued to arise long after the original intent was fulfilled. These variations used the basic MyDoom engine and spread in similar fashion, but had differing effects. As late as February 2005, new variations of MyDoom were showing up.

**FYI: Are Macs Safe?**

Some users of Macintosh assume their systems are safe from viruses. However, the MacSecurity virus, along with related viruses such as MacDefender, prove this belief is unfounded. These related viruses were all fake antiviruses designed for the Macintosh operating system. They became prevalent in 2011 and 2012. Although it is certainly true that far fewer viruses are written for the Macintosh, increasingly more viruses have been written for it as Apple has gained a larger market share.

**Gameover ZeuS**

Gameover ZeuS is a virus that creates a peer-to-peer botnet. Essentially, it establishes encrypted communication between infected computers and the command and control computer, allowing the attacker to control the various infected computers. In 2014 the U.S. Department of Justice was able to temporarily shut down communication with the command and control computers; then in 2015 the FBI announced a reward of $3 million for information leading to the capture of Evgeniy Bogachev for his alleged involvement with Gameover ZeuS.

A command and control computer is the computer used in a botnet to control the other computers. These are the central nodes from which a botnet will be managed.

**CryptoLocker and CryptoWall**

One of the most widely known examples of ransomware is the infamous CryptoLocker, first discovered in 2013. CryptoLocker utilized asymmetric encryption to lock the user’s files. Several varieties of CryptoLocker have been detected.

CryptoWall is a variant of CryptoLocker first found in August 2014. It looked and behaved much like CryptoLocker. In addition to encrypting sensitive files, it would communicate with a command and control server and even take a screenshot of the infected machine. By March 2015 a variation of CryptoWall had been discovered that is bundled with the spyware TSPY\_FAREIT.YOI and actually steals credentials from the infected system, in addition to holding files for ransom.

**Defending Against DoS Attacks**

No guaranteed way exists to prevent all DoS attacks, just as no guaranteed way exists to prevent any hacking attempt or a cyber attack. However, you can take steps to minimize the danger. This section examines some steps administrators can take to make their systems less susceptible to a DoS attack in addition to the use of SYN and RST cookies discussed previously.

One of the first things to consider is how these attacks are perpetrated. They might be executed via ICMP packets that are used to send error messages on the Internet or are sent by the ping and traceroute utilities. Simply configuring your firewall to refuse ICMP packets from outside your network will be a major step in protecting your network from DoS attacks. Because 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 might seem like a radical step, but it is certainly a secure one.

**FYI: Blocking All Traffic**

Most networks must allow some incoming traffic. This traffic might be to the network’s web server or to an e-mail server. For this reason you will not often see a firewall that blocks all incoming traffic. If you can’t block all incoming traffic, be as selective as possible and only allow in the traffic that is absolutely necessary.

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, if packets make it past your firewall, they will not be propagated throughout the network. Because all TCP packets have a source IP address, determining whether a packet originated within the network or from outside the network is not difficult. Another possibility is disabling directed IP broadcasts on all routers. This prevents the router from sending broadcast packets to all machines on the network, thus stopping many DoS attacks.

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/worm attacks and Trojan horses. Protecting against these attacks is discussed later in this chapter, but for now three important points to remember are

images Always use virus-scanning software and keep it updated.

images Always keep operating system and software patches updated.

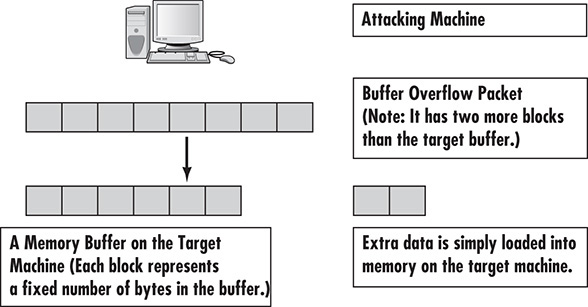
images Have an organizational policy stating that employees cannot download anything onto their machines unless the download has been cleared by the IT staff.

None of these steps will make your network totally secure from 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/](http://www.sans.org/dosstep/). This site has many good tips on preventing DoS attacks.

**DEFENDING AGAINST BUFFER OVERFLOW ATTACKS**

Virus, DoS, and Trojan horse attacks are probably the most common ways to attack a system, but they are not the only methods of attack available. Another way of attacking a system is called a buffer overflow (or buffer overrun) attack. Some experts would argue that the buffer overflow occurs as often, if not more often, than the DoS attack, but this is less true now than it was a few years ago. A buffer overflow attack is designed to put more data in a buffer than the buffer was designed to hold. However, recall that at least one worm used a buffer overflow to infect targeted machines. This means that although this threat might be less than it once was, it is still a very real threat.

Any program that communicates with the Internet or a private network must receive some data. This data is stored, at least temporarily, in a space in memory called a buffer. If the programmer who wrote the application was careful, the buffer will truncate or reject any information that exceeds the buffer limit. Given the number of applications that might be running on a target system and the number of buffers in each application, the chance of having at least one buffer that was not written properly is significant enough to cause any prudent system administrator some concern. A person moderately skilled in programming can write a program that purposefully writes more data into the buffer than it can hold. For example, if the buffer can hold 1024 bytes of data and you try to fill it with 2048 bytes, the extra 1024 bytes is then simply loaded into memory. [Figure 2-7](https://learning.oreilly.com/library/view/network-defense-and/9780134893112/ch02.xhtml#ch2Fig7) illustrates this concept.



**FIGURE 2-7** Buffer overflow attack

If the extra data is actually a malicious program, then it has just been loaded into memory and is running on the target system. Or perhaps the perpetrator simply wants to flood the target machine’s memory, thus overwriting other items that are currently in memory and causing them to crash. Either way, the buffer overflow is a very serious attack.

Fortunately, buffer overflow attacks are a bit harder to execute than the DoS or a simple MS Outlook script virus. To create a buffer overflow attack, a hacker must have a good working knowledge of some programming language (C or C++ is often chosen) and understand the target operating system/application well enough to know whether it has a buffer overflow weakness and how he might exploit the weakness.

**FYI: What Is an Outlook Script Virus?**

Microsoft Outlook is designed so that a programmer can write scripts using a subset of the Visual Basic programming language, called Visual Basic for Applications, or simply VBA. This scripting language is, in fact, built into all Microsoft Office products. Programmers can also use the closely related VBScript language. Both languages are quite easy to learn. If such a script is attached to an e-mail and if the recipient is using Outlook, then the script can execute. That execution can do any number of things, including scanning the address book, looking for addresses, sending out e-mail, deleting e-mail, and more.

Susceptibility to a buffer overflow attack is entirely contingent on software flaws. A perfectly written program would not allow buffer overflows. Because perfection is unlikely, the best defense against buffer overflow attacks is to routinely patch software so that flaws are corrected when the vendor discovers a vulnerability.

**FYI: How Do Buffer Overflows Occur?**

A buffer overflow attack can only occur if some flaw exists in the software, usually the operating system or web server. This means that the only way to prevent this type of attack is to improve the quality of software. Unfortunately, many software vendors seem to place more emphasis on being quick to market than on extensive testing and review of their software. Security-savvy administrators will pay attention to the testing methods used by their software vendors.

**DEFENDING AGAINST IP SPOOFING**

IP spoofing is essentially a technique used by hackers to gain unauthorized access to computers. Although this is the most common reason for IP spoofing, it is occasionally done simply to mask the origins of a DoS attack. In fact DoS and DDoS attacks often mask the actual IP address from which the attack is originating.

With IP spoofing, the intruder sends messages to a computer system with an IP address indicating that the message is coming from a different IP address than it is actually coming from. If the intent is to gain unauthorized access, then the spoofed IP address will be that of a system the target considers to be a trusted host. To successfully perpetrate an IP spoofing attack, the hacker must first find the IP address of a machine that the target system considers a trusted source. Hackers might employ a variety of techniques to find an IP address of a trusted host. After they have that trusted IP address, they can then modify the packet headers of their transmissions so it appears that the packets are coming from that host.

IP spoofing, unlike many other types of attacks, was actually known to security experts on a theoretical level before it was ever used in a real attack. The concept of IP spoofing was initially discussed in academic circles as early as the 1980s. Although the concept behind this technique was known for some time, it was primarily theoretical until Robert Morris discovered a security weakness in the TCP protocol known as sequence prediction. Stephen Bellovin discussed the problem in-depth in his famous paper, “Security Problems in the TCP/IP Protocol Suite.”

IP spoofing attacks are becoming less frequent, primarily because the venues they use are becoming more secure and in some cases are simply no longer used. However, spoofing can still be used, and all security administrators should address it. A couple different ways to address IP spoofing include:

images To not to reveal any information regarding your internal IP addresses. This helps prevent those addresses from being “spoofed.”

images To monitor incoming IP packets for signs of IP spoofing using network monitoring software. One popular product is Netlog. This and similar products seek incoming packets to the external interface that have both the source and destination IP addresses in your local domain, which essentially means an incoming packet that claims to be from inside the network, when it is clearly coming from outside your network. Finding one means an attack is underway.

The danger from IP spoofing is that some firewalls do not examine packets that appear to come from an internal IP address. Routing packets through filtering routers is possible if they are not configured to filter incoming packets whose source address is in the local domain.

Examples of router configurations that are potentially vulnerable include

images Routers to external networks that support multiple internal interfaces

images Proxy firewalls where the proxy applications use the source IP address for authentication

images Routers with two interfaces that support subnetting on the internal network

images Routers that do not filter packets whose source address is in the local domain

The best method of preventing IP spoofing is to install a filtering router. Filtering routers filter incoming packets by not allowing a packet through if it has a source address from your internal network. In addition, you should filter outgoing packets that have a source address different from your internal network to prevent a source IP spoofing attack from originating at your site. Many commercial firewall vendors, such as Cisco, FortiGate, D-Link, and Juniper, offer this option.

If your vendor’s router does not support filtering on the inbound side of the interface and you feel the need to immediately filter such packets, you can filter the spoofed IP packets by using a second router between your external interface and your outside connection. Configure this router to block, on the outgoing interface connected to your original router, all packets that have a source address in your internal network. For this purpose, you can use a filtering router or a Unix system with two interfaces that supports packet filtering.

**DEFENDING AGAINST SESSION HIJACKING**

Another form of attack is session hacking or hijacking. TCP session hijacking is a process whereby a hacker takes over a TCP session between two machines. Because authentication frequently is done only at the start of a TCP session, this allows the hacker to break into the communication stream and take control of the session. For example, a person might log on to a machine remotely. After she has established a connection with the host, the hacker might use session hacking to take over that session and thereby gain access to the target machine.

One popular method for session hacking is using source-routed IP packets. This allows a hacker at point A on the network to participate in a conversation between B and C by encouraging the IP packets to pass through the hacker’s machine.

The most common sort of session hacking is the “man-in-the-middle attack.” In this scenario a hacker uses some sort of packet-sniffing program to simply listen in on the transmissions between two computers, taking whatever information he or she wants but not actually disrupting the conversation. A common component of such an attack is to execute a DoS attack against one end point to stop it from responding. Because that end point is no longer responding, the hacker can now interject his own machine to stand in for that end point.

The point of hijacking a connection is to exploit trust and to gain access to a system to which one would not otherwise have access.

The only way to truly defend against session hacking is to use encrypted transmissions. [Chapter 6](https://learning.oreilly.com/library/view/network-defense-and/9780134893112/ch06.xhtml#ch06), “Encryption Fundamentals,” discusses various encryption methods. If the packets are not encrypted, then the communication is vulnerable to session hacking. Many network administrators do use encrypted transmissions when communicating outside their network, but fewer encrypt internal communications. For a truly high level of security, consider encrypting all transmissions.

Packet sniffers are discussed in more detail in [Chapter 15](https://learning.oreilly.com/library/view/network-defense-and/9780134893112/ch15.xhtml#ch15), “Techniques Used by Attackers.” Right now what you need to be aware of is that a packet sniffer is software that intercepts packets going across a network or the Internet and copies them. This gives the attacker a copy of every packet you send. These tools have legitimate uses in network traffic monitoring, but are also used by hackers to intercept communications and, in some cases, for session hacking.

**BLOCKING VIRUS AND TROJAN HORSE ATTACKS**

The preceding sections of this chapter examined the denial of service attack, buffer overflow attack, and session hacking. However, virus attacks are the most prevalent threats to any network. Therefore, protecting against virus attacks is essential for any security-conscious network administrator.

**Viruses**

By definition, a computer virus is a program that self-replicates. Generally, a virus also has some other unpleasant function, but the self-replication and rapid spread are its hallmarks. This growth, in and of itself, can be a problem for an infected network. Worms are viruses that can replicate without human interaction.

Consider the infamous Slammer virus and the effects of its rapid, high-volume scanning. Any rapidly spreading virus can reduce the functionality and responsiveness of a network. It can lead to too much network traffic and prevent the network from functioning properly. Simply by exceeding the traffic load a network was designed to carry, the network can be rendered temporarily nonfunctional.

**How Does a Virus Spread?**

You have seen how viruses can impact infected systems and have looked at a few actual cases. Clearly the key to stopping a computer virus is to prevent it from spreading to other computers. To do this you must have a good understanding of how viruses typically spread. A virus usually spreads in one of two ways. The first is by scanning a computer for connections to a network and then copying itself to other machines on the network to which that machine has access. This is the most efficient way for a virus to spread and is a typical spread method of worms. However, this method requires more programming skill than other methods. The second and more popular method is reading the e-mail address book and sending itself to everyone in it. Programming this type of virus is a trivial task, which explains why its use is so prevalent.

The latter method is, by far, the most usual method for virus propagation, and Microsoft Outlook might be the one e-mail program most often hit with such virus attacks. The reason is not so much a security flaw in Outlook as it is the ease of working with Outlook. All Microsoft Office products are made so that a legitimate programmer can access many of that application’s internal objects and thereby easily create applications that integrate the applications within the Microsoft Office suite. For example, a programmer could write an application that would access a Word document, import an Excel spreadsheet, and then use Outlook to automatically e-mail the resulting document to interested parties. Microsoft has done a good job of making this process very easy. Accomplishing these tasks usually takes a minimal amount of programming. In the case of Outlook, referencing Outlook and sending out an e-mail takes less than five lines of code to do. This means a program can literally cause Outlook itself to send e-mails, unbeknownst to the user. Numerous code examples on the Internet show exactly how to do this, free for the taking.

However a virus arrives, after it is on a system it attempts to spread. In many cases the virus also attempts to cause some harm to the system. After a virus is on a system it can do anything a legitimate program can do. That means it could potentially delete files, change system settings, or cause other harm. The threat from virus attacks cannot be overstated. Let’s take a moment to look at a few virus outbreaks, see how they operated, and describe the damage they caused. Some of these are older, some recent.

**The Sobig Virus**

The Sobig virus is an older virus but it is a good example of how viruses spread. One interesting thing about this virus was the multi-modal approach by which it spread. In other words, it used more than one mechanism to spread and infect new machines. Sobig copied itself to any shared drives on the network and it e-mailed itself out to everyone in the address book. Because of this approach, Sobig can be classified as a worm rather than simply a virus. This multi-modal ability to spread meant that Sobig was particularly *virulent*—a term signifying that the virus spread rapidly and easily infected new targets.

This multi-modal spread ability is why ensuring that each and every person in your organization is cautioned about proper security policies and procedures is so critical. If just one person on a network was unfortunate enough to open an e-mail containing the Sobig virus, it infected not only that machine but also every shared drive on the network that this person could access.

Like most e-mail distributed virus attacks, this one had telltale signs in the e-mail subject or title that could be used to identify the e-mail as one infected by a virus. The e-mail would have a title such as “here is the sample” or “the document” and encourage you to open the attached file. The virus then copied itself into the Windows system directory. Some variants of Sobig caused computers to download a file from the Internet that would then cause printing problems. Some network printers would just start printing junk. The Sobig.E variant even wrote to the Windows registry, causing the virus to be included in the computer startup. These complex characteristics indicate that the creator of Sobig knew how to access the Windows registry, access shared drives, alter the Windows startup, and access Outlook.

A method I personally use and recommend to all security administrators is to routinely send out an e-mail to everyone in your organization telling them the telltale signs to be wary of in e-mails. Websites such as [www.f-secure.com](http://www.f-secure.com/) list current viruses and what to look for in an e-mail. I summarize this list and send it out once or twice a month to everyone in my organization. That way all members of the organization are aware of e-mails that they should definitely not open. If you couple this with instilling a healthy caution toward unexpected e-mails, you can drastically reduce the chance of becoming infected with a virus.

This particular virus spread so far and infected so many networks that the multiple copying of the virus alone was enough to bring some networks to a standstill. This virus did not destroy files or damage the system, but it generated enough traffic to bog down the networks infected by it. The virus itself was of moderate sophistication. After it was out, many variants began to spring up, further complicating the situation.

**Virus Variations**

Sometimes, some intrepid programmer with malicious intent receives a copy of a virus (perhaps his or her own machine becomes infected) and decides to reverse-engineer it. Many virus attacks are in the form of a script attached to an e-mail. This means that unlike traditional compiled programs, their source code is readily readable and alterable. The programmer in question then simply takes the original virus code, introduces some change, and then re-releases the variant. The people who are most frequently caught for creating viruses are the developers of variants who simply lack the skill of the original virus writer and are therefore easily caught.

**FYI: The Economic Impact of Viruses**

Getting an exact accounting of the damage caused by Flame, FakeAV, MacDefender, or any other virus is impossible. However, if one considers the number of hours IT professionals spend working on cleaning up a given virus, the worldwide cost of any virus is in the millions of dollars. If the amount of money spent trying to defend against such viruses through antivirus software, hiring consultants, and purchasing books like this one is included, the annual impact of all viruses can easily reach billions of dollars. In fact, one study showed that in 2007 the economic damages from viruses exceeded $14 billion.

For this reason many security experts recommend that governments begin enacting stronger penalties for virus creators. Federal law enforcement agencies taking a more active role in investigating these crimes would also be helpful. For example, some, private companies such as Microsoft have begun offering substantial bounties for information leading to the arrest of virus creators, a very positive step.

You can read more about any virus, past or current, at the following websites:

images [www.f-secure.com/en/web/labs\_global/from-the-labs](http://www.f-secure.com/en/web/labs_global/from-the-labs)

images [www.cert.org/news/](http://www.cert.org/news/)

images [www.symantec.com/security-center](http://www.symantec.com/security-center)

**FYI: Why Write a Virus?**

In the case of Flame or Stuxnet, ascertaining the virus writer’s motivations was not hard. These viruses were designed to spy on or disrupt specific government activities of specific countries. In other cases, the virus (particularly with ransomware) is part of a scheme to extract money from the victim. And again, the motives are not hard to discern. However, with other viruses such as Bagle and Mimail, understanding why the virus was created in the first place is difficult. To the best of my knowledge, no formal psychological studies exist regarding the mentality of virus writers. However, having interacted with alleged virus writers in various forums, and having read interviews with convicted virus writers, I can provide you with some insight into their mentality.

In some cases the virus writer simply wanted to prove that he could do it. For some people, simply knowing that they “outwitted” numerous security professionals gives them a feeling of satisfaction. For others, the ability to cause damage on a wide scale imbues them with a sense of power they probably do not otherwise feel. When virus writers are caught they usually turn out to be young, intelligent, and technically skilled; have strong antisocial leanings; and generally don’t fit into any peer group. The writing of a virus gives them the same cathartic feeling that other people get from vandalism and graffiti.

**The Virus Hoax**

Another type of virus has been somewhat popular: the “non-virus virus,” or virus hoax. Rather than actually writing a virus, a hacker simply sends an e-mail to every address he has. The e-mail claims to be from some well-known antivirus center and warns of a new virus that is circulating. The e-mail then instructs the user to delete a file from the computer to get rid of the virus. The file, however, is not really a virus but part of the computer’s system. One of the earliest examples, the jdbgmgr.exe virus hoax, used this scheme. It encouraged the reader to delete a file that was actually needed by the system. Surprisingly, a large number of people followed this advice and not only deleted the file but promptly e-mailed their friends and colleagues to warn them to delete the file from their machines.

A common theme with all virus attacks (except the hoax) is that they instruct the recipient to open some type of attachment. The predominant way for a virus to spread is as an e-mail attachment. This realization leads to some simple rules that can drastically reduce the odds of a machine becoming infected with a virus:

images Always use a virus scanner. McAfee and Norton are the two most widely accepted and used virus scanners. Malware Bytes, AVG, and others are also effective. Each costs about $30 a year to keep updated. Do it. [Chapter 9](https://learning.oreilly.com/library/view/network-defense-and/9780134893112/ch09.xhtml#ch09), “Defending Against Virus Attacks,” discusses virus attacks and virus scanners in more detail.

images If you are unsure about an attachment, do not open it.

images You might even exchange a code word with friends and colleagues. Tell them that if they want to send you an attachment, they should put the code word in the title of the message. Without seeing the code word, you will not open any attachment.

images Don’t believe “security alerts” that are sent to you. Microsoft does not send out alerts in this manner. Check the Microsoft website regularly, as well as one of the antivirus websites previously mentioned. Microsoft’s security website ([www.microsoft.com/security/](http://www.microsoft.com/security/)) is the only reliable place to get Microsoft security updates. Other security sites might have accurate information (such as [www.sans.org](http://www.sans.org/)) but if you are using a particular vendor’s software (such as Microsoft) then going to its site to find alerts and to get patches is always best.

These rules will not make systems 100% virus proof, but they will go a long way toward protecting them.

**FYI: It’s Not Hard**

One would think that writing a virus would require extensive computer programming skills. Although it is true that writing a sophisticated virus does indeed require a high level of skill, utilities are actually available on the web (such as JPS virus maker) to help you create a virus as well as tools (such as EliteWrap) that can help you create a Trojan horse.

**Types of Viruses**

There are many different types of viruses. In this section we will briefly look at some of the major virus types. Viruses can be classified by either their method of propagation or their activities on the target computers. It must also be noted that various experts differ slightly on how they group viruses. The taxonomy presented in this section is rather common, and I find it to be quite useful. It is one I have developed over the years.

**Macro Viruses**

Macro viruses infect the macros in office documents. Many office products, including Microsoft Office, allow users to write mini-programs called macros. These macros can also be written as a virus. A macro virus is written into a macro in some business application. For example, Microsoft Office allows users to write macros to automate some tasks. Microsoft Outlook is designed so that a programmer can write scripts using a subset of the Visual Basic programming language, called Visual Basic for Applications (VBA). This scripting language is, in fact, built into all Microsoft Office products. Programmers can also use the closely related VBScript language. Both languages are quite easy to learn. If such a script is attached to an e-mail and the recipient is using Outlook, then the script can execute. That execution can do any number of things, including scanning the address book, looking for addresses, sending out e-mail, deleting e-mail, and more.

**Boot Sector**

Boot sector viruses don’t infect the operating system of the target computer, but instead attack the boot sector of the drive. This makes them harder to detect and remove with traditional antivirus software. Such software is installed in the operating system, and to some extent only operates within the context of the operating system. By operating outside the operating system, a boot sector virus is harder to detect and remove. Multipartite viruses attack the computer in multiple ways—for example, infecting the boot sector of the hard disk and one or more files within the operating system.

**Stealth**

Stealth viruses are one of the largest groups of viruses. This category includes any virus that uses one or more techniques to hide itself. In other words, these are viruses that are trying to avoid your antivirus software.

The Trojan horse is an excellent way to hide a virus. By tying it to a legitimate program, it not only will trick the user into installing it, but it may also evade antivirus software.

A polymorphic virus literally changes its form from time to time to avoid detection by antivirus software. A more advanced form of this is called the metamorphic virus, which can completely change itself. This also requires a secondary module to perform the rewriting.

A sparse infector virus attempts to elude detection by performing its malicious activities only sporadically. With a sparse infector virus, the user will see symptoms for a short period, then no symptoms for a time. In some cases the sparse infector targets a specific program but the virus only executes every 10th time or 20th time that target program executes. Or a sparse infector may have a burst of activity and then lie dormant for a period of time. There are a number of variations on the theme, but the basic principle is the same: to reduce the frequency of attack and thus reduce the chances for detection.

Fragmented payload is a rather sophisticated method of hiding a virus. The virus is split into modules. The loader module is rather innocuous and unlikely to trigger any antivirus software. It will then download, separately, the other fragments. When all fragments are present, the loader will assemble them and unleash the virus.

**Ransomware**

It is impossible in modern times to discuss malware and not discuss ransomware. In fact, as I am writing this, in the past 48 hours the world has been hit with a massive ransomware attack. It began by attacking health care systems in England and Scotland, and spread far beyond those. That virus is the infamous WannaCry virus. While many people first began discussing ransomware with the advent of CryptoLocker in 2013, ransomware has been around a lot longer than that. The first known ransomware was the 1989 PC Cyborg Trojan, which only encrypted filenames with a weak symmetric cipher.

In general, ransomware works as a worm, then either disables system services or encrypts user files. It then demands a ransom to release those files/service.

**Trojan Horses**

You have seen the term *Trojan horse* used in this chapter, and you probably already have some idea of what it is. A Trojan horse is a program that looks benign but actually has a malicious purpose. You might receive or download a program that appears to be a harmless business utility or game. More likely, the Trojan horse is just a script attached to a benign-looking e-mail. When you run the program or open the attachment, it does something else other than or in addition to what you thought it would. It might

images Download harmful software from a website.

images Install a key logger or other spyware on your machine.

images Delete files.

images Open a backdoor for a hacker to use.

Finding virus and Trojan horse attack combinations is commonplace. In these instances, the Trojan horse spreads like a virus. The MyDoom virus opened a port on machines that a later virus, Doomjuice, would exploit, thus making MyDoom a combination virus and Trojan horse.

**FYI: Was MyDoom a Trojan Horse?**

Some experts say that MyDoom was not actually a Trojan horse because it did not pretend to be benign software. However, one could argue that the e-mail attachment that delivered MyDoom did indeed claim to be a legitimate attachment and, thus, could be classified as a Trojan horse. Whether or not you agree that MyDoom is a Trojan horse, it is certainly a good illustration of how malicious software can take multiple avenues to cause harm.

A Trojan horse also could be crafted especially for an individual. If a hacker wanted to spy on a certain individual, such as the company accountant, she could design a program specifically to attract that person’s attention. For example, if she knew the accountant was an avid golfer, she could write a program that computed handicap and listed best golf courses. She would post that program on a free web server. She would then e-mail a number of people, including the accountant, telling them about the free software. The software, once installed, could check the name of the currently logged-on person. If the logon name matched the accountant’s name, the software could then go out, unknown to the user, and download a key logger or other monitoring application. If the software did not damage files or replicate itself, then it would probably go undetected for quite a long time.

Writing such a program could be within the skillset of virtually any moderately competent programmer. This is one reason many organizations have rules against downloading any software onto company machines. I am unaware of any actual incident of a Trojan horse being custom tailored in this fashion. However, remember that those who create virus attacks tend to be innovative people.

Another scenario to consider is one that would be quite devastating. Without divulging programming details, the basic premise is outlined here to illustrate the grave dangers of Trojan horses. Imagine a small application that displays a series of unflattering pictures of Osama Bin Laden. This would probably be popular with many people in the United States, particularly people in the military, the intelligence community, or defense-related industries. Now assume that the application simply sits dormant on the machine for a period of time. It need not replicate like a virus because the computer user will probably send it to many of his or her associates. On a certain date and time, the software connects to any drive it can, including network drives, and begins deleting all files.

If such a Trojan horse were released “in the wild,” within 30 days it would probably be shipped to thousands, perhaps millions, of people. Imagine the devastation when thousands of computers begin deleting files and folders.

This scenario is mentioned precisely to frighten you a little. Computer users, including professionals who should know better, routinely download all sorts of files from the Internet, including amusing Flash animations and cute games. Every time an employee downloads something of this nature, the chance of downloading a Trojan horse exists. One need not be a statistician to realize that if employees continue that practice long enough they will eventually download a Trojan horse on to a company machine. A user can only hope it is not one as vicious as the theoretical one just outlined here.

**FYI: Using Trojan Horse Examples**

Readers are strongly cautioned against attempting to actually create any of these Trojan horse scenarios. Releasing such an application is a crime that will probably result in a lengthy prison sentence and serious civil penalties. These examples are provided simply to show you just how devastating a Trojan horse can be.

The people who create Trojan horses and viruses are quite creative. New variations pop up frequently. It is highly likely that someone else has already thought of something similar to the scenarios I present. The point of presenting these scenarios is to make sure that network administrators exercise the appropriate level of caution. To be totally frank, it is my wish that every network administrator have a certain level of paranoia regarding viruses and Trojan horses.