

Introducing Network Analysis

Solutions in this Chapter:

- What is Network Analysis and Sniffing?
 - Who Uses Network Analysis?
 - How Does it Work?
 - Detecting Sniffers
 - Protecting Against Sniffers
 - Network Analysis and Policy
-
- ☑ Summary
 - ☑ Solutions Fast Track
 - ☑ Frequently Asked Questions

Introduction

“Why is the network slow?” “Why can’t I access my e-mail?” “Why can’t I get to the shared drive?” “Why is my computer acting strange?” If you are a systems administrator, network engineer, or security engineer you have probably heard these questions countless times. Thus begins the tedious and sometimes painful journey of troubleshooting. You start by trying to replicate the problem from your computer. Sure enough, you can’t get to anything on the local network or the Internet either. Now what? Go to each of the servers and make sure they are up and functioning? Check that your router is functioning? Check each computer for a malfunctioning network card?

What about this scenario: you go to your main access switch, or border router, and configure one of the unused ports for port mirroring. You plug in your laptop, fire up your network analyzer, and see thousands of User Datagram Protocol (UDP) packets destined for port 1434 with various, apparently random, Internet Protocol (IP) addresses. You immediately apply access filters to block these packets from entering or exiting your network until you do more investigating. A quick search on the Internet holds the answer. The date is January 25, 2003, and you have just been hit with the SQL Slammer worm. You were able to contain the problem relatively quickly thanks to your knowledge and use of your network analyzer.

What is Network Analysis and Sniffing?

Network analysis is the process of capturing network traffic and inspecting it closely to determine what is happening on the network. A network analyzer decodes, or dissects, the data packets of common protocols and displays the network traffic in human-readable format. Network analysis is also known by several other names: traffic analysis, protocol analysis, sniffing, packet analysis, and eavesdropping to name a few. Sniffing tends to be one of the most popular terms in use today. However, as you will see later in this chapter, due to malicious users it has had a negative connotation in the past.

A network analyzer can be a standalone hardware device with specialized software, or it can simply be software that you install on your desktop or laptop computer. Network analyzers are available both free and commercially. Differences between network analyzers tend to depend on features such as the number of supported protocol decodes, the user interface, and graphing and statistical capabilities. Other differences include inference capabilities, such as expert

analysis features, and the quality of packet decodes. Although several network analyzers all decode the same protocols, some may decode better than others.

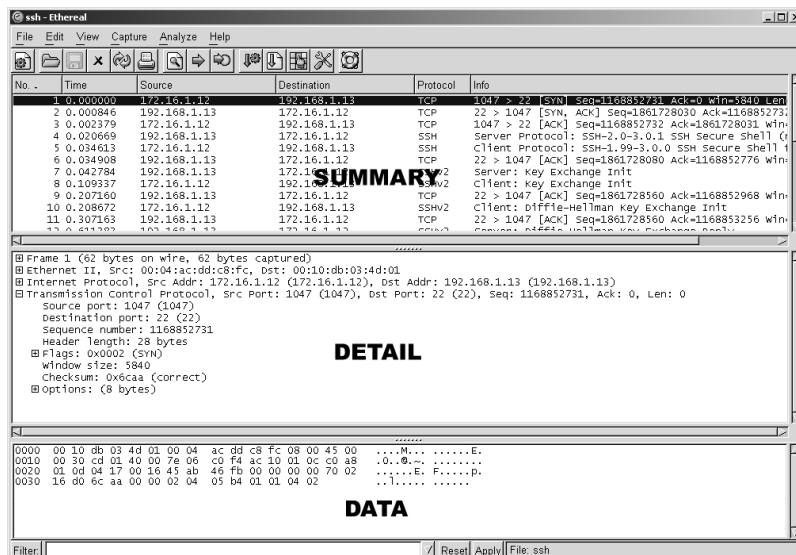
NOTE

Sniffer (with a capital “S”) is a trademark owned by Network Associates referring to its Sniffer product line. However, it has become common industry usage that a “sniffer” (with a lower case “s”) is a program that captures and analyzes network traffic.

Figure 1.1 shows the Ethereal Network Analyzer display windows. A typical network analyzer displays the captured traffic in three panes:

- **Summary** This pane displays a one line summary of the capture. Fields usually include date, time, source address, destination address, and the name and information about the highest-layer protocol.
- **Detail** This pane provides all of the details for each of the layers contained inside the captured packet in a tree-like structure.
- **Data** This pane displays the raw captured data both in hexadecimal and ASCII format.

Figure 1.1 Example Network Analyzer Display



A network analyzer is a combination of hardware and software. Although there are differences in each product, a network analyzer is composed of five basic parts:

- **Hardware** Most network analyzers are software-based and work with standard operating systems (OSs) and network interface cards (NICs). However, there are some special hardware network analyzers that offer additional benefits such as analyzing hardware faults including: Cyclic Redundancy Check (CRC) errors, voltage problems, cable problems, jitter, jabber, negotiation errors, etc. Some network analyzers only support Ethernet or wireless adapters, while others support multiple adapters and allow users to customize their configuration. Sometimes you will also need a hub or a cable tap to connect to the existing cable.
- **Capture driver** This is the part of a network analyzer that is responsible for actually capturing the raw network traffic from the cable. It will also filter out the traffic that you want and store the data in a buffer. This is the core of a network analyzer and you cannot capture data without it.
- **Buffer** This component stores the captured data. Data can be stored in a buffer until it is full, or in a rotation method such as “round robin” where the newest data replaces the oldest data. Buffers can be disk-based or memory-based.
- **Real-time analysis** This feature analyzes the data as it comes off the cable. Some network analyzers use this to find network performance issues, and network intrusion detection systems do this to look for signs of intruder activity.
- **Decode** This component displays the contents of the network traffic with descriptions so that it is human-readable. Decodes are specific to each protocol, so network analyzers tend to vary in the number of decodes they currently support. However, new decodes are constantly being added to network analyzers.

NOTE

Jitter is a term used to describe the random variation in the timing of a signal. Electromagnetic interference and crosstalk with other signals can cause jitter. *Jabber* is when a device is improperly handling electrical signals, thus affecting the rest of the network. Faulty network interface cards can cause jabber.

Who Uses Network Analysis?

System administrators, network engineers, security engineers, system operators, even programmers, all use network analyzers. Network analyzers are invaluable tools for diagnosing and troubleshooting network problems. Network analyzers used to be dedicated hardware devices that were very expensive. New advances in technology have allowed for the development of software network analyzers. This makes it more convenient and affordable for administrators to effectively troubleshoot a network. It also brings the capability of network analysis to anyone who wishes to perform it.

The art of network analysis is a double-edged sword. While network, system, and security professionals use it for troubleshooting and monitoring of the network, intruders can also use network analysis for harmful purposes. A network analyzer is a tool, and like all tools they can be used for both good and bad intentions.

The following list describes a few reasons why administrators use network analyzers:

- Converting the binary data in packets to human-readable format
- Troubleshooting problems on the network
- Analyzing the performance of a network to discover bottlenecks
- Network intrusion detection
- Logging network traffic for forensics and evidence
- Analyzing the operations of applications
- Discovering a faulty network card
- Discovering the origin of a Denial of Service (DoS) attack

- Detecting spyware
- Network programming to debug in the development stage
- Detecting a compromised computer
- Validating compliance with company policy
- As an educational resource when learning about protocols
- For reverse-engineering protocols in order to write clients and supporting programs

How are Intruders Using Sniffers?

When used by malicious individuals, sniffers can represent a significant threat to the security of your network. Network intruders often use network sniffing to capture valuable, confidential information. The terms sniffing and eavesdropping have often been associated with this practice. However, sniffing is now becoming a non-negative term and most people use the terms sniffing and network analysis interchangeably.

Using a sniffer in an illegitimate way is considered a passive attack. It does not directly interface or connect to any other systems on the network. However, the computer that the sniffer is installed on could have been compromised using an active attack. The passive nature of sniffers is what makes detecting them so difficult. We will discuss the methods used to detect sniffers later in this chapter.

The following list describes a few reasons why intruders are using sniffers on the network:

- Capturing clear-text usernames and passwords
- Compromising proprietary information
- Capturing and replaying Voice over IP telephone conversations
- Mapping a network
- Passive OS fingerprinting

Obviously, these are illegal uses of a sniffer, unless you are a penetration tester whose job it is to find these types of weaknesses and report them to an organization.

For sniffing to occur, an intruder must first gain access to the communication cable of the systems that are of interest. This means being on the same shared net-

work segment, or tapping into the cable somewhere between the path of communications. If the intruder is not physically present at the target system or communications access point, there are still ways to sniff network traffic. These include:

- Breaking into a target computer and installing remotely controlled sniffing software.
- Breaking into a communications access point, such as an Internet Service Provider (ISP) and installing sniffing software.
- Locating/finding a system at the ISP that already has sniffing software installed.
- Using social engineering to gain physical access at an ISP to install a packet sniffer.
- Having an insider accomplice at the target computer organization or the ISP install the sniffer.
- Redirecting communications to take a path that includes the intruder's computer.

Sniffing programs are included with most *rootkits* that are typically installed on compromised systems. Rootkits are used to cover the tracks of the intruder by replacing commands and utilities and clearing log entries. They also install other programs such as sniffers, key loggers, and backdoor access software. Windows sniffing can be accomplished as part of some RAT (Remote Admin Trojan) such as SubSeven or Back Orifice. Often intruders will use sniffing programs that are configured to detect specific things, such as passwords, and then electronically send them to the intruder (or store them for later retrieval by the intruder). Vulnerable protocols for this type of activity include telnet, FTP, POP3, IMAP, SMTP, HTTP, rlogin, and SNMP.

One example of a rootkit is T0rnKit, which works on Solaris and Linux. The sniffer that is included with this rootkit is called t0rns and is installed in the hidden directory `/usr/src/.puta`. Another example of a rootkit is Lrk5 (Linux Rootkit 5), which installs with the linsniff sniffer.

Intruders commonly use sniffer programs to control back doors. One method is to install a sniffer on a target system that listens for specific information. Then, backdoor control information can be sent to a neighboring system. The sniffer picks this up, and acts appropriately on the target computer. This type of backdoor control is often hard for investigators to detect, since it looks like the innocent neighbor system is the compromised target.

cd00r is an example of a backdoor sniffer that operates in non-promiscuous mode, making it even harder to detect. Using a product like Nmap to send a series of Transmission Control Protocol (TCP) SYN packets to several predefined ports will trigger the backdoor to open up on a pre-configured port. More information about Cdoor can be found at www.phenoelit.de/stuff/cd00r.c.

NOTE

A *rootkit* is a collection of trojan programs that are used to replace the real programs on a compromised system in order to avoid detection. Some common commands that get replaced are *ps*, *ifconfig*, and *ls*. Rootkits also install additional software such as sniffers.

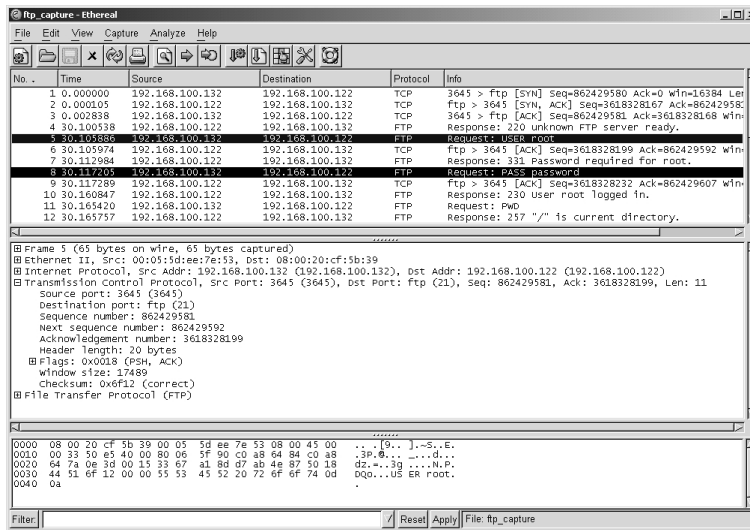
NOTE

Nmap is a network scanning tool used for network discovery and auditing. It can send raw IP packets to destination ports on target systems.

What does Sniffed Data Look Like?

We have done a lot of talking about sniffers and what they are used for, but the easiest way to grasp the concepts previously discussed is watching a sniffer in action. Figure 1.2 shows a capture of a simple FTP session from a laptop to a Sun Solaris system. The two highlighted packets show you just how easy it is to sniff the username and password. In this case, the username is “root” and the password is “password”. Of course, allowing root to FTP into a system is a very poor security practice; this is just for illustration purposes!

Figure 1.2 Example of Sniffing a Connection



Common Network Analyzers

A simple search on SecurityFocus (www.securityfocus.org/tools/category/4) shows the diversity and number of sniffers available. Some of the most prominent ones are:

- Ethereal** Of course, this one is the topic of this book! Ethereal is obviously one of the best sniffers available. It is being developed as a free commercial quality sniffer. It has numerous features, a nice graphical user interface (GUI), decodes for over 400 protocols, and it is actively being developed and maintained. It runs on both UNIX-based systems and Windows. This is a great sniffer to use, even in a production environment. It is available at www.ethereal.com.
- WinDump** This is the Windows version of tcpdump available at <http://windump.polito.it>. It uses the WinPcap library and runs on Windows 95/98/ME/NT/2000/XP.
- Network Associates Sniffer** This is one of the most popular commercial products available. Now marketed under McAfee Network Protection Solutions, Network Associates has an entire Sniffer product line for you to peruse at www.nai.com.

- **Windows 2000/NT Server Network Monitor** Both Windows 2000 Server and NT Server have a built-in program to perform network analysis. It is located in the Administrative tools folder, but is not installed by default, so you may have to add it from the installation CD.
- **EtherPeek** This is a commercial network analyzer by WildPackets. There are versions for both Windows and Mac, as well as other network analysis products that can be found at www.wildpackets.com.
- **Tcpdump** This is the oldest and most common network sniffer. The Network Research Group (NRG) of the Information and Computing Sciences Division (ICSD) at Lawrence Berkeley National Laboratory (LBNL) developed tcpdump. It is command line-based and runs on UNIX-based systems. It is being actively developed and maintained at www.tcpdump.org.
- **Snoop** This command line network sniffer is included with the Sun Solaris operating system. It is especially competent at decoding Sun-specific protocols.
- **Sniffit** This network sniffer runs on Linux, SunOS, Solaris, FreeBSD and IRIX. It is available at <http://reptile.rug.ac.be/~coder/sniffit/sniffit.html>.
- **Snort** This is a network intrusion detection system that uses network sniffing. It is actively developed and maintained at www.snort.org. For more information, refer to *Snort 2.0: Intrusion Detection* (Syngress Publishing, ISBN: 1-931836-74-4)
- **Dsniff** This is very popular network sniffing package. It is a collection of programs to sniff specifically for interesting data such as passwords, and to facilitate the sniffing process such as evading switches. It is actively maintained at www.monkey.org/~dugsong/dsniff.
- **Ettercap** This sniffer is designed specifically to sniff in a switched network. It has built-in features such as password collecting, OS fingerprinting, and character injection. It runs on several platforms including Linux, Windows, and Solaris. It is actively maintained at <http://ettercap.sourceforge.net>.
- **Analyzer** This is a free sniffer for the Windows OS that is being actively developed by the makers of WinPcap and WinDump at

Politecnico di Torino. It can be downloaded from <http://analyzer.polito.it>.

- **Packetyzer** This is a free sniffer for the Windows OS that uses Ethernet's core logic. It tends to run a version or two behind the current release of Ethernet. It is actively maintained by Network Chemistry at www.networkchemistry.com/products/packetyzer/index.html.

Notes from the Underground...

Carnivore or Vegetarian?

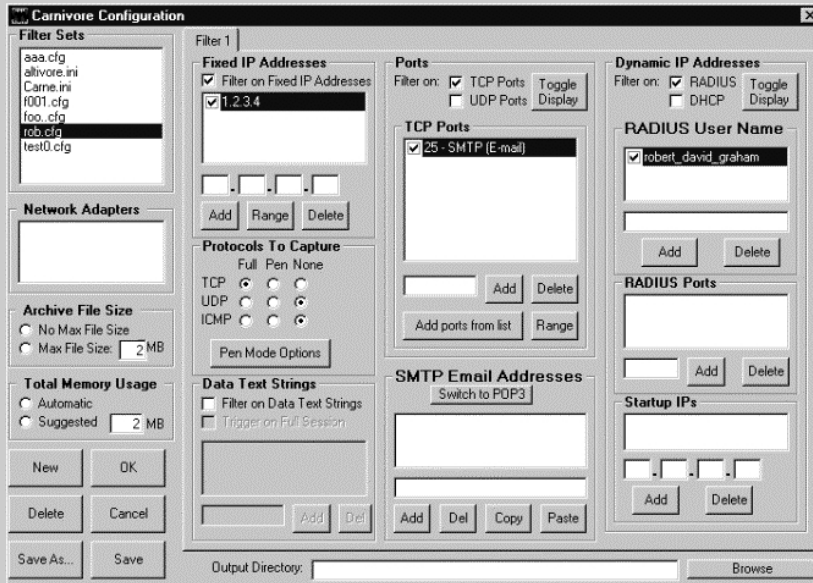
No talk about network analyzers would be complete without the mention of Carnivore. While certainly not a commonly used network analyzer, it has created a lot of talk in the security world as well as the media. Carnivore is the code name for the FBI's network analyzer. It is used to monitor relevant communications among selected individuals as part of a criminal investigation. Its name has been changed to DCS100 in an attempt to obscure its image and to calm the public's fear of its misuse. When necessary, federal agents will arrive at a suspect's ISP with a "black box", which is merely a dedicated server running Windows 2000 or NT and the FBI's Carnivore software preloaded. The server is placed on the ISP's trunk to read header information for any traffic going to or coming from the suspect. This was rather common at numerous ISPs after September 11, 2001.

Many people have been concerned about the use of Carnivore and its ability to intercept all traffic, mostly due to privacy issues. They are concerned about how Carnivore works, how it could be misused by law enforcement, and the privacy debate over cable taps in general.

Carnivore is an Internet wiretap designed by the U.S. Federal Bureau of Investigation (FBI). It is designed with the special needs of law enforcement in mind. For example, some court orders might allow a pen-register monitoring of just the From/To e-mail addresses, whereas other court orders might allow a full capture of the e-mail. A summary of Carnivore's features can be seen within the configuration program shown in Figure 1.3.

Continued

Figure 1.3 Carnivore Configuration Program



The features are:

- **Filter sets** The settings are saved in configuration files; the user can quickly change the monitoring by selecting a different filter set.
- **Network adapters** A system may have multiple network adapters; only one can be selected for sniffing at a time.
- **Archive file size** A limit can be set on how much data is captured; by default, it fills up the disk.
- **Total memory usage** Network traffic may come in bursts faster than it can be written to disk; memory is set aside to buffer the incoming data.
- **Fixed IP address** All traffic to/from a range of IP addresses can be filtered. For example, the suspect may have a fixed IP address of 1.2.3.4 assigned to their cable modem. The FBI might get a court order allowing them to sniff all of the suspect's traffic.

Continued

- **Protocols to capture** Typically, a court order will allow only specific traffic to be monitored, such as SMTP over TCP. In Pen mode, only the headers are captured.
- **Data text strings** This is the Echelon feature that looks for keywords in traffic. A court order must specify exactly what is to be monitored, such as an IP address or e-mail account. Such wide-open keyword searches are illegal in the United States. The FBI initially denied that Carnivore had this feature.
- **Ports** A list of TCP and UDP ports can be specified. For example, if the FBI has a court order allowing e-mail capture, they might specify the e-mail ports of 25 (SMTP), 110 (POP3), and 143 (IMAP).
- **SMTP e-mail addresses** A typical scenario is where Carnivore monitors an ISP's e-mail server, discarding all e-mails except those of the suspects. An e-mail session is tracked until the suspect's e-mail address is seen, then all the packets that make up the e-mail are captured.
- **Dynamic IP addresses** When users dial up the Internet, they are logged in via the RADIUS protocol, which then assigns them an IP address. Normally, the FBI will ask the ISP to reconfigure their RADIUS servers to always assign the same IP address to the suspect, and will then monitor all traffic to/from that IP address. Note: if you are a dial-up user and suspect the FBI is after you, check to see if your IP address is the same every time you dial up. Sometimes this isn't possible. Carnivore can be configured to monitor the RADIUS protocol and dynamically discover the new IP address assigned to the suspect. Monitoring begins when the IP address is assigned, and stops when it is unassigned.

The FBI developed Carnivore because other existing utilities do not meet the needs of law enforcement. When an e-mail is sent across the wire, it is broken down into multiple packets. A utility like mailsnarf will reassemble the e-mail back into its original form. This is bad because the suspect's defense attorneys will challenge its accuracy: Did a packet get dropped somewhere in the middle that changes the meaning of the e-mail? Did a packet from a different e-mail somehow get inserted into the message? By capturing the raw packets rather than reassembling

Continued

them, Carnivore maintains the original sequence numbers, ports, and timestamps. Any missing or extra packets are clearly visible, allowing the FBI to defend the accuracy of the system.

Another problem that the FBI faces is minimization of the sniffed data. When the FBI wiretaps your line, they must assign an agent to listen in. If somebody else uses your phone (like your spouse or kids), they are required to turn off the tape recorders. In much the same way, Carnivore is designed to avoid capturing anything that does not belong to the suspect. A typical example would be using Carnivore to monitor the activities of a dial-up user. Carnivore contains a module to monitor the RADIUS traffic that is used by most ISPs to authenticate the user and assign a dynamic IP address. This allows Carnivore to monitor only that user without intercepting any other traffic.*

The following websites have more information on Carnivore:

- www.fbi.gov
- www.robertgraham.com/pubs/carnivore-faq.html
- www.stopcarnivore.org

*Excerpt from Robert Graham's chapter in *Hack Proofing Your Network, Second Edition*. Syngress Publishing 1-928994-70-9.

How Does It Work?

This section provides an overview of how all of this sniffing takes place. It gives you a little background on how networks and protocols work; however, there are many excellent resources out there that fill entire books themselves! The most popular and undoubtedly one of the best resources is Richard Stevens' "TCP/IP Illustrated, Vol. 1 – 3".

Explaining Ethernet

Ethernet is the most popular protocol standard used to enable computers to communicate. A protocol is like speaking a particular language. Ethernet was built around a principle of a shared medium where all computers on the local network segment share the same cable. It is known as a *broadcast* protocol because when a computer has information to send, it sends that data out to all other computers on the same network segment. This information is divided up into

manageable chunks called packets. Each packet has a header, which is like an envelope containing the addresses of both the destination and source computers. Even though this information is sent out to all computers on a segment, only the computer with the matching destination address will respond. All of the other computers on the network still see the packet, but if they are not the intended receiver they will disregard and discard it, unless a computer is running a sniffer. When you are running a sniffer, the packet capture driver that we mentioned earlier will put the computer's NIC into what is known as promiscuous mode. This means that the sniffing computer will be able to see all of the traffic on the segment regardless of who it is being sent to. Normally computers run in non-promiscuous mode, listening for information only designated for themselves. However, when a NIC is in promiscuous mode it can see conversations to and from all of its neighbors.

Ethernet addresses are known as Media Access Control (MAC) addresses, hardware addresses, or sometimes just Ethernet addresses. Since many computers may share a single Ethernet segment, each must have an individual identifier. These identifiers are hard-coded on to the NIC. A MAC address is a 48-bit number, also stated as a 12-digit hexadecimal number. This number is broken down into two halves, the first 24-bits identify the vendor of the Ethernet card, and the second 24-bits is a serial number assigned by the vendor.

The following steps will allow you to view your NIC's MAC address:

- **Windows 9x** Access **Start | Run**, and type **wiipcfg.exe**. The MAC address will be listed as “Adapter Address”.
- **Windows NT/2000/XP** Access the command line and type **ipconfig /all**. The MAC address will be listed as “Physical Address”.
- **Linux and Solaris** Type **ifconfig -a** at the command line. The MAC address will be listed as “HWaddr” on Linux and “ether” on Solaris.

You can also view the MAC addresses of other computers that you have communicated with recently, by using the command **arp -a**. More will be discussed about this in the “Defeating Switches” section.

MAC addresses are unique, and no two computers should have the same one. However, this is not always the case. Occasionally there could be a manufacturing error that would cause more than one network interface card to have the same MAC address, but mostly, people will change their MAC addresses on purpose. This can be done with a program, such as **ifconfig**, that will allow you to fake your MAC address. Faking your MAC address is also called *spoofing*. Also, some

adapters allow you to use a program to reconfigure the runtime MAC address. And lastly with the right tools and skill you can physically re-burn the address into the network interface card.

NOTE

Spoofting is the altering of network packet information such as the IP source address, MAC address, or even an e-mail address. This is often done to masquerade as another device in order to exploit a trust relationship, or to make tracing the source of attacks difficult. Address spoofing is also used in denial of service (DoS) attacks, such as Smurf, where the return address of network requests are spoofed to be the IP address of the victim.

Understanding the OSI model

The International Standards Organization (ISO) developed the Open Systems Interconnection (OSI) model in the early 1980's to describe how network protocols and components work together. It divides network functions into seven layers, and each layer represents a group of related specifications, functions, and activities.

The layers of the OSI model are:

- **Application layer** This topmost layer of the OSI model is responsible for managing communications between network applications. This layer is not the application program itself, although some applications may have the ability and the underlying protocols to perform application layer functions. For example, a Web browser is an application, but it is the underlying Hypertext Transfer Protocol (HTTP) protocol that provides the application layer functionality. Examples of application layer protocols include File Transfer Protocol (FTP), Simple Network Management Protocol (SNMP), Simple Mail Transfer Protocol (SMTP), and Telnet.
- **Presentation layer** This layer is responsible for data presentation, encryption, and compression.

- **Session layer** The session layer is responsible for creating and managing sessions between end systems. The session layer protocol is often unused in many protocols. Examples of protocols at the session layer include NetBIOS and Remote Procedure Call (RPC).
- **Transport layer** This layer is responsible for communication between programs or processes. Port or socket numbers are used to identify these unique processes. Examples of transport layer protocols include: TCP, UDP, and Sequenced Packet Exchange (SPX).
- **Network layer** This layer is responsible for addressing and delivering packets from the source computer to the destination computer. The network layer takes data from the transport layer and wraps it inside a packet or datagram. Logical network addresses are generally assigned to computers at this layer. Examples of network layer protocols include IP and Internetwork Packet Exchange (IPX). Devices that work at this layer are routers and Layer 3 switches.
- **Data link layer** This layer is responsible for delivering frames between NICs on the same physical segment. Communication at the data link layer is generally based on MAC addresses. The data link layer wraps data from the network layer inside a frame. Examples of data link layer protocols include Ethernet, Token Ring, and Point-to-Point Protocol (PPP). Devices that operate at this layer include bridges and switches.
- **Physical layer** This layer defines connectors, wiring, and the specifications on how voltage and bits pass over the cabled or wireless media. Devices at this layer include repeaters, concentrators, hubs, and cable taps. Devices that operate at the physical layer do not have an understanding of network paths.

NOTE

The terms *frame* and *packet* tend to be used interchangeably when talking about network traffic. However, the difference lies in the various layers of the OSI model. A frame is a unit of transmission at the data link layer. A packet is a unit of transmission at the network layer, however many people use the term packet to refer to data at any layer.

The OSI model is very generic and can be used to explain virtually any network protocol. Various protocol suites are often mapped against the OSI model for this purpose. A solid understanding of the OSI model aids tremendously in network analysis, comparison, and troubleshooting. However, it is also important to remember that not all protocols map nicely to the OSI model. For example, TCP/IP was designed to map to the U.S. Department of Defense (DoD) model. In the 1970s, the DoD developed its four-layer model. The core Internet protocols adhere to this model.

The DoD model is merely a condensed version of the OSI model. Its four layers are:

- **Process layer** This layer defines protocols that implement user-level applications such as mail delivery, remote login, and file transfer.
- **Host-to-host layer** This layer handles the connection, data flow management, and retransmission of lost data.
- **Internet layer** This layer is responsible for delivering data from source host to destination host across a set of different physical networks that connect the two machines.
- **Network access layer** This layer handles the delivery of data over a particular hardware media.

Notes from the Underground...

The TCP/IP Protocols

You will be seeing a lot of references in this book to TCP/IP and its associated protocols, specifically IP, TCP, and UDP. TCP/IP, developed by the Defense Advanced Research Projects Agency (DARPA), is the most widely used routed protocol today. IP is a Layer 3 protocol that contains addressing and control information that allows packets to be routed. IP is a connectionless protocol; therefore, it provides unreliable best-effort packet delivery service. Since IP only provides best-effort delivery, a packet may be discarded during transmission. All IP packets consist of a header and a payload (data from upper layers).

At the transport layer of the TCP/IP stack, the two commonly used protocols are TCP and UDP. The headers for both of these protocols

Continued

include a source and destination port number, which are used to determine the application or process that the TCP segment or UDP datagram originate from and destined to. TCP is a connection-oriented protocol, and UDP is a connectionless protocol. The TCP header includes sequence and acknowledgment numbers for reliable delivery. When IP needs reliable, guaranteed transfers it depends on TCP to provide this functionality.

Since TCP is a connection-oriented protocol it creates a dialog between the two communicating hosts to establish a connection. This is known as the three-way handshake. It starts by Host A sending a *SYN* packet to Host B letting it know that it wants to talk. Host B then responds with a *SYN/ACK*, saying that it is available to talk. Host A then finalizes the connection with an *ACK*.

TCP can also use the sliding window principle. The sliding window algorithm allows a buffer to be placed between the application program and the network data flow. Data received from the network is placed into this buffer until the application is ready to read it. The window is the amount of data that can be fetched into the buffer before an acknowledgment must be sent. Examples of applications that use TCP include FTP, Telnet, Network File System (NFS), SMTP, HTTP, Domain Name System (DNS), and Network News Transfer Protocol (NNTP). Examples of applications that use UDP include DNS, Routing Information Protocol (RIP), NFS, SNMP, and Dynamic Host Configuration Protocol/Boot Protocol (DHCP/BOOTP). As you can see, some applications (such as DNS and NFS) can use both protocols.

Notes from the Underground...

Writing Your Own Sniffer

There is an excellent paper titled “Basic Packet-Sniffer Construction from the Ground Up” by Chad Renfro located at www.unixgeeks.org/security/newbie/security/sniffer/sniffer_construction.txt. In this paper he presented a very basic 28-line packet sniffer written in C, called `sniff.c`. Even if you aren’t a programmer, Chad explains the program line by line in an

Continued

easy to understand manner. The program demonstrates the use of the RAW_SOCKET device to read TCP packets from the network and print basic header information to std_out. For simplicity, the program operates in non-promiscuous mode, so you would first need to put your interface in promiscuous mode by using the **ifconfig eth0 promisc** command.

There is also a header file that has to be copied into the same directory as sniff.c. It provides standard structures to access the IP and TCP fields. The structures identify each field in the IP and TCP header. It contains more information than what the sniff.c actually uses, but it least it is there to build upon.

To run the program, copy the sniff.c and headers.h into the same directory, and enter the command **gcc -o sniff sniff.c**. This will compile the program and create an executable file called sniff, which can be run by typing **./sniff**. The following text shows the output of the sniff program when I attempted a TELNET and FTP connection:

```
Bytes received :::      48
Source address ::: 192.168.1.1
IP header length ::: 5
Protocol ::: 6
Source port ::: 1372
Dest port  ::: 23
Bytes received :::      48
Source address ::: 192.168.1.1
IP header length ::: 5
Protocol ::: 6
Source port ::: 1374
Dest port  ::: 21
```

Once you are done capturing data, you can end the program by typing **CTRL-C**. You may also want to remove your interface from promiscuous mode by typing the command **ifconfig eth0 -promisc**.

CSMA/CD

Ethernet uses the Carrier Sense Multiple Access/Collision Detection (CSMA/CD) protocol for devices on the network to exchange data. The term