Chapter 1

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 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, but you can't connect to the local network or the Internet either. What should you do? 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?

Now consider this scenario. You go to your main network 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 Transmission Control Protocol (TCP) packets (destined for port 25) with various Internet Protocol (IP) addresses. You investigate and learn that there is a virus on the network that spreads through e-mail, and immediately apply access filters to block these packets from entering or exiting your network. Thankfully, you were able to contain the problem relatively quickly because of your knowledge and use of your network analyzer.

What Is Network Analysis and Sniffing?

Network analysis (also known as traffic analysis, protocol analysis, sniffing, packet analysis, eavesdropping, and so on) is the process of capturing network traffic and inspecting it closely to determine what is happening on the network. A network analyzer decodes the data packets of common protocols and displays the network traffic in readable format. A *sniffer* is a program that monitors data traveling over a network. Unauthorized sniffers are dangerous to network security because they are difficult to detect and can be inserted almost anywhere, which makes them a favorite weapon of hackers.

A network analyzer can be a standalone hardware device with specialized software, or software that is installed on a desktop or laptop computer. The differences between network analyzers depend on features such as the number of supported protocols it can decode, the user interface, and its graphing and statistical capabilities. Other differences include inference capabilities (e.g., expert analysis features) and the quality of packet decodes. Although several network analyzers decode the same protocols, some will work better than others for your environment.

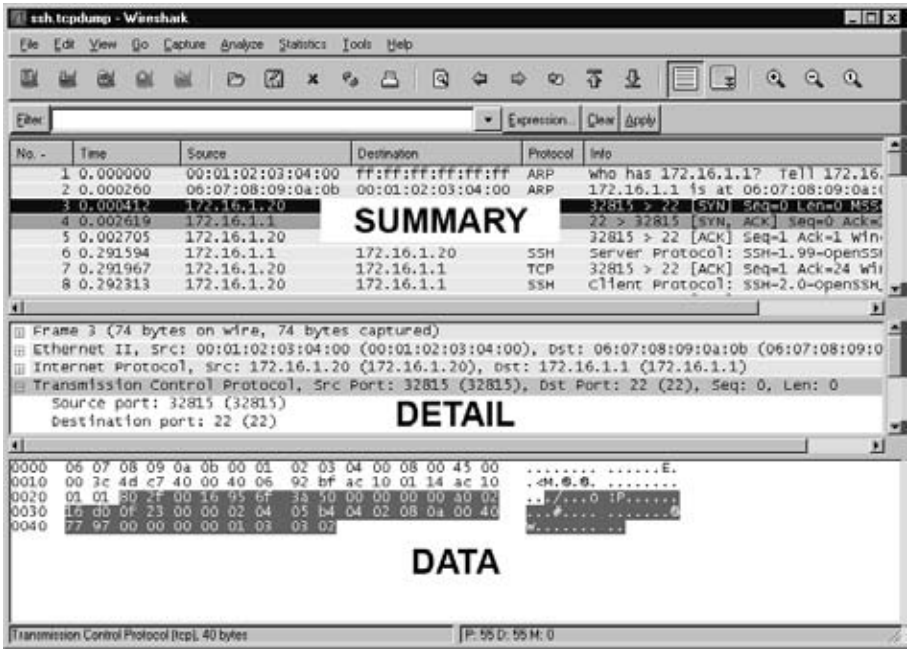
NOTE

The "Sniffer™" trademark, (owned by Network General) refers to the Sniffer product line. In the computer industry, "sniffer" refers to a program that captures and analyzes network traffic.

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

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

Figure 1.1 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 (OSes) and network interface cards (NICs). However, some hardware network analyzers offer additional benefits such as analyzing hardware faults (e.g., cyclic redundancy check (CRC) errors, voltage problems, cable problems, jitter, jabber, negotiation errors, and so on). Some network analyzers only support Ethernet or wireless adapters, while others support multiple adapters and allow users to customize their configurations. Depending on the situation, you may also need a hub or a cable tap to connect to the existing cable.
- **Capture Driver** This is the part of the network analyzer that is responsible for capturing raw network traffic from the cable. It filters out the traffic that you want to keep and stores the captured data in a buffer. This is the core of a network analyzer—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 (e.g., a "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 it to find network performance issues, and network intrusion detection systems (IDSes) use it to look for signs of intruder activity.
- **Decode** This component displays the contents (with descriptions) of the network traffic so that it is readable. Decodes are specific to each protocol, thus network analyzers vary in the number of decodes they currently support. However, new decodes are constantly being added to network analyzers.

NOTE

Jitter is the term that is used to describe the random variation of signal timing (e.g., electromagnetic interference and crosstalk with other signals can cause jitter). *Jabber* is the term that is used to describe when a device is improperly handling electrical signals, thus affecting the rest of the network (e.g., faulty NICs can cause jabber).

Who Uses Network Analysis?

System administrators, network engineers, security engineers, system operators, and programmers all use network analyzers, which are invaluable tools for diagnosing and troubleshooting network problems, system configuration issues, and application difficulties. Historically, network analyzers were dedicated hardware devices that were expensive and difficult to use. However, new advances in technology have allowed for the development of software-based network analyzers, which make it more convenient and affordable for administrators to effectively troubleshoot a network. It also brings the capability of network analysis.

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

A network analyzer is used for:

- Converting the binary data in packets to 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 faulty network cards
- Discovering the origin of virus outbreaks or Denial of Service (DoS) attacks
- 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
- Reverse-engineering protocols 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 a network. Network intruders use sniffing to capture confidential information, and the terms *sniffing* and *eavesdropping* are often associated with this practice. However, sniffing is becoming a non-negative term; most people use the terms sniffing and network analysis interchangeably.

Using a sniffer in an illegitimate way is considered a *passive attack*, because it does not directly interface or connect to any other systems on the network. A sniffer can also be installed as part of the compromise of a computer on a network using an *active attack*. The passive nature of sniffers is what makes detecting them difficult. (The methods used to detect sniffers are detailed later in this chapter.)

Intruders use sniffers on networks for:

- Capturing cleartext usernames and passwords
- Discovering the usage patterns of the users on a network
- Compromising proprietary information
- Capturing and replaying Voice over IP (VoIP) telephone conversations
- Mapping the layout of a network
- Passive OS fingerprinting

The above are all illegal uses of a sniffer unless you are a penetration tester whose job is to find and report these types of weaknesses.

For sniffing to occur, an intruder must first gain access to the communication cable of the systems of interest, which means being on the same shared network segment or tapping into the cable somewhere between the communication path. If the intruder is not physically present at the target system or communications access point (AP), there are still ways to sniff network traffic, including:

- Breaking into a target computer and installing remotely controlled sniffing software.
- Breaking into a communications access point (e.g., an Internet Service Provider [ISP]) and installing sniffing software.
- Locating a system at the ISP that already has sniffing software installed.
- Using social engineering to gain physical access to an ISP in order to install a packet sniffer.

- Having an inside accomplice at the target computer organization or the ISP install the sniffer.
- Redirecting or copying 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 an intruder by replacing commands and utilities and clearing log entries. Intruders also install other programs such as sniffers, key loggers, and backdoor access software. Windows sniffing can be accomplished as part of a Remote Admin Trojan (RAT) such as SubSeven or Back Orifice. Intruders often use sniffing programs that are configured to detect specific things (e.g., 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, File Transfer Protocol (FTP), Post Office Protocol version 3 (POP3), Internet Message Access Protocol (IMAP), Simple Mail Transfer Program (SMTP), Hypertext Transfer Protocol (HTTP), Remote Login (rlogin), and Simple Network Management Protocol (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/srec/.puta*. Another example of a rootkit is Linux Rootkit 5 (Lrk5), which installs with the linsniff sniffer.

Intruders may also use sniffer programs to control back doors (This practice isn't quite "common," but it isn't unheard of). One method is to install a sniffer on a target system that listens for specific information and then sends the backdoor control information to a neighboring system. This type of backdoor control is hard to detect, because of the passive nature of sniffers.

cd00r is an example of a backdoor sniffer that operates in non-promiscuous mode, making it even harder to detect. Using a product like Fyodor's Nmap (http://insecure.org/nmap) to send a series of TCP synchronize (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*.

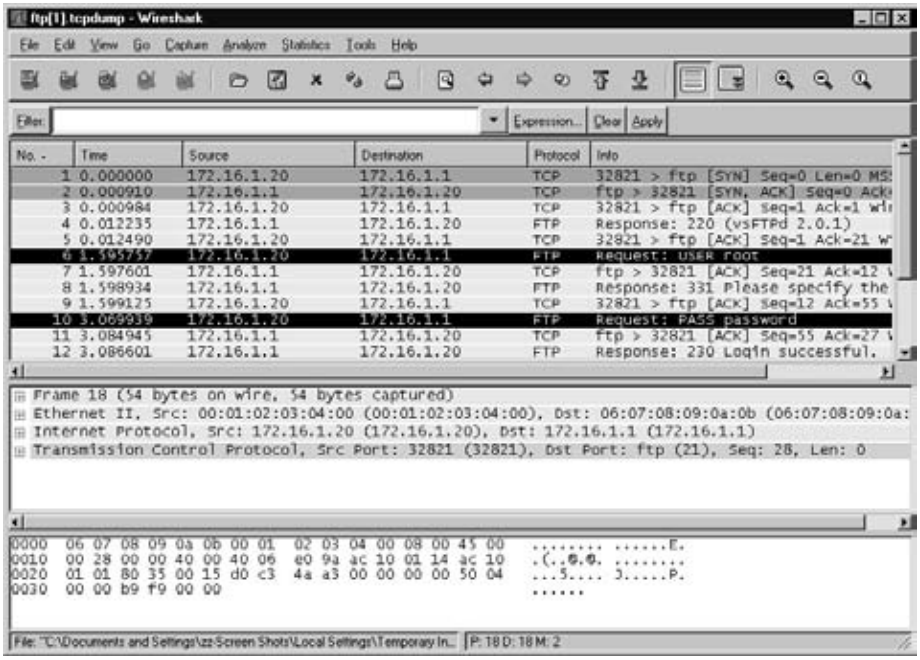
NOTE

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

What Does Sniffed Data Look Like?

The easiest way to grasp the concept of a sniffer is to watch one in action. Figure 1.2 shows a capture of a simple FTP session from a laptop to a Linux system. The two highlighted packets show how easy it is to sniff the username and password (i.e., "root" and "password").

Figure 1.2 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 are:

- **Wireshark** Wireshark is one of the best sniffers available and is being developed as a free, commercial-quality sniffer. It has numerous features, a nice graphical user interface (GUI), decodes over 400 protocols, and is actively being developed and maintained. It runs on UNIX-based systems, Mac OS X, and Windows. This is a great sniffer to use in a production environment, and is available at www.wireshark.org.

- **WinDump** WinDump is the Windows version of tcpdump, and is available at www.winpcap.org/windump. It uses the WinPcap library and runs on Windows 95, 98, ME, NT, 2000, and XP.
- **Network General Sniffer** A Network General Sniffer is one of the most popular commercial sniffers available. Now a suite of enterprise network capture tools, there is an entire Sniffer product line at www.networkgeneral.com.
- **Windows 2000 and 2003 Server Network Monitor** Both the Windows 2000 Server and the Windows 2003 Server have a built-in program to perform network analysis. It is located in the "Administrative Tools" folder, but is not installed by default; therefore, you have to add it from the installation CD.
- **EtherPeek** EtherPeek is a commercial network analyzer developed by WildPackets. Versions for both Windows and Mac, and other network analysis products can be found at www.wildpackets.com.
- **Tcpdump** Tcpdump is the oldest and most commonly used network sniffer, and was developed by the Network Research Group (NRG) of the Information and Computing Sciences Division (ICSD) at Lawrence Berkeley National Laboratory (LBNL). It is command line-based and runs on UNIX-based systems, including Mac OS X. It is actively developed and maintained at www.tcpdump.org.
- **Snoop** Snoop is a command-line network sniffer that is included with the Sun Solaris OS.
- **Snort** Snort is a network IDS that uses network sniffing, and is actively developed and maintained at www.snort.org. For more information, refer to *Nessus, Snort, & Ethereal Power Tools: Customizing Open Source Security Applications* (Syngress Publishing: 1597490202) and *Snort Intrusion Detection and Prevention Toolkit* (Syngress, ISBN: 1597490997).
- **Dsniff** Dsniff is a very popular network-sniffing package. It is a collection of programs that are used to specifically sniff for interesting data (e.g., passwords) and to facilitate the sniffing process (e.g., evading switches). It is actively maintained at www.monkey.org/~dugsong/dsniff.
- **Ettercap** Ettercap was specifically designed to sniff a switched network. It has built-in features such as password collecting, OS fingerprinting, and character injection, and runs on several platforms including Linux, Windows, and Solaris. It is actively maintained at ettercap.sourceforge.net.

- **Analyzer** Analyzer is a free sniffer that is used for the Windows OS. It is being actively developed by the makers of WinPcap and WinDump at Politecnico di Torino, and can be downloaded from analyzer.polito.it.
- **Packetyzer** Packetyzer is a free sniffer (used for the Windows OS) that uses Wireshark's core logic. It tends to run a version or two behind the current release of Wireshark. It is actively maintained by Network Chemistry at www.networkchemistry.com/products/packetyzer.php.
- **MacSniffer** MacSniffer is specifically designed for the Mac OS X environment. It is built as a front-end for tcpdump. The software is shareware and can be downloaded from personalpages.tds.net/~brian_hill/macsniffer.html.

How Does It Work?

This section provides an overview of how sniffing takes place, and gives background information on how networks and protocols work. However, there are many other excellent resources available, including the most popular and undoubtedly one of the best written, 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 the 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 it sends that data to all other computers on the same network segment. This information is divided up into manageable chunks called *packets*, and each packet has a header 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 responds. All of the other computers on the network still see the packet, but if they are not the intended receiver they disregard it, unless a computer is running a sniffer. When running a sniffer, the packet capture driver puts the computer's NIC into *promiscuous mode*. This means that the sniffing computer can 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 designated only for themselves. However, when a NIC is in promiscuous mode, it can see conversations to and from all of its neighbors.

Ethernet addresses are also known as Media Access Control (MAC) addresses and hardware addresses. Because many computers may share a single Ethernet segment, each one must have an individual identifier hard-coded onto the NIC. A MAC address is a 48-bit number, which is 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 comprise a serial number assigned by the vendor.

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

- **Windows 9x/ME** Access **Start | Run** and type **winipcfg.exe**. The MAC address will be listed as the "Adapter Address."
- **Windows NT, 2000, XP, and 2003** Access the command line and type **ipconfig /all**. The MAC address will be listed as the "Physical Address."
- **Linux and Solaris** Type **ifconfig –a** at the command line. The MAC address will be listed as the "HWaddr" on Linux and as "ether" on Solaris.
- **Macintosh OS X** Type **ifconfig –a** at the Terminal application. The MAC address will be listed as the "Ether" label.

You can also view the MAC addresses of other computers that you have recently communicated with, by typing the command **arp –a**. (Discussed in more detail in the "Defeating Switches" section.)

MAC addresses are unique, and no two computers have the same one. However, occasionally a manufacturing error may occur that causes more than one NIC to have the same MAC address. Thus, most people change their MAC addresses intentionally, which can be done with a program (e.g., *ifconfig*) that allows you to fake your MAC address. Faking your MAC address (and other types of addresses) is also known as *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 reburn the address into the NIC.

NOTE

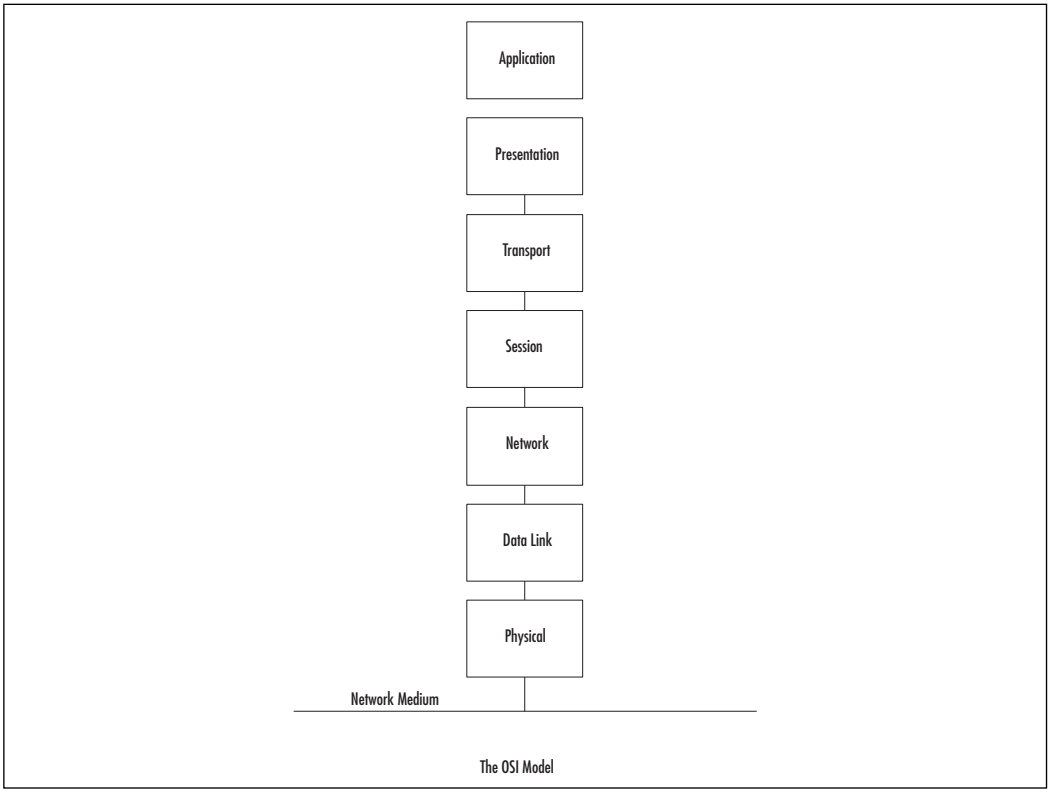
Spoofing is the process of altering network packet information (e.g., the IP source address, the MAC address, or the e-mail address). This is often done to masquerade as another device in order to exploit a trust relationship or to make tracing the source of attacks difficult. Address spoofing is also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

Understanding the Open Systems Interconnection Model

The International Standards Organization (ISO) developed the Open Systems Interconnection (OSI) model in the early 1980s to describe how network protocols and components work together. It divides network functions into seven layers, each layer representing a group of related specifications, functions, and activities (see Figure 1.3). Although complicated at first, the terminology is used extensively in networking, systems, and development communities.

The following sections define the seven layers of the OSI model.

Figure 1.3 Seven boxes corresponding to OSI model.



The following sections define the seven layers of the OSI model.

NOTE

The OSI model is not necessarily reflective of the way that applications and OSes are actually written. In fact, some security tools use the differences in protocol implementations to extract information from computers (including their OSes) and specific patches and services packs that may have been installed.

"We still talk about the seven layers model, because it's a convenient model for discussion, but that has absolutely zero to do with any real-life software engineering. In other words, it's a way to talk about things, not to implement them. And that's important. Specs are a basis for talking about things. But they are not a basis for implementing software."

– Linus Torvalds, project coordinator for the Linux kernel, in an e-mail dated September 29, 2005.

Layer 1: Physical

The first layer of the OSI model is the *Physical* layer, which specifies the electrical and mechanical requirements for transmitting data bits across the transmission medium (cable or airwaves). It involves sending and receiving the data stream on the carrier, whether that carrier uses electrical (cable), light (fiber optic), radio, infrared, or laser (wireless) signals. The Physical layer specifications include:

- Voltage changes
- The timing of voltage changes
- Data rates
- Maximum transmission distances
- The physical connectors to the transmission medium (plug)
- The topology or physical layout of the network

Many complex issues are addressed at the Physical layer, including digital vs. analog signaling, baseband vs. broadband signaling, whether data is transmitted synchronously or asynchronously, and how signals are divided into channels (multiplexing).

Devices that operate at the Physical layer deal with signaling (e.g., transceivers on the NIC), repeaters, basic hubs, and simple connectors that join segments of cable). The data handled by the Physical layer is in bits of 1s (ones) and 0s (zeros), which

are represented by pulses of light or voltage changes of electricity, and by the state of those pulses (*on* generally representing 1 and *off* generally representing *0*).

How these bits are arranged and managed is a function of the Data Link layer (layer 2) of the OSI model.

Layer 2: Data Link

Layer 2 is the *Data Link* layer, which is responsible for maintaining the data link between two computers, typically called *hosts* or *nodes*. It also defines and manages the ordering of bits to and from packets. *Frames* contain data arranged in an organized manner, which provides an orderly and consistent method of sending data bits across the medium. Without such control, the data would be sent in random sizes or configurations and the data on one end could not be decoded at the other end. The Data Link layer manages the physical addressing and synchronization of the data packets. It is also responsible for flow control and error notification on the Physical layer. Flow control is the process of managing the timing of sending and receiving data so that it doesn't exceed the capacity (speed, memory, and so on) of the physical connection. Since the Physical layer is only responsible for physically moving the data onto and off of the network medium, the Data Link layer also receives and manages error messaging related to the physical delivery of packets.

Network devices that operate at this layer include layer 2 switches (switching hubs) and bridges. A layer 2 switch decreases network congestion by sending data out only on the port that the destination computer is attached to, instead of sending it out on all ports. Bridges provide a way to segment a network into two parts and filter traffic, by building tables that define which computers are located on which side of the bridge, based on their MAC addresses.

The Data Link layer is divided into two sublayers: the Logical Link Control (LLC) sublayer and the MAC sublayer.

The MAC Sublayer

The MAC sublayer provides control for accessing the transmission medium. It is responsible for moving data packets from one NIC to another, across a shared transmission medium such as an Ethernet or fiber-optic cable.

Physical addressing is addressed at the MAC sublayer. Every NIC has a unique MAC address (also called the *physical address*) which identifies that specific NIC on the network. The MAC address of a NIC is usually burned into a read-only memory (ROM) chip on the NIC. Each manufacturer of network cards is provided a unique set of MAC addresses so that theoretically, every NIC that is manufactured has a

unique MAC address. To avoid any confusion, MAC addresses are permanently burned into the NIC's memory, which is sometimes referred to as the Burned-in Address (BIA).

NOTE

On Ethernet NICs, the physical or MAC address (also called the *hardware* address) is expressed as 12 hexadecimal digits arranged in pairs with colons between each pair (e.g., 12:3A:4D:66:3A:1C). The initial three sets of numbers represent the manufacturer, and the last three bits represent a unique NIC made by that manufacturer.

MAC refers to the method used to allocate network access to computers while preventing them from transmitting at the same time and causing data collisions. Common MAC methods include Carrier Sense Multiple Access/Collision Detection (CSMA/CD) used by Ethernet networks, Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) used by AppleTalk networks, and token passing used by Token Ring and Fiber Distributed Data Interface (FDDI) networks. (CSMA/CD is discussed later in this chapter.)

The LLC Sublayer

The LLC sublayer provides the logic for the data link; thus it controls the synchronization, flow control, and error-checking functions of the Data Link layer. This layer manages connection-oriented transmissions; however, connectionless service can also be provided by this layer. Connectionless operations are known as Class I LLC, whereas Class II can handle either connectionless or connection-oriented operations. With connection-oriented communication, each LLC frame sent is acknowledged. The LLC sublayer at the receiving end keeps up with the LLC frames it receives (also called Protocol Data Units [PDUs]); therefore, if it detects that a frame has been lost during transmission, it can send a request to the sending computer to start the transmission over again, beginning with the PDU that never arrived.

The LLC sublayer sits above the MAC sublayer, and acts as a liaison between the upper layers and the protocols that operate at the MAC sublayer (e.g., Ethernet, Token Ring, and so on). The LLC sublayer is defined by Institute of Electrical & Electronics Engineers (IEEE) 802.2. Link addressing, sequencing, and definition of Service Access Points (SAPs) also take place at this layer.

Layer 3: Network

The next layer is the *Network* layer (layer 3), which is where packets are sequenced and logical addressing is assigned. Logical addresses are nonpermanent, software-assigned addresses that can only be changed by administrators. The IP addresses used by the TCP/IP protocols on the Internet, and the Internet Package Exchange (IPX) addresses used by the IPX/Sequenced Packet Exchange (SPX) protocols on NetWare networks are examples of logical addresses. These protocol stacks are referred to as *routable* because they include addressing schemes that identify the network or subnet and the particular client on that network or subnet. Other network/transport protocols (e.g., NETBIOS Extended User Interface [NetBEUI]) do not have a sophisticated addressing scheme and thus cannot be routed between different types of networks.

NOTE

To understand the difference between physical and logical addresses, consider this analogy: A house has a physical address that identifies exactly where it is located. This is similar to the MAC address on a NIC.

A house also has a logical address assigned to it by the post office that consists of a street name and number. The post office occasionally changes the names of streets or renumbers the houses located on them. This is similar to the IP address assigned to a network interface.

The Network layer is also responsible for creating a virtual circuit (i.e., a logical connection, not a physical connection) between points or nodes. A node is a device that has a MAC address, which typically includes computers, printers, and routers. This layer is also responsible for routing, layer 3 switching, and forwarding packets. *Routing* refers to forwarding packets from one network or subnet to another. Without routing, computers can only communicate with computers on the same network. Routing is the key to the global Internet, and is one of the most important duties of the Network layer.

Finally, the Network layer provides additional levels of flow control and error control. As mentioned earlier, from this point on, the primary methods of implementing the OSI model architecture involve software rather than hardware.

Devices that operate at this layer include routers and layer 3 switches.

Layer 4: Transport

Layer 4 is the *Transport* layer, and is responsible for transporting the data from one node to another. It provides transparent data transfer between nodes, and manages the end-to-end flow control, error detection, and error recovery.

The Transport layer protocols initiate contact between specific ports on different host computers, and set up a virtual circuit. The transport protocols on each host computer verify that the application sending the data is authorized to access the network and that both ends are ready to initiate the data transfer. When this synchronization is complete, the data is sent. As the data is being transmitted, the transport protocol on each host monitors the data flow and watches for transport errors. If transport errors are detected, the transport protocol provides error recovery.

The functions performed by the Transport layer are very important to network communication. Just as the Data Link layer provides lower-level reliability and connection-oriented or connectionless communications, the Transport layer does the same thing but at a higher level. The two protocols most commonly associated with the Transport layer are the Transmission Control Protocol (TCP), which is connection-oriented and the User Datagram Protocol (UDP), which is connectionless.

NOTE

What's the difference between a connection-oriented protocol and a connectionless protocol? A connection-oriented protocol (e.g., TCP) creates a connection between two computers before sending the data, and then verifies that the data has reached its destination by using acknowledgements (ACKs) (i.e., messages sent back to the sending computer from the receiving computer that acknowledge receipt). Connectionless protocols send the data and trust that it will reach the proper destination or that the application will handle retransmission and data verification.

Consider this analogy: You need to send an important letter to a business associate that contains valuable papers. You call him before e-mailing the letter, to let him know that he or she should expect it (establishing the connection). A few days later your friend calls to let you know that he received the letter, or you receive the return receipt (ACK). This is how connection-oriented communication works. When mailing a postcard to a friend, you drop it in the mailbox and hope it gets to the addressee. You don't expect or require any acknowledgement. This is how connectionless communication works.

The Transport layer also manages the logical addressing of ports. Think of a port as a suite or apartment number within a building that defines exactly where the data should go.

Table 1.1 Commonly Used Internet Ports

Internet Protocol (IP) Port(s)	Protocol(s)	Description
80	TCP	HTTP, commonly used for Web servers
443	TCP	Hypertext Transfer Protocol Secure sockets (HTTPS) for secure Web servers.
53	UDP and TCP	Domain Name Server/Service (DNS) for resolving names to IP addresses
25	TCP	Simple Mail Transfer Protocol(SMTP), used for sending e-mail
22	TCP	The Secure Shell (SSH) protocol
23	TCP	Telnet, an insecure administration protocol
20 and 21	TCP	An insecure Fire Transfer Protocol (FTP)
135–139 and 445	TCP and UDP	Windows file sharing, login, and Remote Procedure Call (RPC)
500	UDP	Internet Security Association and Key Management Protocol (ISAKMP) key negotiation for Secure Internet Protocol (IPSec) virtual private networks (VPNs)
5060	UDP	Session Initiation Protocol (SIP) for some VoIP uses
123	UDP	Network Time Protocol (NTP) for network time synchronization

A computer may have several network applications running at the same time (e.g., a Web browser sending a request to a Web server for a Web page, an e-mail client sending and receiving e-mail, and a file transfer program uploading or downloading information to and from an FTP server). The mechanism for determining which incoming data packets belong to which application is the function of port numbers. The FTP protocol is assigned a particular port, whereas the Web browser

and e-mail clients use different protocols (e.g., HTTP and POP3 or Internet Message Access Protocol [IMAP]) that have their own assigned ports; thus the information intended for the Web browser doesn't go to the e-mail program by mistake. Port numbers are used by TCP and UDP.

Finally, the Transport layer deals with name resolution. Most users prefer to identify computers by name instead of by IP address (i.e., *www.microsoft.com* instead of 207.46.249.222), however, computers only interpret numbers, therefore, there must be a way to match names with numerical addresses. Name resolution methods such as the DNS solve this problem.

Layer 5: Session

After the Transport layer establishes a virtual connection, a communication session is made between two processes on two different computers. The Session layer (layer 5) is responsible for establishing, monitoring, and terminating sessions, using the virtual circuits established by the Transport layer.

The Session layer is also responsible for putting header information into data packets that indicates where a message begins and ends. Once header information is attached to the data packets, the Session layer performs synchronization between the sender's Session layer and the receiver's Session layer. The use of ACKs helps coordinate the transfer of data at the Session-layer level.

Another important function of the Session layer is controlling whether the communications within a session are sent as full-duplex or half-duplex messages. Half-duplex communication goes in both directions between the communicating computers, but information can only travel in one direction at a time (e.g., radio communications where you hold down the microphone button to transmit, but cannot hear the person on the other end). With full-duplex communication, information can be sent in both directions at the same time (e.g., a telephone conversation, where both parties can talk and hear one another at the same time).

Whereas the Transport layer establishes a connection between two machines, the Session layer establishes a connection between two processes. An application can run many processes simultaneously to accomplish the work of the application.

After the Transport layer establishes the connection between the two machines, the Session layer sets up the connection between the application process on one computer and the application process on another computer.

Layer 6: Presentation

Data translation is the primary activity of the Presentation layer (layer 6). When data is sent from a sender to a receiver, it is translated at the Presentation layer (i.e., the

sender's application passes data down to the Presentation layer, where it is changed into a common format). When the data is received on the other end, the Presentation layer changes it from the common format back into a format that is useable by the application. Protocol translation (i.e., the conversion of data from one protocol to another so that it can be exchanged between computers using different platforms or OSes) takes place here.

The Presentation layer is also where *gateway* services operate. Gateways are connection points between networks that use different platforms or applications (e.g., e-mail gateways, Systems Network Architecture (SNA) gateways, and gateways that cross platforms or file systems). Gateways are usually implemented via software such as the Gateway Services for NetWare (GSNW). Software redirectors also operate at this layer.

This layer is also where data compression takes place, which minimizes the number of bits that must be transmitted on the network media to the receiver. Data encryption and decryption also take place in the Presentation layer.

Layer 7 Application

The *Application* layer is the point at which the user application program interacts with the network. Don't confuse the networking model with the application itself. Application processes (e.g., file transfers or e-mail) are initiated within a user application (e.g., an e-mail program). Then the data created by that process is handed to the Application layer of the networking software. Everything that occurs at this level is application-specific (e.g., file sharing, remote printer access, network monitoring and management, remote procedure calls, and all forms of electronic messaging).

Both FTP and Telnet function within the Application layer, as does the Simple Mail Transfer Protocol (SMTP), Post Office Protocol (POP), and Internet Message access Protocol *(*IMAP), all of which are used for sending or receiving e-mail. Other Application-layer protocols include HTTP, Network News Transfer Protocol (NNTP), and Simple network Management Protocol (SNMP).

You have to distinguish between the protocols mentioned and the applications that might bear the same names, because there are many different FTP programs made by different software vendors that use the FTP to transfer files.

The OSI model is generic and can be used to explain all network protocols. Various protocol suites are often mapped against the OSI model for this purpose. A solid understanding of the OSI model aids in network analysis, comparison, and troubleshooting. However, it is important to remember that not all protocols map well to the OSI model (e.g., 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 a condensed version of the OSI model. Its four layers are:

- **Process Layer** This layer defines protocols that implement user-level applications (e.g., e-mail delivery, remote login, and file transfer).
- **Host-to-host Layer** This layer manages the connection, data flow management, and retransmission of lost data.
- **Internet Layer** This layer delivers data from the source host to the destination host across a set of physical networks that connect the two machines.
- **Network Access Layer** This layer manages the delivery of data over a particular hardware media.

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 that is located at www.packetstormsecurity.org /sniffers/Sniffer_construction.txt. In this paper, he presents a basic 28-line packet sniffer that is written in C, called *sniff.c*, which he explains line-by-line in an easy-to-understand manner. The program demonstrates how to use the *raw_socket* device to read TCP packets from the network, and how to print basic header information to *stdout*. For simplicity, the program operates in non-promiscuous mode; therefore, you first need to put your interface in promiscuous mode using the **ifconfig eth0 promisc** command.

There is also a header file that must be copied into the same directory as *sniff.c*, that provides standard structures to access the IP and TCP fields in order to identify each field in the IP and TCP header.

To run the program, copy the *sniff.c* and *headers.h* files into one directory, and enter **gcc -o sniff sniff.c**. This compiles the program and creates an executable file called *sniff*, which is run by typing **./sniff**. The following text shows the output of the sniff program when a Telnet and FTP connection was attempted:

```
Bytes received :::    48
Source address ::: 192.168.1.1
IP header length ::: 5
```

Continued

```
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 **ifconfig eth0 –promisc** command.

CSMA/CD

Ethernet uses the CSMA/CD protocol in order for devices to exchange data on the network. The term *multiple access* refers to the fact that many network devices attached to the same segment have the opportunity to transmit. Each device is given an equal opportunity; no device has priority over another. *Carrier sense* describes how an Ethernet interface on a network device listens to the cable before transmitting. The network interface ensures that there are no other signals on the cable before it transmits, and listens while transmitting to ensure that no other network device transmits data at the same time. When two network devices transmit at the same time, a *collision* occurs. Because Ethernet interfaces listen to the media while they are transmitting, they can identify the presence of others through *collision detection*. If a collision occurs, the transmitting device waits for a small, random amount of time before retransmitting. This function is known as *random backoff*.

Traditionally, Ethernet operation has been half-duplex, which means that an interface can either transmit or receive data, but not at the same time. If more than one network interface on a segment tries to transmit at the same time, a collision occurs per CSMA/CD. When a crossover cable is used to connect two devices, or a single device is attached to a switch port, only two interfaces on the segment need to transmit or receive; no collisions occur. This is because the transmit (TX) of device A is connected to the receive (RX) of device B, and the TX of B is connected to the RX of device A. The collision detection method is

no longer necessary, therefore, interfaces can be placed in full-duplex mode, which allows network devices to transmit and receive at the same time, thereby increasing performance.

The Major Protocols: IP, TCP, UDP, and ICMP

The next four protocols are at the heart of how the Internet works today.

NOTE

Other, different protocols are used across the Internet, and new protocols are constantly created to fulfill specific needs. One of these is Internet Protocol version 6 (IPv6), which seeks to improve the existing Internet protocol suite by providing more IP addresses, and by improving the security of network connections across the Internet using encryption. For more information on IPv6, see http://en.wikipedia.org/wiki/IPv6.

IP

The IP is a connectionless protocol that manages addressing data from one point to another, and fragments large amounts of data into smaller, transmittable packets. The major components of Internet Protocol datagrams are:

- **IP Identification (IPID)** Tries to uniquely identify an IP datagram.
- **Protocol** Describes the higher-level protocol contained within the datagram.
- **Time-to-live (TTL)** Attempts to keep datagrams and packets from routing in circles. When TTL reaches 0, the datagram is dropped. The TTL allows traceroute to function, identifying each router in a network by sending out datagrams with successively increasing TTLs, and tracking when those TTLs are exceeded.
- **Source IP Address** The IP address of the host where the datagram was created.
- **Destination IP Address** The destination of where the datagram should be sent.

IP Address Source Spoofing

It is possible to spoof any part of an IP datagram; however, the most commonly spoofed IP component is the source IP address. Also, not all protocols function completely with a spoofed source IP address (e.g., connection-oriented protocols such as TCP require handshaking before data can be transmitted, thereby reducing the effectiveness of spoofing-based attacks).

Spoofing can also be used as a DoS attack. If Network A sends a datagram to Network B, with a spoofed source IP host address on Network C, Network C will see traffic going to it that originates from Network B, perhaps without any indication that Network A is involved at all.

The best practice for network administrators is to ensure that the network can only originate packets with a proper Source IP address (i.e., an IP address in the network itself).

Internet Control Message Protocol

The Internet Control Message Protocol (ICMP) manages errors that occur between networks on the IP. The following are common types of ICMP messages:

- **Echo Request/Reply** Used by programs such as *ping* to calculate the delay in reaching another IP address.
- **Destination Unreachable: Network Unreachable and Port Unreachable** Sent to the source IP address of a packet when a network or port cannot be reached. This happens when a firewall rejects a packet or if there is a network problem. There are a number of subtypes of Destination Unreachable messages that are helpful at diagnosing communication issues.
- **Time Exceeded** Occurs when the TTL of a packet reaches 0.

TCP

TCP packets are connection-oriented, and are used most often to transmit data. The connection-oriented nature of TCP packets makes it a poor choice for source IP address spoofing. Many applications use TCP, including the Web (HTTP), e-mail (SMTP), FTP, SSH, and the Windows Remote Desktop Protocol (RDP).

The TCP Handshake

An important concept of the TCP is *handshaking*. Before any data can be exchanged between two hosts, they must agree to communicate. Host A sends a packet with the SYN flag set to Host B. If Host B is willing and able to communicate, it returns the SYN packet and adds an ACK flag. Host A begins sending data, and indicates to Host B that it also received the ACK. When the communication between the hosts ends, a packet with the FIN (finish) flag is sent, and a similar acknowledgement process is followed.

TCP Sequence

Another important component of TCP is *sequence identification*, where each packet sent is part of a sequence. Through these numbers, TCP handles complex tasks such as retransmission, acknowledgement, and order.

UDP

UDP packets are the connectionless equivalent to TCP, and are used for many purposes, the most important being that DNS uses UDP for most of its work. DNS finds out which IP address corresponds to which hostname (e.g., www.example.com is not routable as an IP address inside an IP datagram; however, through a DNS system it can find the IP address to route traffic to). Other uses of UDP include VoiP and many online games and streaming media types.

Hardware: Cable Taps, Hubs, and Switches

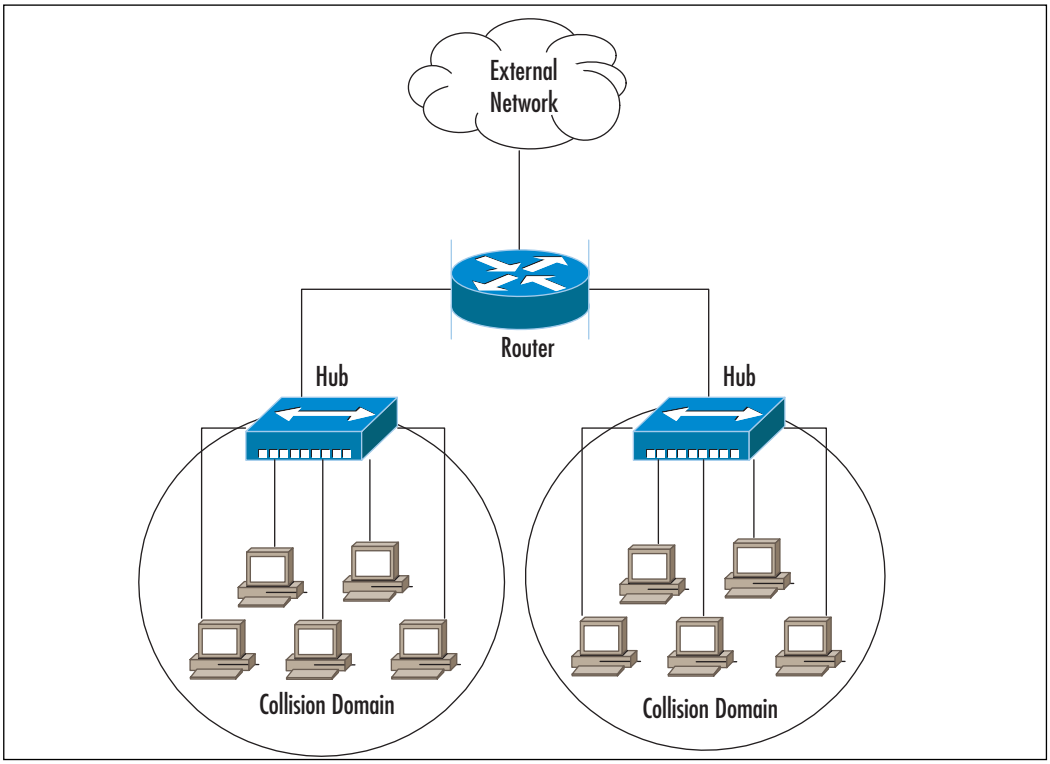
Cable taps are hardware devices that assist in connecting to a network cable. Test access points (Taps) use this device to access any cables between computers, hubs, switches, routers, and other devices. Taps are available in full- or half-duplex for 10, 100, and 1,000 Mbps Ethernet links. They are also available in various multi-port sizes. The following is a list of some popular cable tap products:

- Net Optics carries several types of network taps for copper and fiber cables, and is available at www.netoptics.com.
- The Finisar Tap family offers a variety of taps for copper and fiber cables, and is available at www.finisar.com/nt/taps.php.

A *hub* is a device that allows you to connect multiple hosts together on a shared medium (e.g., Ethernet). When a computer sends information, it travels into the hub and the hub forwards the information to all other computers connected to it. The computer that the information was intended for will recognize its own MAC address

in the packet header and accept the data. The area that the hub forwards all information to is known as a *collision domain* (also known as *broadcast domain*). A hub has only one collision domain for all traffic to share. Figure 1.4 shows a network architecture with collision domains related to hubs. Large collisions make sniffing easier and create performance issues such as bandwidth hogging or excessive traffic on the hub.

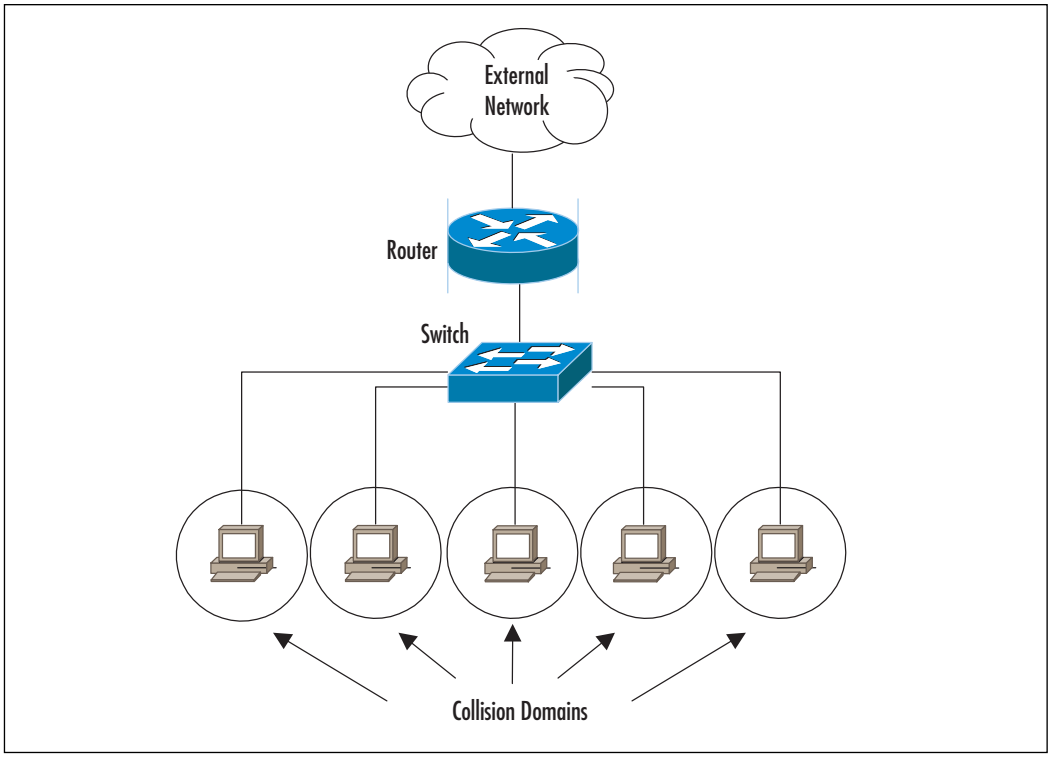
Figure 1.4 Hub Collision Domains



A switch is also used to connect computers together on a shared medium; however, when a switch receives information, it doesn't blindly send it to all other computers; it looks at the packet header to locate the destination MAC address, and maintains a list of all MAC addresses and corresponding ports on the switch that the computers are connected to. It then forwards the packets to the specified port. This narrows the collision domain to a single port (see Figure 1.5). This type of collision domain also provides a definite amount of bandwidth for each connection rather than a shared amount on a hub. Because the price of switches has fallen dramatically in the last few years, there is no reason not to replace hubs with switches, or to choose switches when purchasing new equipment. Also, some

of the more costly switches include better technology that makes them more resistant to sniffing attacks.

Figure 1.5 Switch Collision Domains



As you can see from the diagrams, hubs make sniffing easier and switches make sniffing more difficult. However, switches can be tricked, as discussed in the "Defeating Switches" section.

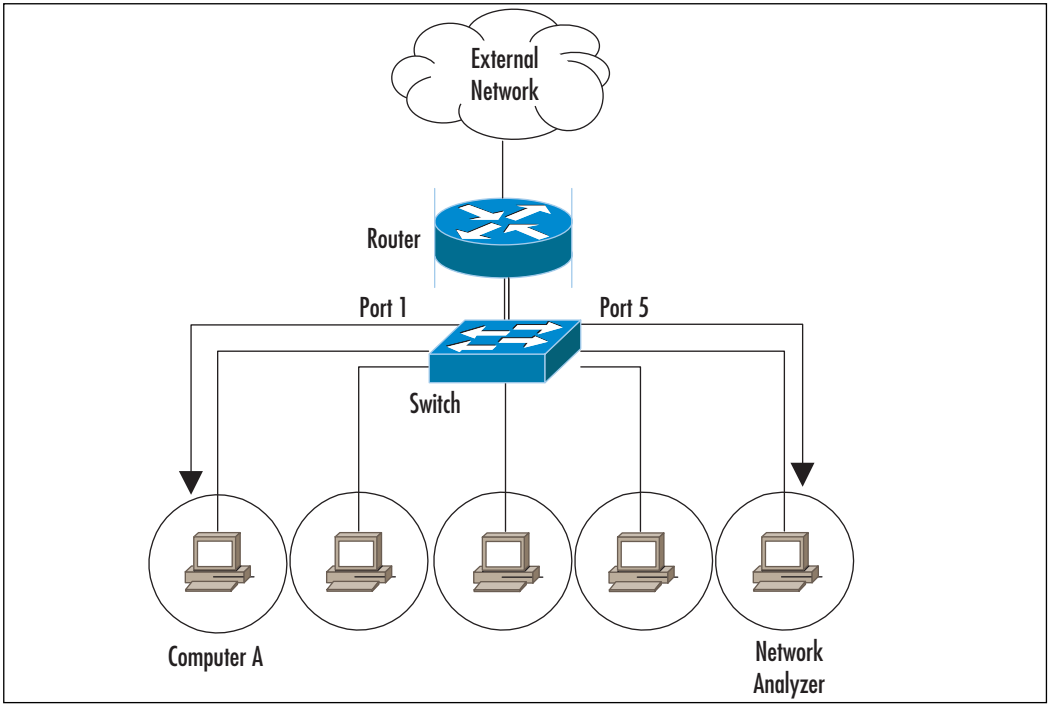
Port Mirroring

If you are working on a network that uses switches and you want to perform network analysis legitimately, you are in luck; most switches and routers come with *port mirroring* (also known as *port spanning*). To mirror ports, you have to configure the switch to duplicate the traffic from the port you want to monitor to the port you are connected to.

Using port spanning does not interfere with the normal operation of switches, but you should always check the documentation of the exact switch you are configuring and periodically check the device's logs. You won't affect the switch, but you

will increase the amount of traffic on a specific destination port, therefore, make sure your properly configured network analyzer is the destination port. Also, consult the documentation for your specific switch to learn the exact command to enable port mirroring (see Figure 1.6). The switch is configured to mirror all port 1 traffic to port 5, and the network analyzer sees all traffic to and from Computer A. Sometimes administrators mirror the uplink port on a switch, so that they can see all of the traffic to and from the switch and all of its ports.

Figure 1.6 Port Mirroring



NOTE

Span stands for Switched Port Analyzer. Cisco uses the word *span* to describe the concept of port mirroring. In Cisco terms, spanning a port is the same as mirroring a port.

Defeating Switches

As mentioned earlier, using switches on a network makes sniffing more difficult. In theory, you should only see traffic destined for you own computer on a switch; however, there are ways to circumvent its technology. The following list describes several ways in which a switch can be defeated:

- **Switch Flooding** Some switches can be made to act like a hub, where all packets are broadcast to all computers. This can be accomplished by overflowing the switch address table with a large number of fake MAC addresses (known as a device *failing open)*, thus removing all security provisions. Devices that *fail close* incorporate some type of security measure (e.g., shutting down all communications). The Dsniff package comes with a program called *macof* that is designed to switch MAC address flooding. It can be downloaded from www.monkey.org/~dugsong/dsniff.
- **ARP Redirects** When a computer needs the MAC address of another computer, it sends an Address Resolution Protocol (ARP) request. Each computer also maintains an ARP table that stores the MAC addresses of the computers it talks to. ARPs are broadcast on a switch; therefore, all computers on that switch see the request and the response. There are several methods that use ARP to trick a switch into sending traffic somewhere it shouldn't. First, an intruder can subvert a switch by sending out an ARP claiming to be someone else. An intruder can also send an ARP claiming to be the router, in which case computers will try to send their packets through the intruder's computer. Or, an intruder will send an ARP request to just one victim, claiming to be the router, at which point the victim starts forwarding packets to the intruder.
- **ICMP Redirect** Sometimes computers are on the same physical segment and switch, but different logical segments. This means they are in different IP subnets. When Computer A wants to talk to Computer B it sends its request through a router. The router knows that they are on the same physical segment, so it sends an ICMP Redirect to Computer A letting it know that it can send its packets directly to Computer B. An intruder (Computer X) can send a fake ICMP redirect to Computer A, telling it to send Computer B's packets to Computer X.
- **ICMP Router Advertisements** These advertisements tell computers which router to use. Intruders then send out advertisements claiming to be that router, at which point the computers begin forwarding all packets through the intruder.

- **MAC Address Spoofing** An intruder can pretend to use a different computer by spoofing its MAC address. Sending out packets with the source address of the victim tricks the switch. The switch enters the spoofed information into its table and begins sending packets to the intruder. But what about the victim who is still on the switch, sending updates that are causing the switch to change the table back? This can be solved by taking the victim offline with some type of DoS attack, and then redirecting the switch and continuing communications. An intruder could also broadcast the traffic that he or she receives to ensure that the victim computer still receives the packets. Some switches have a countermeasure that allows you to statically assign a MAC address to a port. This may be difficult to manage if you have a large network, but it will eliminate MAC spoofing. Other switch configurations allow a port to be locked to the first MAC it encounters, and presents a compelling balance between manageability and security in environments where physical port access is restricted.

  To spoof your MAC on Linux or Solaris use the **ifconfig** command as follows:

```
ifconfig eth0 down
ifconfig eth0 hw ether 00:02:b3:00:00:AA
ifconfig eth0 up
```

  Register the MAC on all hosts by broadcast ping: **ping -c 1 -b 192.168.1.255**.

  Now you can sniff all traffic to the computer that owns this MAC address.

- **Reconfigure Port Spanning On the Switch** As mentioned earlier, switch ports can be configured to see traffic destined for other ports. An intruder can perform this by connecting to the switch via Telnet or some other default back door. An intruder can also use SNMP if it is not secured.
- **Cable Taps** As mentioned earlier, cable taps can be used to physically tap into the cable. Tapping into the uplink cable on a switch shows you all of the traffic entering and exiting that switch.

There are many methods for defeating switches that are contingent on how a switch operates. Not all of the methods discussed work, especially with new, more technologically savvy switches. The Dsniff Frequently Asked Questions (FAQ)

helpful information for sniffing in a switched environment, and can be located at www.monkey.org/~dugsong/dsniff/faq.html.

Detecting Sniffers

As mentioned earlier, sniffers are a form of passive attack. They don't interact with any devices or transmit any information, thus making them very difficult to detect. Although tricky, detecting sniffers is possible. The easiest method is to check your network interfaces to see if they are in promiscuous mode. On UNIX-based systems, the command **ifconfig –a** lists the network adapters on the system. Look for the PROMISC flag in the output, such as in the following example:

```
[root@localhost root]# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:02:B3:06:5F:5A
          inet addr:192.168.1.2  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:204 errors:0 dropped:0 overruns:0 frame:0
          TX packets:92 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:46113 (45.0 Kb)  TX bytes:5836 (5.6 Kb)
          Interrupt:11 Base address:0x1800 Memory:e8120000-e8120038
```

If **ifconfig** does not detect a sniffer that you know is currently installed and in promiscuous mode, you can try using the **ip link** command, a TCP/IP interface configuration and routing utility. The following example shows the output from the IP command:

```
[root@localhost root]# ip link
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,PROMISC,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:02:b3:06:5f:5a brd ff:ff:ff:ff:ff:ff
```

Detecting promiscuous mode on Windows systems is difficult because there are no standard commands that list that type of information. However, there is a free tool called PromiscDetect (developed by Arne Vidstrom), which detects promiscuous mode network adapters for Windows NT, 2000, and XP. It can be downloaded from www.ntsecurity.nu/toolbox/promiscdetect. The following example shows the output of PromiscDetect: the D-link adapter is in normal operation mode, and the Intel adapter is running Wireshark:

```
C:\>promiscdetect
PromiscDetect 1.0 - (c) 2002, Arne Vidstrom (arne.vidstrom@ntsecurity.nu)
                  - http://ntsecurity.nu/toolbox/promiscdetect/
Adapter name:
 - D-Link DWL-650 11Mbps WLAN Card
Active filter for the adapter:
 - Directed (capture packets directed to this computer)
 - Multicast (capture multicast packets for groups the computer is a member
of)
 - Broadcast (capture broadcast packets)
Adapter name:
 - Intel(R) PRO/100 SP Mobile Combo Adapter
Active filter for the adapter:
 - Directed (capture packets directed to this computer)
 - Multicast (capture multicast packets for groups the computer is a member
of)
 - Broadcast (capture broadcast packets)
 - Promiscuous (capture all packets on the network)
WARNING: Since this adapter is in promiscuous mode there could be a sniffer
         running on this computer!
```

Some sniffers cover their tracks by hiding PROMISC flags. Also, if a sniffer is installed on a compromised system using a rootkit, the intruder probably replaces commands such as **ifconfig**. The following list describes several other methods that can be used to detect sniffers on the network:

- **Monitor DNS Reverse Lookups** Some sniffers perform DNS queries to resolve IP addresses to host names. Performing a network ping scan or pinging your entire network address space can trigger this activity.
- **Send TCP/IP Packets with Fake MAC Addresses to All IP Addresses On the Same Ethernet Segment** Normally, the NIC drops packets with the wrong MAC address. However, when in promiscuous mode, some systems answer with a reset (RST) packet. This might also work in a switched environment, because switches forward broadcast packets that they don't have MAC addresses for. Many new sniffers have built-in defenses for this technique, altering the way they handle MAC addresses.
- **Carefully Monitor Hub Ports** Ideally, you have a network diagram and your cables are labeled. Then, if something unusual appears (e.g., a new

device or a newly active hub port), you will recognize it. However, in reality, wiring closets and cabling can be a nightmare. If your hubs are being monitored with a protocol such as SNMP via a network management system, you may be able to use the information to detect any unusual connects and disconnects.

- **Remember How ARP is Used to Link IP Addresses to MAC Addresses** Normally, an ARP is sent out as a broadcast to everyone. However, you can also send out an ARP to a non-broadcast address, followed by a broadcast ping. No one should have your information in an ARP table except the sniffer that was listening to all of the traffic (including the non-broadcast traffic). Therefore the computer with the sniffer responds.
- **Use a Honeypot** A honeypot is a server that contains fake data and services to monitor the activity of intruders. In this case, an intruder can create fake administrator or user accounts on the honeypot, and then create connections across the network using cleartext protocols such as Telnet or FTP. If sniffers are monitoring for usernames and passwords, they will see the honeypot and the intruder will probably try to log into it. Honeypots run IDS to monitor activity, and special signatures can be added to trigger alerts when fake accounts are used.
- **Carefully Monitor Your Hosts** This includes disk space, central processing unit (CPU) utilization, and response times. Sniffers gradually consume disk space as they log traffic, and can occasionally put a noticeable load on the CPU. As the infected computer's resources become more utilized, it begins to respond slower than normal.

There are several tools that can be used to detect sniffers on a network. Many of them are outdated, no longer actively maintained, and sometimes hard to find. New sniffers have been rewritten to evade detection. The following is a list of some of those tools:

- **PromiScan Ver 0.27** This free program was developed by Security Friday, and is up-to-date and actively maintained. It runs on Windows 2000 and XP and requires the WinPcap driver. It scans the local network looking for remote promiscuous mode adapters using ARP packets, and can be downloaded from www.securityfriday.com/products/promiscan.html.
- **Sentinel** This free program performs remote promiscuous detection, and runs on various versions of Berkeley Software Distribution (BSD) and

Linux. It requires the libpcap and libnet libraries to operate, and can be downloaded from www.packetfactory.net/projects/sentinel.

- **Check Promiscuous Mode (CPM)** This is a free UNIX-based program developed by the Computer Emergency Response Team/Coordination Center (CERT/CC) in response to increased network sniffing. More information, including the program, can be obtained from www.cert.org/advisories/CA-1994-01.html.
- ***Ifstatus*** This is a free UNIX-based program that detects promiscuous mode interfaces on Solaris and Advanced IBM Unix (AIX) systems. It can be downloaded from ftp://ftp.cerias.purdue.edu/pub/tools/unix/sysutils/ifstatus.
- ***Promisc.c*** This is a free UNIX-based program that detects promiscuous mode interfaces on Linux and some SunOS systems. It can be downloaded from www.phreak.org/archives/exploits/unix/network-sniffers/promisc.c.

Sniffing Wireless

From the airport, to the coffee shop, to the library, to your next door neighbor, wireless networks are all around us; therefore, wireless security is a serious concern. There are historical weaknesses in security protocols, because intruders no longer need to be inside a building to attack an internal network. A wireless network is still a network, however, and with a few exceptions maps well to the Ethernet and OSI models.

Hardware Requirements

While most Ethernet cards are capable of packet sniffing in promiscuous mode, many wireless chipsets cannot use *monitor* mode, which is the wireless equivalent of promiscuous mode. Complicating the situation is that wireless card manufacturers do not generally list the chipset that they use in a readily available form. Also, chipsets can vary within model families. It is best to select the software you want to use, and then identify which chipsets and specific manufacturer's model numbers work best with the specific drivers necessary for the software to function.

Here are some general guidelines on chipset compatibility:

- **Atheros** This chipset is compatible with most software and widely available in a number of adapters.

- **Prism2** This chipset is one of the most capable used with the Host AP drivers. Not only is it supported by most software, it can also run in an AP mode.
- **Orinoco** One of the first chipsets that supported monitor mode. Supported by most software. Cannot receive 802.11g traffic.
- **Broadcom** There is no native support in Linux for this chipset. With included drivers, tools such as Kismet do not function with it. You may be able to use Windows drivers through a Network Driver Interface Specification (NDIS) compatibility wrapper such as the commercial DriverLoader, which can be downloaded from www.linuxant.com/driverloader.

Software

The proper combination of hardware, software, and drivers will enable you to effectively sniff wireless networking traffic. The following tools may be helpful:

- **Netstumbler** Netstumbler is more of a network scanner than a network sniffing tool, but is useful for listing networks detectable from your location. Netstumbler is an active network scanner that sends out probes that are detectable by others. It can be downloaded for free from www.netstumbler.org.
- **Kismet** Kismet is an open-source, free, wireless network scanner and vulnerability detector, which keeps track of wireless clients and their network associations. Unlike other scanners, it is a completely passive network scanner, and can be downloaded from www.kismetwireless.net.
- **Wireshark** Wireshark has a number of dissectors for wireless management traffic; however, it does not track by Service Set Identifier (SSID), nor does it show signal strength.
- **CommView for WiFi** CommView for WiFi is a commercial wireless network monitor and scanner that can export in tcpdump format, which Wireshark imports and reads easily. CommView for WiFi can be downloaded from www.tamos.com/products/commwifi/.

NOTE

Bootable CD-ROMs
There are several bootable Linux distributions that come prepackaged with the correct drivers and software necessary for wireless and wired network sniffing. All of these include Kismet and Ethereal or Wireshark. Below are some that are available and free:

- **Backtrack** Backtrack is the result of two highly respected bootable penetration toolsets combining their efforts toward one unified bootable CDROM. For additional information, go to www.remote-exploit.org.
- **Professional Hacker's Linux Assault Kit (Phlack)** Includes many security tools and wireless auditing and scanning software. For additional information, go to www.phlak.org.
- **Knoppix Security Tools Distribution (Knoppix-STD)** A general-purpose collection of security tools on a bootable Linux image. For additional information, go to www.s-t-d.org.

Protocol Dissection

Now that we've reviewed many of the critical portions of layers 1 through 4 of the OSI networking model, some attention should be paid to some of the protocols that you may run across while using Wireshark. The larger the network that you are sniffing, the more types of protocols (and protocol anomalies) you are likely to encounter.

DNS

The DNS translates hostnames into IP addresses, and vice versa. Most DNS traffic is transferred over UDP port 53 in a client/server fashion. DNS can be considered *forward* or *reverse*. Forward DNS translates a hostname into an IP address, and reverse DNS translates an IP address into a hostname. On the protocol level, forward and reverse lookups are nearly identical.

To get an IP address from a given hostname, a DNS system (also known as a *resolver*) requests an address (A) record from a DNS server. In the following example, we ask the authoritative name server for the IP address of *www.example.com*. In tcpdump format, we see the following traffic:

```
IP 192.168.0.1.33141 > 192.0.34.43.53:  42827+ A? www.example.com.
IP 192.0.34.43.53 > 192.168.0.1.33141:  42827*- 1/2/2 A 192.0.34.166
```

The resolver (192.168.0.1) asked the authoritative name server (192.0.34.43) on UDP port 53 for the "A record" for www.example.com. Via UDP, the name server returned one A record for that name with IP address 192.0.34.166.

Wireshark can also be used to view more information about this DNS transaction. Wireshark would return the following information about the query and response:

```
Domain Name System (query)
    Transaction ID: 42827
    Flags: 0x0100 (Standard query)
        0... .... .... .... = Response: Message is a query
        .000 0... .... .... = Opcode: Standard query (0)
        .... ..0. .... .... = Truncated: Message is not truncated
        .... ...1 .... .... = Recursion desired: Do query recursively
        .... .... .0.. .... = Z: reserved (0)
        .... .... ...0 .... = Non-authenticated data OK
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
    Queries
        www.example.com: type A, class IN
            Name: www.example.com
            Type: A (Host address)
            Class: IN (0x0001)
Domain Name System (response)
    Transaction ID: 42827
    Flags: 0x8180 (Standard query response, No error)
        1... .... .... .... = Response: Message is a response
        .000 0... .... .... = Opcode: Standard query (0)
        .... .0.. .... .... = Authoritative
        .... ..0. .... .... = Truncated: Message is not truncated
        .... ...1 .... .... = Recursion desired: Do query recursively
        .... .... 1... .... = Recursion available
        .... .... .0.. .... = Z: reserved (0)
        .... .... ..0. .... = Answer authenticated
        .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 1
    Authority RRs: 13
```

```
    Additional RRs: 2
    Queries
        www.example.com: type A, class IN
            Name: www.example.com
            Type: A (Host address)
            Class: IN (0x0001)
    Answers
        www.example.com: type A, class IN, addr 192.0.34.166
    Authoritative nameservers
        com: type NS, class IN, ns C.GTLD-SERVERS.NET
       ...
        com: type NS, class IN, ns B.GTLD-SERVERS.NET
    Additional records
        A.GTLD-SERVERS.NET: type A, class IN, addr 192.5.6.30
```

NOTE

DNS uses TCP instead of UDP for transmitting data when the data size exceeds 512 bytes. DNS also uses TCP for transferring entire DNS zones between zones. In either case, port 53 is used.

NTP

The NTP is another helpful protocol that keeps things running smoothly in the background. In this case, NTP makes sure that all of your computer and device clocks are synchronized. NTP can use peering or client/server architecture; the network traffic will be similar either way. UDP port 123 is used for NTP.

In the following example, NTP client (192.168.0.1) asks a NTP server (192.168.0.2) for the current timestamp:

```
IP 192.168.0.1.ntp > 192.168.0.2.ntp: NTPv4, Client, length 48
IP 192.168.0.2.ntp > 192.168.0.1.ntp: NTPv4, Server, length 48

Network Time Protocol
    Flags: 0xe3
        11.. .... = Leap Indicator: alarm condition (clock not synchronized)
        ..10 0... = Version number: NTP Version 4 (4)
```

```
        .... .011 = Mode: client (3)
    Peer Clock Stratum: unspecified or unavailable (0)
    Peer Polling Interval: 6 (64 sec)
    Peer Clock Precision: 0.000008 sec
    Root Delay:    0.0000 sec
    Clock Dispersion:    0.0039 sec
    Reference Clock ID: Unindentified reference source 'INIT'
    Reference Clock Update Time: NULL
    Originate Time Stamp: Mar 29, 2006 06:09:01.6976 UTC
    Receive Time Stamp: Mar 29, 2006 06:09:01.7563 UTC
    Transmit Time Stamp: Mar 29, 2006 06:10:07.7525 UTC

Network Time Protocol
    Flags: 0x24
        00.. .... = Leap Indicator: no warning (0)
        ..10 0... = Version number: NTP Version 4 (4)
        .... .100 = Mode: server (4)
    Peer Clock Stratum: secondary reference (5)
    Peer Polling Interval: 6 (64 sec)
    Peer Clock Precision: 0.000008 sec
    Root Delay:    0.0000 sec
    Clock Dispersion:    0.0122 sec
    Reference Clock ID: 127.127.1.0
    Reference Clock Update Time: Mar 29, 2006 06:09:48.4681 UTC
    Originate Time Stamp: Mar 29, 2006 06:10:07.7525 UTC
    Receive Time Stamp: Mar 29, 2006 06:10:07.6674 UTC
    Transmit Time Stamp: Mar 29, 2006 06:10:07.6675 UTC
```

HTTP

HTTP is the most widely used protocol that supports the Web. HTTP uses TCP to transmit data exclusively, and in a default configuration uses port 80. Each object (e.g., Web page, image, audio) fetched from a Web server is transmitted via an individual HTTP session.

To begin an HTTP session, a client establishes a regular TCP connection on port 80 and sends a packet with the SYN flag set. A packet is returned from the Web server, with an ACK flag added to the SYN flag. Finally, the client sends a packet with the ACK flag set, and then sends another packet requesting a specific HTTP object.

The following is an example of a client's request to a HTTP server:

```
GET /index.html HTTP/1.1
```

The client requests the *index.html* page using HTTP v1.1:

```
Host: www.example.com
```

The hostname that was typed in the browser allows a server to host multiple Web services on one IP address:

```
User-Agent: ELinks/0.11.0 (textmode; Linux; 80x25-2)
```

The User-Agent describes the Web browser version to the server. Some browsers allow users to change hostnames; thereby deeming it unreliable.

```
Accept-Encoding: gzip
Accept-Language: en
Connection: Keep-Alive
```

These lines tell the Web server that the client supports compression of the object requested, accepts pages in English, and the Web server doesn't have to disconnect upon completion of the object request.

The Web server sends back the following information to the client:

```
HTTP/1.1 200 OK
```

The Web server responds in HTTP/1.1 with status code "200 OK," which indicates to the browser that the object was successfully fetched. Other codes are "403 Forbidden," (the server does not have permission to send the object to the client, and "404," (the server cannot find the object that the client requested).

```
Date: Thu, 30 Mar 2006 05:23:29 GMTLast-Modified: Wed, 29 Mar 2006 16:22:05 GMT
Server: Apache/2.2.0
```

These lines allow the client to cache content efficiently. It tells the client what time the server thinks it is, and when the content was last modified. The server also identifies its product (Apache) and version (2.2.0), although this can be changed by the server administrator.

```
Accept-Ranges: bytes
Content-Length: 40Connection: close
Content-Type: text/html; charset=UTF-8<HTML><BODY>Hello, world!</BODY></HTML>
```

If needed to restart the transfer, the server tells the client in what form it can request portions of a file (in this case it accepts bytes). The server then tells the client to close the connection after the data is finished. The actual HTTP data follows, beginning with the line "Content-Type."

SMTP

SMTP is the mechanism by which most e-mail is sent over the Internet. SMTP uses TCP to transmit data exclusively, and in most situations the server uses port 25. The entire e-mail (headers and contents) is sent in one SMTP session. It is easy to emulate a SMTP session using the Telnet program to port 25 of an e-mail server.

Think of an SMTP connection the same as sending a memo through the regular mail. On the outside of the envelope are a return address and a destination. The return address and destination might also be repeated inside the envelope, but the mail carrier doesn't care about what is inside the envelope. In an SMTP connection, the message envelope is transmitted first, followed by the letter contents inside.

The following is an example of an SMTP conversation. The client sends in normal text, and the server responds in italics:

```
220 example.org ESMTP Mail Service
HELO client.example.com
250 Ok
```

Upon connection, the server indicates its presence with a *banner* that includes the version of the e-mail server program, but can also be configured by the user to be an arbitrary banner, as long as it begins with the hostname of the server. The client says "HELO" to the server, and tells it what name it wants to go by. The **HELO** command is also used for clients that support advanced SMTP features such as encryption. The server acknowledges the client with an "Ok" response.

```
MAIL FROM:<person@example.com>
250 Ok
```

The client sends a **MAIL FROM** command, indicating the return address, which may or may not match the letter's contents. Your Mail User Agent or e–mail reader normally only show the address contained in the letter, disregarding the envelope.

```
RCPT TO:<anotherperson@example.org
250 Ok
```

The client sends the destination of the envelope and the server acknowledges it.

At this point, you may see a "Relaying Denied" message, indicating that the server will not accept the e-mail. At the beginning of e-mail, systems were free to send e-mail directly to each other, or by relaying it through any other system on the Internet. However, this arrangement broke down in the 1990s when spam became a major issue on the Internet. Most systems now accept e-mail for themselves only, and relaying is generally only relevant to ISPs.

```
DATA
354 End data with <CR><LF>.<CR><LF>
From: "person" <person@example.com>
To: example@example.com
Example E-Mail subject.
Example e-mail contents.
.
250 Ok: queued as C8243B4039QUIT221
Quit 221 Bye
```

The client sends the **DATA** command, telling the server that the contents of the letter will be transmitted. The e-mail's headers are normally repeated at this point, and it is from here that e-mail is communicated. To end the e-mail, the server instructs the client to send a linefeed, followed by a dot and another linefeed. The client politely issues the *QUIT* command, and the server bids the client farewell.

Protecting Against Sniffers

So far, you have learned what sniffing is and how it works. You have also learned some of the tricks that can be used by intruders for sniffing, and some not-so-fool-proof methods of detecting sniffers. None of this sheds a positive light on your plight to protect your network and data. However, there are some methods on your network that offer protection against sniffing.

We talked earlier about using switches instead of hubs, and we learned the methods used to defeat switches. Using switches is a network best practice that allows increased performance and security. While switches present a barrier to casual sniffing, the best method of protecting your data is to use encryption, which is the best form of protection against traffic interception on public networks and internal networks. Intruders can still sniff the traffic, but the data appears unreadable. Only the intended recipient should be able to decrypt and read the data; however, some methods of encryption leave the packet headers in cleartext, thereby allowing intruders to see the source and destination addresses and map the network. However, the data contained within the packet is protected. Other forms of encryption also mask the header portion of the packet.

A VPN uses encryption and authentication to provide secure communication over an otherwise insecure network. VPNs protect the transmission of data over the Internet and over your internal network. However, if an intruder compromises either of the end nodes of a VPN, the protection is rendered useless. Different types of VPN families are not interchangeable, but they can be combined and used in multiples. The

following list describes some of the VPN methods used today that protect your data against sniffing:

- **SSH** SSH is an application-level VPN that runs over TCP to secure client-to-server transactions. This is often used for system logins and to administer servers remotely, and is typically used to replace Telnet, FTP, and the BSD **r** commands. However, any arbitrary TCP protocol can be tunneled through an SSH connection and used for numerous other applications. SSH provides authentication using Rivest, Shamir, & Adleman (RSA) or Digital Signature Algorithm (DSA) asymmetric key pairs, and many encryption options for protecting data and passwords sent over the network. The headers in an SSH session are not encrypted, so an intruder can still view the source and destination addresses.
- **Secure Sockets Layer (SSL)/Transport Layer Security (TLS)** SSL was originally developed by Netscape Communications to provide security and privacy to Internet sessions. It has been replaced by TLS, as stated in RFC 2246. TLS provides security at the transport layer and overcomes some security issues of SSL. It is used to encapsulate the network traffic of higher-level applications such as HTTP, Lightweight Directory Access Protocol (LDAP), FTP, SMTP, POP3, and IMAP. It provides authentication and integrity via digital certificates and digital signatures and the source and destination IP headers in a SSL session are not encrypted.
- **IP Security (IPSec)** IPSec is a network-level protocol that incorporates security into the IPv4 and IPv6 protocols directly at the packet level, by extending the IP packet header. This allows the ability to encrypt any higher-layer protocol. It has been incorporated into routing devices, firewalls, and clients for securing trusted networks to one another. IPSec provides several means for authentication and encryption, supporting a lot of public key authentication ciphers and symmetric key encryption ciphers. It can operate in *tunnel* mode to provide a new IP header that masks the original source and destination addresses in addition to the data being transmitted. Since IPSec uses protocols other than TCP and UDP, getting the IPSec traffic through a firewall or NAT device can be challenging.
- **OpenVPN** OpenVPN is a tunneling SSL VPN protocol, which can encrypt both the contents of a packet and its IP headers. OpenVPN uses a single TCP or UDP port; therefore, it can be easier to use in environments with challenging NAT and firewall architectures. Additionally, it can act as a virtual network bridge (a layer 2 VPN).

One-time passwords (OTP) are another method to protect against sniffing. S/key, One-time Passwords In Everything (OPIE), and other one-time password techniques protect against the collection and reuse of passwords. They operate using a challenge-response method, and a different password is transmitted each time authentication is needed. The passwords that a sniffer collects are eventually useless since they are only used once. Smart cards are a popular method of implementing one-time passwords. However, OTP technologies cannot help protect your password after you enter it.

E-mail protection is a hot topic for companies and individuals. Two methods of protecting e-mail (i.e., encrypting it in-transit and in storage) are Pretty Good Privacy (PGP) and Secure Multipurpose Internet Mail Extensions (S/MIME). Each of these methods also provides authentication and integrity using digital certificates and digital signatures.

Network Analysis and Policy

Before cracking open your newly installed network analyzer at work, read your company policy! A properly written and comprehensive "Appropriate Use" network policy will prohibit you from running network analyzers. Usually the only exception to this is if network analysis is in your job description. Also, just because you provide security consulting services for company clients does not mean that you can use your sniffer on the company network. However, if you are an administrator and allowed to legitimately run a sniffer, you can use it to enforce your company's security policy. If the policy on using sniffers is not clear in your organization, take the time to get permission in writing from the appropriate departments before using them or any other security-related tools. On the other hand, if your security policy prohibits using file-sharing applications (e.g., KaZaA, Morpheus, BitTorrent or messaging services such as Internet Relay Chat [IRC] or Instant Messenger [IM]), you could use a sniffer to detect this type of activity.

Also, if you provide security services for clients, be sure that using a sniffer is included in your Rules of Engagement. Be very specific about how, where, and when it will be used. Also, provide clauses (e.g., Non-Disclosure Agreements) that will exempt you from the liability of learning confidential information.

Ensure that your sniffing activities do not violate any laws against wiretapping. In many countries, wiretapping laws were enacted at a time when modems were the most complicated network access device, so the clarification of laws and related regulatory requirements can be complex and differ based on the situation and the parties involved.

CAUTION

Many ISPs prohibit using sniffers in their "Appropriate Use" policy. If they discover that you are using one while attached to their network, they may disconnect your service. The best place to experiment with a sniffer is on your home network that is not connected to the Internet. All you need is two computers with a crossover cable between them, or a virtual machine application. You can use one as a client, and install server services on the other (e.g., Telnet, FTP, Web, and e-mail).

NOTE

You can download packet traces from numerous Web sites and read them with your network analyzer to get used to analyzing and interpreting packets.

The HoneyNet Project at www.project.honeynet.org has monthly challenges and other data for analysis.

The Wireshark "wiki" also has many well-described capture files that are located at www.wiki.wireshark.org/SampleCaptures.

Summary

Network analysis is the key to maintaining an optimized network and detecting security issues. Proactive management can help find issues before they turn into serious problems and cause network downtime or compromise confidential data. In addition to identifying attacks and suspicious activity, your network analyzer can be used to identify security vulnerabilities and weaknesses and enforce your company's security policy. Sniffer logs can be correlated with IDSes, firewalls, and router logs to provide evidence for forensics and incident handling. A network analyzer allows you to capture data from the network (packet-by-packet), decode the information, and view it in an easy-to-understand format. Network analyzers are easy to find, often free, and easy to use; they are a key part of any administrator's toolbox.

This chapter covered the basics of networking, Ethernet, the OSI model, and the hardware that is used in a network architecture; however, it only scratched the surface. A good networking and protocol reference should be on every administrator's bookshelf. It will come in handy when you discover some unknown or unusual traffic on your network.

As an administrator, you should know how to detect the use of sniffers by intruders. You should keep up-to-date on the methods that intruders use to get around security measures that are meant to protect against sniffing. As always, you also need to make sure that your computer systems are up-to-date with patches and security fixes to protect against rootkits and other backdoors.

This chapter also covered a variety of methods used to protect data from eavesdropping by sniffers. It is important to remain up-to-date on the latest security technologies, encryption algorithms, and authentication processes. Intruders are constantly finding ways to defeat current security practices, thus more powerful methods are developed (e.g., cracking the Data Encryption Standard [DES] encryption scheme and its subsequent replacement with Triple Data Encryption Standard (3DES), followed by the Advanced Encryption Standard (AES).

Finally, remember the rule of network analysis—only do it if you have permission and the law is on your side. A curious, up-and-coming administrator could easily be mistaken for an intruder. Make sure you have permission, or use your own private network to experiment.

Solutions Fast Track

What is Network Analysis and Sniffing?

- ☑ Network analysis is capturing and decoding network data.
- ☑ Network analyzers can be hardware or software, and are available both free and commercially.
- ☑ Network analyzer interfaces usually have three panes: Summary, Detail, and Data.
- ☑ The five parts of a network analyzer are: hardware, capture driver, buffer, real-time analysis, and decode.

Who Uses Network Analysis?

- ☑ Administrators use network analysis for troubleshooting network problems, analyzing the performance of a network, and intrusion detection.
- ☑ When intruders use sniffers, it is considered a passive attack.
- ☑ Intruders use sniffers to capture user names and passwords, collect confidential data, and map the network.
- ☑ Sniffers are a common component of a rootkit.
- ☑ Intruders use sniffers to control backdoor programs.

How Does it Work?

- ☑ Ethernet is a shared medium that uses MAC or hardware addresses.
- ☑ The OSI model has seven layers and represents a standard for network communication.
- ☑ Hubs send out information to all hosts on the segment, creating a shared collision domain.
- ☑ Switches have one collision domain per port and keep an address table of the MAC addresses that are associated with each port.
- ☑ Port mirroring is a feature that allows you to sniff on switches.

- ☑ Switches make sniffing more difficult; however, the security measures in switch architectures can be overcome by a number of methods, thus allowing the sniffing of traffic designated for other computers.
- ☑ Sniffing wired traffic can be done with many kinds of NICs; wireless sniffing requires greater attention to hardware details such as chipset and drivers.

Detecting Sniffers

- ☑ Sometimes sniffers can be detected on local systems by looking for the PROMISC flag.
- ☑ There are several tools available that attempt to detect promiscuous mode using various methods.
- ☑ Carefully monitoring hosts, hub and switch ports, and DNS reverse lookups assists in detecting sniffers.
- ☑ Honeypots are a good method to detect intruders on your network who are attempting to use compromised passwords.
- ☑ New sniffers are smart enough to hide from traditional detection techniques.

Protocol Dissection

- ☑ DNS packets can use either TCP or UDP, depending on the purpose of the query and the amount of data transmitted.
- ☑ NTP data transmissions generally use UDP port 123 for both the client and server side ports.
- ☑ Multiple virtual HTTP servers can listen on one port. The Host: header indicates to the server which virtual server the client intended to connect with.
- ☑ A SMTP connection can be emulated with a simple network program such as Telnet. If your SMTP server is left open on the Internet, it will eventually be used to send spam.

Protecting Against Sniffers

- ☑ Switches offer little protection against sniffers.
- ☑ Encryption is the best method of protecting your data from sniffers.
- ☑ SSH, SSL/TLS, and IPSec are all forms of VPNs that operate at various layers of the OSI model.
- ☑ IPSec tunnel mode can protect the source and destination addresses in the IP header by appending a new header.

Network Analysis and Policy

- ☑ Make sure you have permission to use a sniffer on a network that is not your own.
- ☑ Read the appropriate use policies of your ISP before using a sniffer.
- ☑ If you are hired to assess a computer network and plan to use a sniffer, make sure you have a non-disclosure agreement in place, because you may have access to confidential data.
- ☑ One-time passwords render compromised passwords useless.
- ☑ E-mail should be protected while in transit, and stored with some type of data encryption method.

Frequently Asked Questions

The following Frequently Asked Questions, answered by the authors of this book, are designed to both measure your understanding of the concepts presented in this chapter and to assist you with real-life implementation of these concepts. To have your questions about this chapter answered by the author, browse to **www.syngress.com/solutions** and click on the **"Ask the Author"** form.

Q: What can I do to protect my network from sniffers?

A: Proper network security comes by design, not just through action. Some argue that there is nothing you can do to make your network completely secure. A combination of network access controls like 802.1x, ubiquitous and opportunistic encryption, and strong policies and procedures will go a long way to

protecting your network from sniffers and other security issues. Using several layers of security is known as *defense-in-depth*, and is a standard best practice for secure network architectures.

Q: What is this opportunistic encryption that you speak of, and where can I buy it?

A: Opportunistic Encryption is the practice where communication between two parties becomes encrypted, even when those parties cannot be assured of each other's identity. More information on opportunistic encryption is available at www.en.wikipedia.org/wiki/Opportunistic_encryption free of charge. Ubiquitous opportunistic encryption is the theory that any network communication that *can* be encrypted, *should* be.

Q: How can I ensure that I am sniffing network traffic legally?

A: The best way to ensure that your sniffing activities are legal is to solicit expert legal counsel. In general, you should be safe if all parties to the communication that you are sniffing acknowledge that they have no expectation of privacy on your network. These acknowledgements can be in employment contracts, and should also be set as banners so that the expectation of no privacy is reinforced. It is advised that you get authorization (in writing) from your employer to use sniffing software.

Q: Is a sniffer running a security breach on my network?

A: Possibly. Check the source of the sniffing activity and verify that the interception has been authorized. Hackers and other network miscreants use sniffers to assist themselves in their work. It is best to design networks and other applications that are resilient to network sniffing and other security issues.

Q: Can I use a sniffer as an IDS?

A: While a sniffer can act similarly to an IDS, it is not designed as such. IDSes have threshold, alerting, and reporting systems that are beyond the design specifications for most sniffers.

Q: How do I use a sniffer to see traffic inside a VPN?

A: VPN traffic is normally encrypted and most sniffing software does not have the ability to read encrypted packets, even if you have the decryption key. The best place to see VPN traffic is outside of the VPN tunnel itself.