3

The Windows NT Network Environment

"The network is the computer."
–Scott McNealy

Network security is a major consideration in securing any computing system; Windows NT is by no means an exception to this principle. Before delving too deeply into the topic of security in the Windows NT network environment, however, let's first set the stage by providing an introduction to networking itself paying particular attention to principles applying to the environments in which Windows NT is typically deployed.

You need to understand the fundamentals of networking before you can comprehend many Windows NT security-related vulnerabilities. Unfortunately, Windows NT networking is anything but simple. Many of its many protocols and services are legacies from an earlier era of local area network (LAN) technology; others are part of the NT environment because of the need for interoperability between platforms; and still others are used in providing user services such as *World Wide Web* (WWW) and mail services. Securing these protocols and services is truly a challenging task.

This chapter covers the essentials of networking, including types of networks and network components, network topologies, the *Open Systems Interconnect* (OSI) model and its relationship to Windows NT network protocols and services, fundamental network services and protocols commonly found in Windows NT network environments, and other basic information about how networks operate. The chapter then concludes with a lengthy discussion of how Network Basic Input Output System (NetBIOS)- and Server Message Block (SMB)-based networking mechanisms work in the Windows NT environment. If you are very proficient with networking in general and Windows NT networking in particular, you might want to skip ahead to the final section.

Types of Network Implementations

Although there are many different definitions of networks, for purposes of this book network is defined as a collection of hosts, applications, protocols, and peripheral devices (such as printers)—all of which can communicate with each other through some combination of hardware, software, and protocols. One of the most notable changes in computing over the past decade has in fact been the proliferation of networking. The percentage of network hosts has grown steadily through the 1990s to the point that by the late 1990s, the percentage of non-network-capable ("standalone") hosts has diminished to a very small proportion.

Networks can be implemented in a number of different ways. One possible network implementation is a LAN. A LAN consists of hosts and peripheral devices that are relatively near to each other in terms of physical distance. In a *wide area network* (WAN), the hosts and peripheral devices are more physically disparate from one another.

Another consideration is the relationship of the computers within a given network to each other. In a peer-to-peer network, every network application has basically the same relationship (or at least the same *potential* relationship) to every other as all others do; no application is specialized in terms of its role in the network. Client/server networking, however, entails specialized functions—server and client. Servers provide network services (for example, the domainwide services that Windows NT domain controllers provide, the *Domain Name Service* (DNS), print services, *File Transfer Protocol* (FTP) services, and many others) and access to data stored in a database, for example. Clients provide users and applications with access to the servers through software, such as user interface routines, that translate user input and then send it to applications which reside on the server. The typical outcome is increased efficiency in resource utilization; At a minimum, each client needs enough processing power to remotely communicate with applications residing on servers. In a peer-to-peer network, in contrast, all computers are in many respects both clients and servers. They are clients because they provide an interface to applications and are servers to the degree that they run applications and provide data and services to clients.

Although peer-to-peer networks are advantageous when the need for a LAN with relatively few (perhaps fewer than a dozen) machines exists, client/server-based implementations tend to pose less security risk. In peer-to-peer networking, the security-related configuration of individual systems within the network are not generally built in. Client/server networks, on the other hand, provide support for servers used to centrally set configurations and parameters that affect the type and quantity of access to potentially

every computer within the network. Exceptions to this generalization exist (for example, when client/server networks are poorly configured); but all things considered, client/server networking is more conducive to security than are peer-to-peer implementations.

Some Basic Terminology

The following basic terminology is essential in understanding networking and the many mechanisms involved. Terms covered here include the following:

- Network interface card
- MAC address
- Internet protocol
- Packet
- Networking protocol
- Ethernet
- Source host
- Destination host
- Broadcast
- Multicast
- Sniffer
- Protocol Analyzer
- Socket
- Firewall

Network Interface Card

The *network interface adapter* or *card* (NIC) connects the computer on which it is installed to the cable medium, converting the output that the computer sends to signals (normally electronic in nature) that are sent over this medium. The receiving NIC mirrors this process by receiving a set of electronic signals and then translating them into packets for the computer to receive.

MAC Address

Each machine connected to the network is identified by an address built in to its NIC. This address, called the *Medium Access Control* (MAC) address, is a physical address used to uniquely identify every computer on the network.

IP Address

Internet Protocol (IP) addresses are logical (not physical) addresses that are (with one exception—IPv6[1]) always 32 bits in length. These bits are represented in terms of four octets, a format with which nearly everyone who uses the Internet is familiar. Each host address is typically represented as a four-octet IP address; the hypothetical host `sunshine.global.net` might, for example, have IP address `120.10.20.11`. This address in turn corresponds to the particular set of bits shown in Figure 3.1.

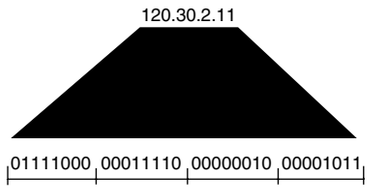

Figure 3.1 *Composition of an IP address in terms of octets (above) and bits (below).*

An Alternative Addressing Convention

Windows NT networking is largely based on the IP protocol. In contrast, Novell Internetwork Packet Exchange (IPX) addressing is a variant of the Xerox Network Service (XNS) addressing scheme originally developed for Ethernets. In the latter scheme, the host address is always 48 bits in length; a node address is always 32 bits. These bits are represented as four-digit hexadecimal numbers. An example of an IPX node address is `41.0000.0d00.12bb`. *The Novell addressing scheme is not used in the Internet; and given that Novell NetWare 5.0 runs TCP/IP natively, this addressing scheme is rapidly becoming a thing of the past. It is, nevertheless, still used in reaching NetWare 4.x (and earlier versions) hosts over the network and is therefore relevant to Windows NT networking because of interoperability considerations.*◆

Packet

Data sent over the network consists of blocks of characters (properly known as datagrams, but most people call them packets). Packets consist of a header, a small set of data about the type of packet, sending and destination host, and so forth, followed by the contents (data).

[1] *IPv6, the newest rendition of the IP protocol, is based on 128-bit addressing. There is no IPv6 stack in Windows 2000.*

Networking Protocol

When networked computers communicate with each other, they must share common methods and meanings to successfully do so. When one computer transmits data, for example, the receiving computer must not only prepare to receive it, but also must determine the context and meaning of this data. Protocols provide the solution by establishing methods of communication between computers: how communications are started, parameters that must be passed between machines, the order in which they are to be sent, and many other considerations.

Ethernet

Ethernet is a very widely used protocol that network adaptors connected to cabling media use. It determines the numeric values given to series of electrical signals to form valid Ethernet packets. Its method of networking is based on *carrier sense multiple access collision detect* (CSMA/CD) control in which hosts on the network wait for the right time to send network transmissions to each other. If no traffic is going over the part of the network to which these hosts interface, they start to send their transmissions. If more than one host sends a transmission at once, however, the packets collide with each other, necessitating another transmission attempt. Collisions are not so much of a problem in smaller Ethernets, but the opposite is true in larger ones. Both switching, a mechanism that makes a direct path between ports, and routing are frequently used to deal with these types of problems.

Source Host

The source host initiates a network transmission (for example, packets).

Destination Host

The destination host is the host designated to receive a network transmission.

Broadcast

Networked computers often need to obtain data about the status and availability of other computers or to provide information to other computers. In other cases, computers need something (an IP address, an executable, for example) to locate a particular type of server or something else from a server. This is where broadcasts fit in. Broadcast packets are designed to be sent to every machine within a local network. With a destination address of `255.255.255.255`, these packets can reach every host on the local network, but by default cannot be sent outside of the local network unless they are encapsulated within another routable protocol.

Multicast

Multicast is an optimized multipoint transmission protocol. It is used in networking contexts, such as multimedia transmission and high volume data transfers.

Sniffer

Sniffer is a trademarked name owned by Network Associates. The Sniffer product that Network Associates makes captures packets transmitted over network cabling, dumps the data contents to storage media, and provides a user interface to view the data. The term *sniffer*, however, is a widely used term within the computing community to refer to any physical device or program that captures packet data. Because of trademark considerations, *packet capture device* or *packet capture program* is the appropriate term.

Protocol Analyzer

Having a device or program that captures data is useful; but having a device or program that dumps and sorts the data in different formats and views is even more useful, given the volume of data normally collected. A protocol analyzer does exactly this—it dumps captured network data in different formats and views to make analysis easier.

Socket

A socket is a convention that enables a user or program to reach a particular program on a destination host.

Firewall

A firewall is a host used as a security barrier between networks. Firewalls selectively screen traffic and sometimes also manage connections between one network and another.

Major Network Components

This section provides a brief discussion of the basic components of most networks—repeaters, bridges, hubs, and routers.

Repeaters

In the most basic sense, networks work by sending electrical signals across cables. These signals, however, attenuate as they traverse each cable; at some point along a cable, the signal may fail to carry any further. Devices called *repeaters* are used in digital networks to regenerate signals as they go down cables so that they can reach destination hosts. Repeaters are often

used to increase the physical size of LANs, allowing additional systems and peripherals to connect to a preexisting LAN. Note, however, that Ethernet does not need repeaters, because of the limited distances over which traffic to segments travels.

Contrasted with other types of network components, repeaters are in many respects more passive in their functionality. In the most elementary sense, they simply take signals, magnify them, and then send them along a network cable without the capacity to selectively filter, in any way, what is sent. A potential problem in connection with repeaters, therefore, is that a repeater can potentially overwhelm a network with volumes of traffic. Hosts in any part of a network in which repeaters are present can produce an enormous amount of traffic volume even though some, much, or all of this traffic may be superfluous to the systems along the way.

You might think, therefore, that repeaters would be frequent targets of network attacks in the Windows NT or any other networking environment. Gaining unauthorized access to a repeater or sending network traffic to a repeater in a manner that causes the repeater to fail or other possible attack scenarios could conceivably lead to widespread denial-of-service. Attacks against other components of networks (for example, routers and firewalls) are, however, generally not only easier to remotely perpetrate, but they are also more likely to subvert higher-level network functionality, such as routing.

Bridges

A *bridge* is very similar in functionality to a repeater, but operates at Layer 2 of the OSI model. Therefore, a bridge actually *filters* traffic transmitted over the network based on the Layer 2, or MAC, address. Bridges dynamically update their routing tables with source addresses as they receive packet traffic. They determine the MAC address for each machine on the basis of the contents of packets (the MAC addresses of both the source and destination hosts) sent over the network. The destination MAC address of each packet is then used to selectively filter traffic packets in the following manner:

- Packets with unidentified destinations are sent on to every network segment to which the bridge is connected.
- Packets with identified destinations on other network segments to which the bridge is connected are sent on to the segment on which the destination machine resides.
- Packets in which the source and destination address are both within the same network segment are not sent to any other network segment.

Bridges are both good and bad in terms of their contribution to the problem of managing networks. An advantage of bridges (at least compared to repeaters) is that the former are not as passive; they can at least send traffic to other network segments or, if appropriate, keep traffic within a particular local segment. They can also selectively filter traffic on the basis of each packet's destination MAC address. A disadvantage is that bridges cannot filter broadcasts (transmissions from hosts that are intended to reach other computers independently of the computers' addresses).

Hubs

Some types of networks have network cables that go from the connected computers to one single point. *Hubs* (sometimes called concentrators) connect the cables at this point. Hubs vary in sophistication. Some of them just rebroadcast any signals they receive from a particular cable on to every other cable. Others are higher end, in that they function as network switches. They determine packets' destinations and then resend the signal exclusively to the appropriate cable over which the packets must be transmitted to reach this destination.

Routers

Routers are the final network component to be considered in this chapter. Routers, like bridges, route and screen traffic; but whereas bridges operate at Layer 2 and therefore use the MAC address to determine whether to forward traffic, routers operate at Layer 3, and therefore forward traffic based on the network layer address. In addition, routers have knowledge of networks outside the networks to which they are directly connected, and consequently can interconnect one LAN to many others. By contrast, bridges have no intelligence other than to determine which network a certain MAC address exists on. They can be used only to interconnect the networks to which they are directly connected. Whereas bridges can interconnect only networks that use the same data link layer protocol, routers can connect networks with different data link layer protocols, as long as they use the same network layer protocol. The important result is that routers can handle packets on a higher, more abstract level rather than information supplied by any NIC. Consider, for example, the effect of dynamic IP addressing. One machine is assigned one particular IP address for a while, and then a different address sometime afterward. Routers have no particular difficulty dealing with dynamic addressing provided the assigned IP addresses are legitimate and routing tables have been set up and maintained suitably.

The basis for routing, therefore, is routing tables. When a router boots, it has few addresses in its table. Routing protocols, such as the *Address*

Resolution Protocol (ARP), add entries to the tables as packets pass through. Network administrators can also add or delete entries in routine tables as desired. Routers may also support Access Control Lists (ACLs), rule lists specifying whether to allow or deny inbound (and also often outbound) traffic according to certain variables. These variables include source and/or destination IP address, type of protocol (for example, Simple Mail Transfer Protocol [SMTP] and File Transfer Protocol (FTP), and type of packet (SYN, ACK, FIN, and so forth). A router's ACL may block all incoming packets destined for a particular IP address, for example. A bridge, on the other hand, sends and filters packets on the basis of the network segment to which a machine with a certain MAC address is connected.

Domain Name Server

When you send a mail message to an Internet address or hit a Web site, a mechanism needs to translate this address (for example, `http:// security.globalintegrity.com`*) into a numeric IP address. The service that performs this task (called address resolution) is domain name service (DNS). Using its database of addresses and IP addresses, the Address Resolution Protocol (ARP) is used to resolve layer three (IP) addresses to layer 2 (MAC) addresses. Similarly, the Reverse Address Resolution Protocol (RARP) is used to get an IP address from a MAC address. Windows NT supports ARP, but not RARP. DNS does not use ARP at all. DNS resolves a Fully Qualified Domain Name (FQDN) to an IP address, using a forward lookup query. To get an FQDN from an IP address, a reverse lookup is performed. Because so many Internet applications and services depend on DNS, internetworking as we know it would be virtually impossible without this service.*

The Internet consists of a large number of computing domains; each Internet-connected host is part of one domain. The host `security.globalintegrity.com`*, for example, is part of a domain called* `globalintegrity.com`*. A given DNS server can resolve addresses only within its own domain, but it also knows how to forward resolution requests to other DNS servers if it cannot resolve the name. If unable to resolve* `globalintegrity.com`*, the DNS servers within a tree forward the request all the way to the top (the root domain) to find the one that knows about the type of organization (for example,* `.com, .org, .gov, .edu`*). From there the request is forwarded to a DNS server that knows* `.com` *addresses, and then to still another that knows* `globalintegrity`*.*

DNS serves as a convenient target for perpetrators for numerous reasons. At a minimum, its mechanisms are automatic and are not selective concerning the particular host allowed to make queries. One of the most common attacks on DNS servers is "cache poisoning," in which the perpetrator replaces part or all of the name cache it builds with bogus entries.◆

Many vendors' routers also support additional functionality, such as encrypting network traffic sent from one router to another. From a security standpoint, routers are potentially very useful network components. On the other hand, if an intruder gains unauthorized access to or causes disruption of a router, the consequences can be extremely unpalatable for security. An intruder could, for example, change a router's ACLs or set the routing configuration to cause traffic to be sent over a fixed route over the network. Ensuring that this fixed route is used is important in a number of different types of network attacks, including attacks in which an attacker has installed a network traffic capture device on one of the machines along this route.

Because the Internet switches packets instead of using circuits to move packets from one point to another, we often refer to it as a packet-switching network. At this point, you might assume networking consists of an orderly flow of packets from one point through one more routers and finally to the destination point. However, this is not true. Packets may, for example, be too large for a router to handle. In this case, IP routers typically simply discard the packets. In other cases they are broken into fragments. When these fragments reach their destination, they are usually reassembled into a continuous data stream. The process of fragmentation and reassembly is normally transparent to users. In TCP/IP, a transmission's packets do not go over fixed routes, but instead look for the best, most efficient route available. Different packets from the same transmission might therefore go over somewhat different routes from the sending to the receiving host. Packets may arrive out of sequence; the receiving host normally can reconstruct the original order.

Network Layouts

The term *network layout* refers to the way that network computers, network components, and other physical parts of a network, such as cabling, are deployed. Three fundamental types exist: linear, star, and ring. The branching tree is also addressed, although this layout is actually two stars that have been connected.

Linear

In a *linear* layout, all computers are connected to the network on one network cable or segment. The linear layout was extremely common in the past but is rapidly being replaced by the star layout. It is now mostly used in networks to which relatively fewer machines (30) are connected. Figure 3.2 illustrates a linear layout.

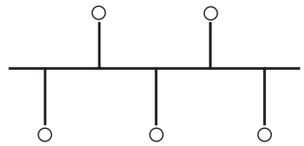

Figure 3.2 *A bus layout.*

Star

In a *star* layout, all computers are directly connected to a hub or a switch, which is a device that allows multiple cables to connect all at one point. Note here that if the interconnecting device in a star network is a hub, the network functions logically, just like a bus network, such that all clients receive all communications. Therefore, a star with a hub is also vulnerable to packet-sniffing attacks. Figure 3.3 shows an example of a star layout.

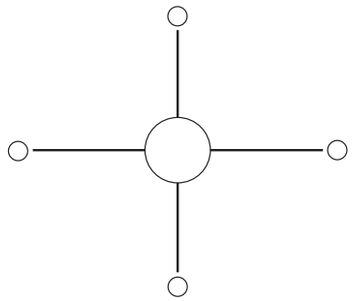

Figure 3.3 *A star layout.*

Branching Tree

In this layout, two or more star layouts are connected to each other (see Figure 3.4).

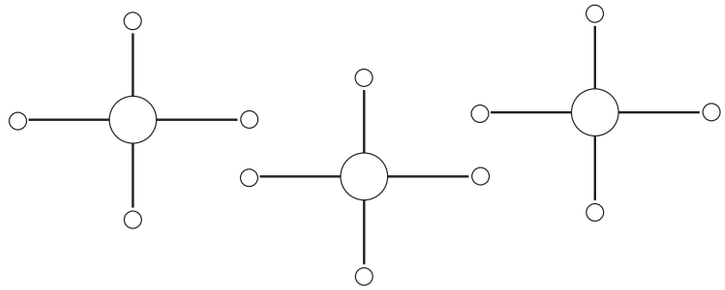

Figure 3.4 *A branching tree layout.*

Ring

A cable can connect to itself, thus forming what amounts to a self-enclosed circuit. In a *ring* layout, all hosts are connected to this circuit (see Figure 3.5).

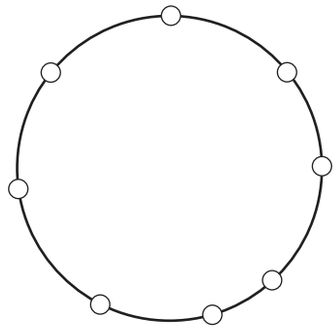

Figure 3.5 *A ring layout.*

Significance

Although other, more sophisticated manifestations of these layouts exist, all can in one form or another be characterized as one of these fundamental types. Modern Ethernets usually use the star and bus layouts together; the star connects systems to a bus together. Layout is also an important consideration in network security because some types of layouts are more vulnerable to certain kinds of attacks than are others. In the bus layout, for example, all computers listen to all packets sent over the network segment to which they are attached. This makes these networks particularly vulnerable to eavesdropping attacks in which one host has a program installed to copy every packet that reaches that host's NIC.

Local Versus Wide Area Networking

Windows NT networking is most fundamentally based on the Internet Protocol. IP addressing is based on the previously discussed *logical* addresses rather than physical addresses (for example, MAC addresses). Packets can therefore reach their destinations regardless of whether they stay within a local network or are sent across network boundaries to an entirely different network. The distinction between a LAN and WAN is, in fact, more arbitrary than anything else. Traditionally, the distinction between a LAN and WAN is based on physical proximity. A network with components entirely within a single building is therefore likely to be

considered a LAN. Similarly, a network that connects LANs dispersed over a wide geographical area is likely to be considered a WAN. More importantly, however, than distance over which a network spans is the protocol used. If a network runs Ethernet, it is a LAN. If it runs *Asynchronous Transfer Mode* (ATM) over leased lines, it is a WAN. If ATM switches are installed in a small area, the network should probably be called a LAN; in these cases, Ethernet runs on top of ATM. What is probably more important from a security perspective is the number and type of boundaries between networks. Consider the following examples:

1. Because an IP limited broadcast packet always has the destination address of `255.255.255.255`, it stays within router boundaries. Attacks such as broadcast flooding attacks (attacks in which one or more machines on the network send a high volume of broadcast packets, bringing the network to a virtual standstill), are very likely to be confined within a portion of the network served by a particular router.
2. The preceding example also applies to Network Basic End User Inteface (NetBEUI), a nonroutable protocol used in Microsoft network environments. Attacks that exploit some implementation of NetBEUI are likely to be initiated from *within* a network for the same reason.
3. NetBIOS is a naming convention and an API. NetBIOS traffic is routable if NetBIOS runs over a routable protocol, such as IP. NetBIOS typically runs over both TCP and IP. Because TCP and IP are robust protocols, attacks that expolit this protocol are therefore feasible from locations quite far away from a target Windows NT host.

The term *enterprise networking* is another interesting one. Although often applied to networking within large corporations with computing facilities in different geographical locations, this term is more of a marketing term than anything else. It has virtually no value from a technical viewpoint. Its usefulness in terms of understanding and dealing with security-related issues is therefore also minimal.

The OSI Model

Let's now move on to the next topic—the OSI model and its relationship to Windows NT networking.

The OSI model is a widely used model of viewing and implementing network functionality. This model specifies seven distinct, but interrelated layers of networking, starting with the physical layer and going up to progressively higher layers. Different network functionality occurs at different levels (see Figure 3.6). The layers are as follows:

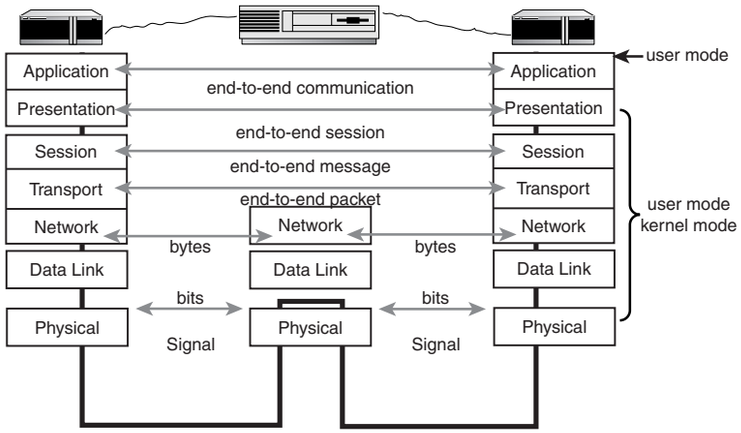

Figure 3.6 *The seven layers of the OSI and their interrelationships.*

- **Physical layer.** The layer at which electrical signals are transmitted over network wires; repeaters work at this layer.
- **Data link layer.** At this layer, data streams identified on the basis of their MAC address are transmitted to computers.
- **Network layer.** This layer supports logical addressability (independent of MAC and Logical Link Control [LLC] addresses) of packets that contain data, such as, source and destination address; routers work at this layer.
- **Transport layer.** Provides reliable end-to-end delivery between hosts, error detection and recovery, and flow control.
- **Session layer.** Whereas the network and transport layers provide connections between hosts, the session layer provides connections between applications on those hosts. The session layer provides a control structure for interprocess communication.
- **Presentation layer.** This layer is responsible for data transformation, such as character set translation, encryption/decryption, and compression.
- **Application layer.** The application layer, the "highest" in the OSI model, provides an interface to applications that run over the network. Note that the application layer does *not* consist of applications per se, only the interface to them.

How to Apply the Model

The OSI Model is, to a large degree, nothing more than an abstraction. Current implementations do not always decompose into neatly defined layers. Nevertheless, the OSI Model provides us with a very useful abstraction mechanism to discuss network funtionality. As two computers communicate across a network, the dialog begins with a top-down set of data transformations within the machine that initiates the communication. The application layer within this computer passes data to the next layer down, the presentation layer, which reformats the data and passes it to the session layer. The session layer again transforms the data for the next layer down, the transport layer, then the network layer, then the data link layer, and finally the physical layer. At this point electrical signals are transmitted along the network medium. The receiving machine receives these signals, which become progressively transformed as the data passes from one layer to the next higher one, starting with the physical layer. In this manner, every layer of one host thus effectively establishes a type of peer-to-peer communication with the same layer on the other host.

Although seven layers exist within this model, the model can be summarized in terms of four basic layers of functionality:

- Below the network layer, each layer focuses on providing an error-free signal from one host to another.
- At the network layer, traffic sent in the form of packets is routed to its intended destination within a network.
- At the transport layer, end-to-end communication between the sending and receiving host occurs.
- Above the transport layer, transmissions depend on the nature of each application involved in these transmissions.

Importance of Understanding the OSI Model

Not every networking expert embraces the OSI model. Critics argue that it adds undue complexity, and that looking only at application, transport, network, and lower layers is all that is necessary to depict that data transformations which occur within the source and destination hosts. In some networking contexts, this view is appropriate. Understanding the OSI model is, nevertheless, extremely helpful in understanding how to secure Windows NT and its many network protocols and services.

Although each layer presents its own set of special challenges from a security perspective, protocols and services at certain layers potentially pose more security-related risk than do others. Attacks, such as IP spoofing, a type of attack in which the attacking host masquerades as another one, capitalize on weaknesses in implementations of programs that utilize the IP

protocol, a Layer 3 protocol. The *User Datagram Protocol* (UDP), an efficient, sessionless, and connectionless protocol, poses a different set of security risks. UDP transmissions are sent from one host to another; if the destination host does not receive the transmission, no effort to notify the sending host that the transmission has failed occurs. UDP transmissions are therefore fundamentally less secure than TCP, which attempts to form a virtual connection between a sending and receiving host.

Services and Protocols

Now that you have had a look at the OSI model, the following section examines the Windows NT network environment's specific services and protocols and how they relate to this model.

Relationship to OSI Model

The Windows NT network environment is extremely complex with respect to the types and ranges of services and protocols supported. Figure 3.7 illustrates just how complicated this environment is. The intent here is not to present an exhaustive set of services and protocols in the Windows NT environment, but rather to provide a basic understanding concerning where within the OSI model some of the most fundamental and widely used protocols typically fit. Note also that the level at which a given protocol is utilized depends on the particular implementation based on that protocol, as explained shortly. In other words, one developer may implement a specific layer in a certain protocol, while another may implement the same funtionality in a different protocol.

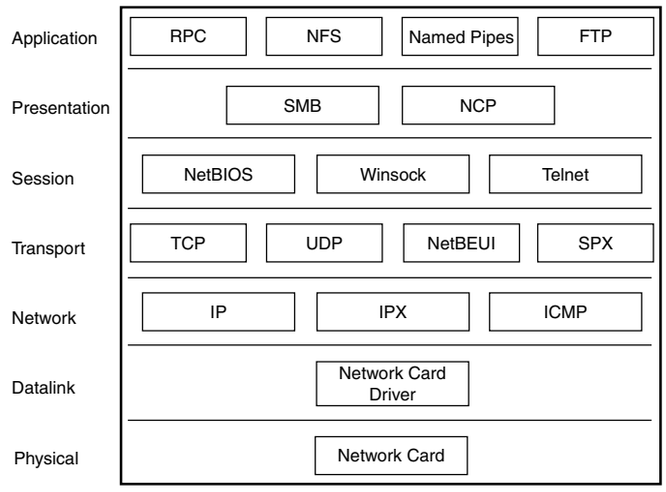

Figure 3.7 *Some major services and protocols in Windows NT networking.*

Layer 3 includes protocols such as IP and Internetwork Packet Exchange (IPX)—a mainstay protocol for Novell NetWare until release 5.0, but also the same protocol that runs in the Windows NT networking environment for interoperability with legacy NetWare hosts. Layer 4 includes the *Transmission Control Protocol* (TCP), the UDP, the NetBEUI, and *Sequenced Packet Exchange* (SPX) protocol (another protocol that in the Windows NT networking environment allows interoperability with NetWare hosts).

Relevance to Security

Although each layer presents its own set of challenges so far as security is concerned, certain layers potentially pose more security-related risk than do others. The physical layer, for example, is the layer vulnerable to packet-capture attacks in which network traffic is captured by a physical or logical (that is, a program) device. Attacks, such as IP spoofing, capitalize on weaknesses in implementations of programs that utilize the IP protocol, a Layer 3 protocol. The UDP, a Layer 4 protocol, is transmitted without provision for notification of the sending host in case the transmission fails or contains one or more errors. In contrast, another Layer 4 protocol, TCP, is a considerably more reliable protocol in that it provides a mechanism for notifying a sending host when something goes wrong with a TCP-based transmission. Therefore the UDP is, in general, more vulnerable to certain types of network attacks than is TCP.

IP Spoofing

IP spoofing is a type of network attack that can be directed against virtually any machine that processes IP traffic. The goal of an IP spoofing attack is to establish a connection between a client unknown to a server by making that client appear to be a legitimate client, and then to exploit a relationship between the server and the bogus client to gain unauthorized access. Here is a well-known way to perpetrate an IP spoofing attack:

1. Make the legitimate client unable to respond to the target server. This can be done by using a utility that "wedges" the legitimate client's ports—making the service or daemon that receives input from each port wait for input that will never come, thereby making the machine unresponsive to other inputs, such as connection request acknowledgements from other servers. This step is necessary because if the legitimate client were able to respond to the target server, the bogus client would not be able to "break in" to their communication.
2. Send a SYN packet from the bogus client to the target server to request that a connection be opened. This packet must indicate that the connection request is from the legitimate client (for example, must bear

the IP address of the legitimate client), even though in reality the packet must originate from the bogus client. The bogus client's request packet includes the *initial sequence number* (ISN) for that client.

3. The target server sends a SYN packet to the legitimate client containing data such as the server's ISN in addition to the client's ISN incremented by one. The legitimate client's ports are wedged, however, so the legitimate client will never respond to this packet.
4. The connection request is dropped if the client does not increment both ISNs (that is, for both the client and the server) and increment them in a manner that the server expects. The software running on the bogus client must then send a reply SYN packet containing the source address of the legitimate client with appropriate ISNs (one for the client and one for the server) incremented by one. Deriving the client's ISN is easy; this ISN is in fact an arbitrary number of which the server is not initially aware. The challenge is guessing the target server's ISN. The best clue concerning the value of the ISN the server has sent to the legitimate client in the first place is within the contents of already captured network traffic; packet dumps can reveal the previous ISNs for the server's connections to other systems. If the target server's ISN for a connection request from an entirely different client began with 24080 a few seconds previously and the ISN is always incremented by one for any new connection request, for example, the next ISN for a new connection is likely to be 24081 if the initial ISN is not random. IP spoofing software that returns a ISN of 24082 from the bogus client to the server would therefore be very likely to correctly anticipate the appropriate ISN.
5. If the bogus client sends the correctly incremented value of both ISNs to the server, the attacker will have established a connection between the two. The attacker can then attempt to exploit a relationship between the two machines to gain unauthorized access to the target server. Windows NT 4.0, for example, supports the (remote shell) **rsh** command that can allow trusted access from one machine to another without requiring that a password be entered.

Although predicted by Steve Bellovin in "Security Problems in the TCP/IP Protocol Suite," (*ACM Computer Communications Review*, Vol. 19, Issue 2, 1989, pp 32–48), the first reported IP spoofing attack was not observed until late 1994. For several years afterward, IP spoofing was one of the most frequently observed types of attack on the Internet. IP spoofing is not now as commonly reported as a few years ago, although it still poses a

potentially major threat to organizations, in that so many automated IP spoofing tools are so widely available. The best (albeit not infallible) countermeasure is deploying a firewall or screening router that blocks all incoming packets that indicate they originated from a host within the network protected by the firewall or screening router. This measure prevents spoofing attacks originating from outside one's network, but does not prevent such attacks if they are initiated from within the same network.[2]

Although most observed IP spoofing attacks have targeted UNIX systems (in particular, implementations based on *Berkeley Standard Distribution* [BSD] UNIX), Windows NT is also vulnerable to these attacks. Unless Service Pack 6[3] for Windows NT 4.0 is installed on a given Windows NT host, that host will (under many connection contexts) linearly increment the server's ISN from one connection to the next in a predictable manner, making the machine extremely vulnerable to IP spoofing attacks. Service Pack 4 for Windows NT 4.0 causes a server that receives an IP connection request to generate a reasonably random ISN in the SYN packet it returns to the requesting client, virtually precluding the possibility of IP spoofing. Several ways exist to verify that this service pack has been installed in 4.0 systems. One way to do this is to bring up the Command Prompt and enter **winver**.

If your Windows NT installation includes Service Pack 4 or up, you will obtain the following type of output from an About Windows NT dialog box (see Figure 3.8).

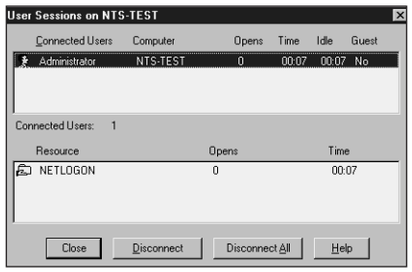

Figure 3.8 *User Panel.*

[2] *IP spoofing is most often initiated by external clients to deceive internal servers that these clients are internal.*

[3] *Actually, this was changed in SP4, not SP6. However, after SP6 it was discovered that there was a bug in the ISN generation. Therefore a new hotfix was issued. It is described in Q243835, and is available for SP4-6.*

Another way you can determine that latest service pack installed is to invoke a Registry Editor, and then check the value of the following key:

Hive: `HKEY_LOCAL_MACHINE`

Key: `Software\Microsoft\WindowsNT\CurrentVersion\`

Value: `CSDVersion`

Another way to determine the service pack level on the local host, as well as hosts across the networks, is to run the SPQuery utility from MTE Software (`http://www.mtesoft.com`). It lists the numbers of installed service packs as standard screen output.

Finally, another way to prevent IP spoofing attacks is to use a firewall or router with ACLs to reject traffic coming into a network that bears the source IP address of any internal host. Any incoming traffic with an internal source address is almost certainly bogus (and could indeed indicate an IP spoofing attack); legitimate incoming traffic should bear the source IP address of some *external* host.

About the Protocols Themselves

The Windows NT networking environment includes a wide variety of protocols, virtually all of which affect security one way or another. What are some of these protocols? Where do they fit into the OSI Model? This section explores these important issues. Figure 3.7, which shows some (but certainly not all) of these protocols, provides an initial overview. Consider the following protocols.

- **Remote Procedure Call (RPC).** RPC is a UDP-based protocol used in setting up communications, such as negotiating the particular ports to be used in establishing a connection.
- **Network File System (NFS).** NFS is a protocol (generally based on UDP) for file sharing that enables a user to connect to remote disks as if connecting to the local machine. Several Windows NT–based NFS implementations (typically based on the SMB protocol) are currently available.
- **Named pipes.** Named pipes are mechanisms that provide a direct channel to services that support applications. They are advantageous because they allow programs to reach them by referring to a name instead of requiring that a full path be specified.
- **File Transfer Protocol (FTP).** FTP is a TCP-based protocol for establishing sessions in which files are transferred between computers.
- **Trivial File Transfer Protocol (TFTP).** TFTP is similar to FTP, although it is not a connection-oriented protocol. Based on UDP, it is used to

download fonts and configuration files to hosts that broadcast their needs. Because it does not confirm whether data are actually being sent to their destination, it involves less overhead to create and maintain connections than does FTP.

- **Simple Mail Transport Protocol (SMTP).** SMTP is the protocol that establishes the structure of Internet mail through a special syntax. It also defines the conventions for setting up SMTP connections, transmitting both the sender and receiver's addresses, and sending the subject and main body of mail messages.[4]
- **Server Message Block (SMB).** SMB is an implementation of redirectors. Redirectors handle client requests for access to remote resources on a drive with a shared directory or another type device (for example, a printer) by taking these requests and reformatting them according to the needs of the protocols that will process these requests. Finally, SMB forwards the requests to either a lower- or higher-level protocol.
- **NetWare Core Protocol (NCP).** NCP is a protocol implemented in Novell NetWare. NCP is used in the Windows NT networking environment for access to remote resources on NetWare machines.
- **Network Basic Input Output System (NetBIOS).** NetBIOS is an *Application Programming Interface* (API) used in the Windows environment to provide session-layer connectivity between machines.
- **Winsock.** Winsock is not a protocol; it is instead a socket (a combination of a service and port) used by APIs for client applications regardless of the underlying protocol. It is commonly used to provide network-based access to Windows applications.
- **Telnet.** Telnet is a protocol that furnishes a command-line interface for emulating a virtual terminal on a remote computer. This in turn enables users to interact with a remote computer.
- **Network Basic End User Interface (NetBEUI).** NetBEUI implements the transport layer and is only used in smaller, local networks because of the fact that it is unroutable. All things considered, NetBEUI is a relatively fast protocol with low overhead.
- **Transmission Control Protocol (TCP).** TCP is the most common transport protocol used today. It is a connection-oriented transport layer protocol that verifies packets sent by one machine (the source machine) arrive at the destination machine. TCP thus establishes a virtual connection between two machines.

[4] *While there are SMTP implementations for NT, it is not a standard protocol.*

- **User Datagram Protocol (UDP).** UDP is a connectionless protocol that does not have built-in guaranteed delivery. One of its main advantages is that it involves less overhead than connection-oriented protocols such as TCP.
- **Sequenced Packet Exchange (SPX).** SPX is a transport layer protocol used primarily in contexts in which Windows NT hosts access Novell NetWare computers.
- **Internet Protocol (IP).** In many respects, IP is described as the protocol that provides the underlying functionality of nearly all higher-layer Internet protocols. Its functionality includes addressing (which includes checking packet headers to determine whether the information therein is correct), fragmentation (in case a router received packets that are too large to handle), and determining each packet's time-to-live (which, if expired, causes the packet to be discarded). IP handles each packet completely independently of any other packets sent over the network.
- **Internetwork Packet Exchange (IPX).** IPX, a routable protocol, is a very fast and highly established protocol, but it cannot be used on the Internet because it supports a different addressing convention from the one used by the IP (see the following section). Novell developed IPX/SPX for use in NetWare, but Microsoft has developed its own implementation of this protocol, the NWLink protocol. This protocol is completely compatible with Novell's IPX/SPX implementation.

The default protocol in most Windows NT network applications is a protocol suite called NetBIOS over TCP/IP (NBT). Numerous problems with NBT have emerged over the years due to dependencies between the NetBIOS and TCP layers of networking, in addition to other reasons. Performance and denial-of-service problems have resulted. As you will see shortly in the section titled "NetBIOS and SMB-Based Vulnerabilities," the NetBIOS layer is also filled with perils and pitfalls, among the more notable of which are dependence on primitive lookup mechanisms to retrieve NetBIOS name data and the capability to crash remote hosts by sending malformed NetBIOS packets or packets with illegal parameters. These and many similar vulnerabilities are discussed in more detail throughout this chapter.

Note again that the protocols described in this section by no means constitute the full range of protocols in the Windows NT networking environment. These protocols are some of the most commonly found ones that also often pose the most serious security-related threats. They are collectivel, only a portion of the possible protocols found within the Windows NT networking environment.

NetBIOS and SMB-Based Networking[5] in Windows NT

The chief focus of this section is the SMB protocol implemented in Windows NT (and the vulnerabilities it presents), although the NetBIOS protocol deservedly gets some of the attention. The *Common Internet File System* (CIFS), which implements the application and presentation layers of the OSI model, is Microsoft's implementation of SMB (a draft standard proposed by Microsoft and several other companies).

Introduction to SMB

SMB packets can be up to 65536 bytes in length, not counting an optional field in the header that indicates how long the packet is. SMB communicates directly with the NetBIOS layer, but it also works in conjunction with lower-layer protocols. At the transport layer, for example, it adds 4 bytes of data concerning transport handling initially transmitted with the TCP stream. SMB itself cannot determine which particular TCP packets are part of each individual session that has been established, so it turns over to TCP the responsibility of actually segregating the sessions. SMB can also, however, work in connection with UDP. In this case SMB sends 12 bytes of filler data to help the receiving system put the packets back in their intended sequence.

SMB also has a built-in mechanism that allows a system to transmit multiple requests within the same session. This allows sending requests in parallel rather than serially. The name of this mechanism is AndX; it is a legacy mechanism from earlier days of networking resulting in part from the need to increase the efficiency of connections.

Structure of SMB Packets

Let's look at the structure of the SMB packet. Associated with each offset is a field that carries designated types of data. Table 3.1 lists each offset, field name, and field length in bytes.

- Note that the first offset contains the optional 4-byte length integer field mentioned previously.
- The header start, a mandatory field, starts at offset 4 and is also 4 bytes in length.

[5] *Many of the facts and ideas in this chapter were developed by Hobbit, an independent vulnerabilities researcher (see* `http://www.avian.org`*). His white paper, "CIFS: Common Insecurities Fail Scrutiny," was the basis for most of the content of this particular section of the book.*

- The SMB command field, 1 byte long, is used to communicate to the receiving system that the packet contains a particular SMB command used in a es the "Key" communication.
- The smb_rcls and smb_err fields, both 2 bytes in length, begin at offsets 9 and 11, respectively. These fields, used to transmit, indicate when something goes wrong in these communications and what the nature of the error condition is.
- The 1-byte smb_flg and 2-byte smb_flg2 fields, beginning at offsets 13 and 14, respectively, are fields used to set flags that indicate the status of the communication between SMB-capable hosts.
- The next field beginning at offset 16 is reserved for data in TCP and UDP information that indicates the sequence of packets.
- The TID ("Tree ID") field (starting at offset 28) is for the 2-byte tree ID, a value used for the "tree connect" mechanism that indicates exactly where in the file system on a host to connect when a session that allows such a connection has been established.
- The PID ("Process ID") field (2 bytes long, starting at offset 30) indicates the process ID of the process associated with the network application on the machine sending the packet.
- The UID is an arbitrary user ID (unrelated to the NT logon ID); this field begins at offset 32 and is also 2 bytes long.
- The MID (2 bytes long, beginning at offset 34) is for the machine ID.
- Offset 36 is the beginning point for the 1-byte parameter word count, a field that tells the receiving system how many of the subsequent smb_vwv fields will follow. These smb_vwv fields contain data used in parsing the data that follow in the variable-length buffers.

Table 3.1 *The Structure of the SMB Packet*

Offset	Name	Number of Bytes
0	Length integer (optional)	4
4	Header start	4
8	SMB command	1
9	smb_rcls	2
11	smb_err	2
13	smb_flg	1
14	smb_flg2	2
16	(filler)	12
28	TID	2
30	PID	2

Offset	Name	Number of Bytes
32	UID	2
34	MID	2
36	Parameter word count	1
37	smb_vwv0	2
39	smb_vwv1	2
41	smb_vwv2	2
Variable	Buffers	Variable

Relationship with the NetBIOS Layer

Networked clients and servers need to how to communicate with each other. They frequently need, for example, to know each other's IP address and computer name. In conventional TCP/IP-based networking, DNS readily provides this kind of information. In the SMB networking arena, however, clients and servers not only require this type of information, but also need information used by the NetBIOS layer of networking. This layer, immediately below SMB in the OSI stack, has it own conventions for communications among other hosts on the network (that is, the NetBIOS names of other hosts). Capability to determine the NetBIOS name depends on a service called the NetBIOS Name Service (nbname). This service, for which the default port binding is UDP port 137, capitalizes on several different sources, including DNS, the *Windows Internet Name Service* (WINS), direct queries, and broadcasts, to build "name caches" that hold the information about other machines that NetBIOS mechanisms need. Using its own naming conventions, NetBIOS subjects the name that the system administrator assigns to a host to a series of transformations to produce a NetBIOS name.[6] Table 3.2 shows these steps.

Table 3.2 *How NetBIOS Names are Derived*

Step	Result
Convert characters of computer name to ASCII	"GLE" becomes ASCII 71 76 69
Convert ASCII characters to hexadecimal	71 76 69 becomes 47 4C 45
Add 20 until there are 16 byte pairs	47 4C 45 20 20 20 20 20 20 20 20 20 20 20 20 20

continues ▶

[6] *For more information on how to encode/decode NetBIOS names, refer to Knowledge Base article Q194203.*

Table 3.2 Continued

Step	Result
Split into two 168-byte hex strings	47 4C 45 20 20 20 20 20, *and* 20 20 20 20 20 20 20 20
Split each byte pair into nibbles (4-bit chunks) of individual bytes	4 7 4 C 4 5 2 0 2 0 2 0 2 0 2 0 2 0 20 2 0 2 0 2 0 2 0 2 0
Add hex value of "A" (41) to each byte pair	45 48 45 4D 45 46 43 41 43 41 43 41 43 41 43 41 43 41 43 41 43 41 43 41 43 41 43 41 43 41[7]

If the name is longer than 15 bytes, the additional bytes are truncated. Then a final byte is appended at the end to indicate the status or meaning of the NetBIOS name. Therefore, the NetBIOS name consists of a 15-byte unicast[8] hexadecimal value plus a 1-byte trailing hexadecimal value. This last byte pair can assume any of the hexadecimal values indicated in Table 3.3.

Table 3.3 Meaning of the Trailing Hexadecimal Values of a NetBIOS Name[9]

Value	Meaning or Status
00	Computer names and workgroup names*
01	Master browser
03	Messaging/alerter service; username of user with logon session
20	Names of available resources on server
1B	Name of domain master browser
1C	Name of domain controller
1E	Response to election announcement

You will see NetBIOS names when you use a packet-capture device to examine network traffic. If your protocol analyzer does not convert NetBIOS names back to computer names, you may have to perform this conversion to identify specific machines referenced in packets. Also, because NetBIOS includes mechanisms for identifying computers with NetBIOS and

[7] *The result if the NetBIOS name were to be converted back to ASCII would be EHEMEFCACACACACACACACACACACACA.*

[8] *Unicast is a format in which each byte pair is separated from the next by a space.*

[9] *This list is not exhaustive.*

other sessions to a given host, you can enter the **nbtstat** command into a Windows command line with the following options to remotely or locally identify these sessions in addition to other information:

```
nbtstat (-a targethost] [-A IPaddress][-c][-n][-r][-R][-s][-S][interval]
```

To get a dump of a host's sessions table with the destination IP addresses, for example, enter the following:

```
nbtstat -S
```

Similarly, given its host name you can get a host to dump its name table (its *name cache*) that it creates by entering the following:

```
nbtstat -a <hostname>
```

You can get a host to dump its name cache given its IP address by entering the following:

```
nbtstat -A <IPaddr>
```

Requests for name cache dump are often called wildcard node status queries. The value of the **nbtstat** command (the execution of which is not limited to priveleged users) to remote attackers goes without saying.

Client/Server Communications

How are SMB-based client/server communications established? A series of conventions are used in which both client and server pass data to each other until, if everything has gone right, the server establishes the SMB connection to a specific point in its hard drive or other device. This negotiation process occurs in four distinct stages: establishing a TCP session, negotiating a dialect, establishing a SMB connection, and accessing resources.

Stage One: Establishing a TCP session

The first stage of the negotiation process begins when a client sends a Session Request Block to the server. The names of both the server and client (in NetBIOS format) are included in the request. The server *does not* check the IP address of client, and immediately creates a TCP connection with the client on port 139 of the server (normally TCP 139) on the server.

Stage Two: Negotiating a Dialect

In this stage, the main purpose is to discover the "highest" version of SMB supported by both client and server. What this in essence means is discovering which implementation of MS Networks (the basic LAN implementation used in Microsoft networking) has been used in both the client and server. The particular implementation is known as the "dialect." The various types of dialects, starting with the earliest and ending with the most recent are as follows:

PC Network Program 1.0
XENIX Core
Microsoft Networks 1.03
LanMan 1.0
Windows for Workgroups 3.1a
LM1.2X002
LanMan 2.1
NT LM 0.12

The client sends the names of all dialects that it supports. Therefore if the client is a Windows NT Workstation 4.0, it will understand the most sophisticated dialect, NT LM 0.12. It will, however, forward to the server all the dialects in the preceding list. When the server receives the dialect name(s) from the client, it responds with a dialect list index showing which to use.[10] It also sets security flags (using the smb_flg and smb_flg2 fields in SMB packets) based on how sophisticated a version of SMB has been negotiated. It may additionally send 8 bytes of ciphertext for authentication; whether this is done depends on the particular authentication program. If any part of the dialect-negotiation process fails, the TCP session is terminated.

Stage Three: Establishing an SMB connection

The third stage of establishing SMB communications between a client and server once again starts with the client. The client forwards a session setup request containing a header that includes data, such as the user ID, domain name, operating system, and other (variable) information. The header is usually only 16 bytes long. The server responds by sending a token incorporating the same types of information included in the session startup request, but this time applicable to the server. Unfortunately as discussed earlier in this chapter, each user ID is not necessarily unique across sessions! If the client has sent a username that the server cannot recognize (that is, the user ID is unknown to the server), one of two results occur: denial of access (and, therefore, failure to create an SMB connection); or creation of a "null session," in which the SMB connection is nevertheless established but without a defined user ID. The big problem here is that the strength of authentication depends on the client, not the server. Incidentally, more negotiation must take place to establish an SMB connection if "share-level" security (an option in Win9x, but not in Windows NT) is in place.

[10] *The server also volunteers its file system type to the client during negotiation.*

Stage Four: Accessing Resources

In the fourth and final stage of the negotiation process between client and server, the server locates the specific point (for example, in the hard drive) to which the SMB connection will be made. The \PIPE\Srvsvc service is used to locate the device, and then a "tree connect" mechanism (based on the *Inter Process Communication* (IPC) mechanism in MS Networks) is used to access the resources (for example, files, directories, named pipes) within this device. The connection request includes the name of the requestor, TID, password (but only in the case in which share-level security is in place), and the name of the type of service being used to access the resource. The client then returns a 2-byte SMB header with the TID to the server after the tree connection is in place to acknowledge to the server that the connection request succeeded. At this point, a share connection is established, allowing the client to access resources with the restrictions imposed by both share permissions and the file system. (Chapters 5, "Configuring Windows NT Server for Security," and 6, "Maintaining Windows NT Security," provide details concerning how these restrictions work.) Because share connections time out if they are not active, a primitive type of SMB echo mechanism keeps the share active.

Null Sessions

Chapter 5 describes Windows NT accounts, groups, and privileges in detail. At this point, you need to be aware that an ID named SYSTEM exists on every Windows NT computer. This ID, which has almost unlimited privileges on the local computer, has no password—you cannot log on to this account. Privileged processes in Windows NT run, almost without exception, as SYSTEM.

Some complications arise when a service that runs as SYSTEM needs to access a remote resource. The destination machine does not recognize, nor give deference to, some other machine's SYSTEM ID. Because the service runs as SYSTEM, it does not have a password to present. The service therefore tries to set up an SMB connection with an undefined (null) username, empty password string, and undefined domain name during stage three of the client/server negotiation process described in the preceding section. The connection that results if the negotiation is successful is on behalf of the anonymous user and is therefore called an anonymous session or null session.

Security-Related Considerations with Null Sessions

Null sessions pose a dilemma. Certain null sessions are created on behalf of processes used in establishing trust between domains and also in other

contexts. You can use the Server Manager tool to see the sessions established on any given machine. Open Start, Programs, Administrative Tools, Server Manager. A list of servers in the current domain displays. Double-click any of them to see the sessions established to that server[11]. (Note, however, that null sessions are not limited to share and named-pipe access mechanisms to reach remote resources.) If you click on the Users button, you can determine exactly to whom each session belongs (see Figure 3.8). If you see Anonymous, you can be sure that it indicates that a null session to that computer exists. You can also determine whether null sessions have connected to resources by looking through a system's Audit log for data concerning the user named Anonymous. Although the system and domain do not have a user account for Anonymous, this entry is the one that results from connections opened by null sessions.

The real danger of null sessions is that they are designed to be used by the SYSTEM ID. Yet at the same time, they are unprivileged sessions that can be initiated by the ever dangerous Everyone group on the remote machine. The network result is a default absence of protection for the remote resources that can be accessed via null sessions. The potential for misuse should by now be readily apparent.

Solutions for Null Sessions

The dilemma in dealing with null sessions is that Windows NT, to some degree, depends on them. You could disable them, but at the cost of possible disruption of certain network services (for example, Windows NT's passthrough authentication used in establishing trusted access between domains).

By default, Windows NT systems at least restrict the way null sessions can be opened. They do not allow null sessions to be opened on any system through any existing share or named-pipe access mechanism. *Not* changing the default Registry setting that controls null sessions is in general a good move for the sake of security. It may, however, be necessary for the sake of some applications to override this default. In this case, use a Registry editor to reach the following path:

```
\System\CurrentControlSet\Services\LanmanServer\Parameters
```

If you want to allow null session access through shares and named pipes, you need only to create a subkey named RestrictNullSessAccess under

[11] *The Server Manager tool is installed by default only on Windows NT Server. You can install it on Windows NT Workstation, and even Windows 9x. It is part of the client tools, which are located in the* `\CLIENTS\SRVTOOLS\WINNT` *or* `WIN95` *directories on the Windows NT Server CD.*

Parameters and set its value to 0. (Note that the key type must be `REG_DWORD`.) Once again, making this modification is not advisable for security's sake in most settings, but it may be necessary to allow a select set of applications to work. To that end, you can specify shares that can be accessed by null users, regardless of the value of `RestrictNullSessAccess`. You can view and edit the names of the individual shares and named pipes through which null sessions can access system resources by examining and editing `NullSessionShares` and `NullSessionPipes` under Parameters. Each item that appears indicates a share or named pipe that can be accessed through null (anonymous) sessions. In this manner, you could selectively limit access via null sessions without disabling null sessions completely, thus reducing the potential for disruption. Finally, if you are running Windows NT 4.0 with Service Pack 3 or higher, you can make another Registry change to restrict access by the anonymous user (as described in Chapter 5).

NetBIOS and SMB-Based Vulnerabilities

NBT is the default network protocol in most built-in Windows NT network functions. This protocol consists of both the NetBIOS (the upper half) and the TCP/IP protocols (the lower half). NetBIOS serves as an abstraction layer in this arrangement. The NetBIOS protocol is used to communicate a considerable amount of information about the status of machines within the network, in particular the nature of processes and sessions running at this layer. Accordingly, this protocol can allow users and processes to query NetBIOS for information helpful in attacking Windows NT hosts and domains. Users with access to a Windows command line can, for example, initiate a node status query to discover information about other machines on the network. The data in the following example shows a data dump from a packet analyzer on a network in which an external user initiated a "wildcard query" (using the **nbtstat -a** command). Note that this query contains a string of 30 zeros followed by 2 zeros, the format used in NetBIOS names. The external user (possibly an attacker) is requesting that the target host dump its NetBIOS name table, a very dangerous request indeed.

```
"Flags""Frame", "Delta
Time", "Destination", "Source", "Protocol", "Summary"
"M",       1,             0, " [90.128.192.131]       ","[141.192.128.233]
","DNS","C ID=15 OP=QUERY
NAME=*<00000000000000000000000000000000><00>"
"   ",        2,        1., 5121, "[90.128.192.131]   ","[141.192.128.233]
","DNS","C ID=16 OP=QUERY
NAME=*<00000000000000000000000000000000><00>"
"   ",        3,        1., 5108, "[90.128.192.131] ","[141.192.128.233]
","DNS","C ID=17 OP=QUERY
NAME=*<00000000000000000000000000000000><00>"
```

In addition, certain commands can allow unauthorized persons to attempt access to administrative shares (default shares set up for the convenience of administrators who must remotely access the systems they administer). Attackers must, however, guess the password of one administrator to successfully do so.

Protocol Analysis in Windows NT Networking

Monitoring network traffic in the Windows NT network environment is extremely important, both in troubleshooting and in discovering possible network-based attacks. You can, for example, use a packet-capture device to determine whether users have initiated NetBIOS status queries (as previously shown in Figure 3.9). Windows NT has a built-in packet-capture program, the Network Monitor *(NM), that comes in two varieties. The simpler version comes with NT Server; it captures only the packets bound for the particular machine on which NM is installed. To install this version, open the Network Control Panel. Click on the Services tab at the top. Click Add. Scroll through the names of services that appear until you reach Network Monitor Tools and Agent, RefSeOK. There is also a Network Monitor Agent service. It is used by the* Systems Management Server *(SMS) version of Network Monitor to remotely capture data. Because of risks due to the potential for unauthorized access, it should not be installed unless there is a particular reason. Now reboot the server. Once NM is running, you need to go from Start to Programs to Administrative Tools to Network Monitor to see this output.*

Another, more powerful version of NM comes with Microsoft's (SMS), a system administration tool that you can purchase. The SMS version can connect to remote computers running the Network Monitor Agent and capture data from there. This latter version is more useful for system administration purposes because it allows packet capture for a potentially wide range of hosts. It also unfortunately poses even greater possible security-related hazards because it allows anyone who obtains unauthorized access to it to view network traffic for potentially many machines on the network. Installing this version is more or less parallel to installing the simpler version except for several steps. You must, for example, find the right path in the SMS Distribution Kit. First locate the top-level SMS directory within this kit, then go down one level to the `NMEXT` *subdirectory, then to your platform (Alpha or i386). Now invoke SETUP.EXE (path:* `SMS_TOP_DIRECTORY\NMEXT\I386\SETUP.EXE`*. Next start the Network Monitor Agent by clicking on Network, Services, Network Monitor Agent, Start, OK. Now reboot the SMS server. After installation, you can find Network Monitor Tools in Network Analysis Tools and the Network Monitor Agent in your system's Control Panel. You must also install the Network Monitor Agent on all computers for which you want to obtain packet dumps. You must then start the*

capture capability for each of these computers from Network Monitor's Capture menu. Finding the specific capture file for each computer may not be trivial. Choose the Find All Names option from the Capture menu to set the computer name that goes with each particular IP and MAC address. To read the logs for the first time, you may also have to experiment a bit by selecting a variety of different names from the Capture menu until you discover the particular log for each system that runs the Network Monitor Agent.

Regardless of which version of NM you choose to run, several critical security considerations apply. Having NM packet dumps fall into the wrong hands enables a perpetrator to potentially obtain passwords (some or all of which may be clear text) and data transmitted across the network. Clear text passwords traversing any network pose a high level of security risk because anyone who captures them can use them to illegally log on to systems. You should, therefore, ensure that permissions on your SMS console allow access of any kind only to SMS administrators.

Note that you need to do more than just run a program or suite of programs to capture network traffic. Remember, too, that NM is only one of many available options for dumping packets. Other alternatives include NetXRay by Network General, Surveyer by Shomiti, LANDecoder by Triticom, Observer by Network Instruments, EtherPeek by AG Group, and many others.◆

NetBIOS Solutions

Although piecemeal solutions that control against certain types of unauthorized access exist, no effective, comprehensive solutions for controlling NetBIOS vulnerabilities currently exist. Consider one solution—disabling the bindings between the TCP/IP and NetBIOS layers of networking. To disable these bindings, bring up the Control Panel, double-click on Network, and then click on the Bindings tab. The screen shown in Figure 3.9 displays. To disable NetBIOS over TCP/IP, click the plus sign next to NetBIOS Interface, select WINS Client (TCP/IP), and then click the Disable button. This effectively eliminates the possibility of a long-range attack that exploits weaknesses in NetBIOS, but again at a cost. Disabling these bindings interferes with services such as share access in wide area networking because the NetBIOS protocol (which is not a particularly good wide area protocol) depends on the underlying TCP and IP layers (both of which are well suited for wide area transmissions) in wide area networking.

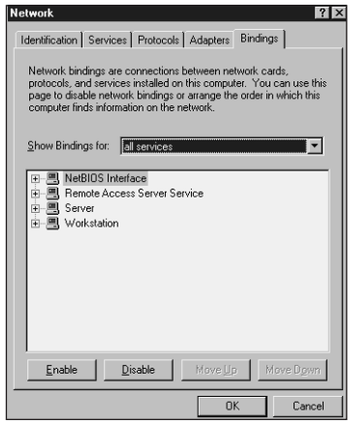

Figure 3.9 *Network panel.*

Consider a second solution, the Policy Editor, a tool in the Windows NT 4.0 Resource Kit that disables all Administrative Shares.
Consider yet another solution: disabling the administrative shares.[12] To disable these shares on a server, set the following Registry value:

Hive: `HKEY_LOCAL_MACHINE`

Key: `System\CurrentControlSet\Services\LanmanServer\Parameters`

Value: `AutoShareServer`

Type: `REG_DWORD`

Data: 1 (default) to create these shares, 0 to disable them.

On workstations, the procedure is the same, but the eegistry value is named `AutoShareWks` instead.

Disabling these shares, however, is in general not advisable because doing so is likely to cause severe disruption or at least inconvenience in system administration activities. Several commercial backup utilities, for example, require remote access to systems via Administrative Shares as they back-up systems. In short, no genuinely suitable solutions for most environments currently exist. Until Microsoft discontinues basing its implementation of Windows NT networking on the NetBIOS API (something that Windows

[12] *When Windows NT boots, it automatically creates certain "administrative" shares. These are so named because they are only available to administrators and are used for administrative purposes. Administrative shares are created for the %systemroot% directory, as well as for all hard drives.*

2000 does), security-related concerns with NetBIOS-related vulnerabilities will continue to be a source of worry.

Problems with SMB/CIFS

Microsoft calls its implementation of SMB the Common Internet File System. The following are some of the most serious types of SMB/CIFS related vulnerabilities and the ways these vulnerabilities could be exploited:

- A perpetrator may repeatedly attempt to gain access to a share even though denied access—the perpetrator may get share access anyway! Flooding a server with share requests can cause it to become so overloaded that it fails to process some of them normally.
- An attacker may break out of the share constraints that exist to gain access to the whole volume, even though the share attaches to a lower point than the root of the volume. This attack, commonly called a "dotdot" attack, requires that the perpetrator repeatedly use the command line (to enter **cd..**, then **cd..**, and so forth, repeatedly) after a TID connect is in place to gain access to a higher point in the file system than the TID specifies. SMB/CIFS-based connection mechanisms may ignore the TID if users repeatedly attempt to gain access to another point in the file system.
- Someone may exploit differences in naming conventions among compatible operating systems to attempt to gain access to files for which access is not allowed.[13] The trick here is initiating an SMB degradation attack in which the client is an older release of Windows or runs a malicious routine that convinces the server that the client understands only a very primitive SMB dialect. The server assumes that the client understands only the 8.3 file naming convention. An attacker must now own one or more file(s) within the same directory or subdirectory that holds a file to which the attacker desires unauthorized access. The attacker may be able to use a command line to access a file that begins with the same 8 characters and has the last 3 character extension as the file to which the attacker has access, even though the file to which the attacker has access and the target file do not have the same name when the extended (256-character5 character) naming convention is used. For example, the target file's name could be MARKETING.XCL and the file owned by the attacker could be MARKETINXXX.XCL. The particular file that the attacker tor access is now a matter of chance; the attacker may now be considered

[13] *A Solution for this problem is to disable 8.3 naming by making a Registry change described in Chapter 6.*

the owner of the target file because of confusion over the naming convention used!

- A perpetrator can replay the SMB/CIFS connection creation sequence from a bogus client. If the attacker can do this soon after this sequence is captured through use of a packet-capture device or program, the target server may not be able to detect whether this sequence is from a legitimate client.[14]
- Someone may forge SMB/CIFS request packets. The server will create a TCP session because it does not check the identity of the client.
- An attacker may gain access to shares that currently exist or that have existed recently (even though the share is to another user ID). The relatively primitive conventions involved in CIFS-based networking allow someone to break in to an existing connection and take it over.
- Someone can access resources as an anonymous user through a null session, as exemplified by the mechanisms used by a program called Red Button. Red Button sets up a null session to a designated host, and then exploits the fact that the anonymous user is by default a member of the Everyone group on the target host to read certain key values from the Registry on that machine.
- Someone can send an incorrect IP address but correct computer name to allow rogue hosts to access resources on CIFS servers. Remember, these servers do not check the IP address of clients before creating a session for the client.
- A malicious user may attempt to abuse weaknesses in the CIFS implementation of the SMB protocol to cause denial-of-service. A fairly recent vulnerability of this type results from the way Windows NT Server 4.0 processes SMB logon requests. A perpetrator who does not even have an account on a target system can send an authentication request with SMB packets that contain illegal values. The victim host that receives these packets processes the authentication request anyway, but the illegal data corrupt memory in the kernel. One of two errors typically results:

```
STOP 0x0000000A
```

or

```
STOP 0x00000050
```

The server now will either reboot or hang displaying the notorious Blue Screen of Death. Microsoft has developed and distributed a patch for this

[14] *Fixed in Service Pack 3, with the SMB signing feature*

bug; the patch is incorporated into Service Pack 4 (but not 3). This vulnerability illustrates only one of a wide range of vulnerabilities of this nature.

How Serious Are SMB Vulnerabilities?

At the time this chapter is being written, the exposures resulting from Windows NT's implementation of SMB in many respects constitute the most serious type of Windows NT security-related vulnerability. *The basic problem is that many Windows NT network mechanisms and protocols incorporate too many legacy negotiation and connection methods*. The fact that the client for the most part drives the level of security in SMB connectivity is also a serious concern. The resulting exposures potentially allow unauthorized users to remotely obtain copies of critical objects, including the SAM database, proprietary applications, business-critical data, and the like, stored on any Windows NT Server. The sheer potential for denial-of-service is, however, probably the most serious concern in many operational environments.

Chapters 6 and 7 discuss solutions for SMB- and NetBIOS-based vulnerabilities. Many of these solutions involve adopting fundamental network security measures.

Future Solutions

The next release of Windows NT (Windows 2000) incorporates Kerberos, a powerful tool that controls against unauthorized access through use of encryption. Kerberos in and of itself may, however, not solve the entire threat of unauthorized file access using SMB-based exploitation methods because Kerberos does not provide protection at the level of underlying protocols. Chapter 10, "Workstation Security," explores this and related topics in greater detail.

Conclusion

The main purpose of this chapter has been to set the stage for subsequent chapters that describe solutions for Windows NT networking vulnerabilities. This chapter has covered the basics of Windows NT networking itself, describing what networking is, and the functions it serves. This chapter has gone over the types of networks, the major components within networks, and the topologies that characterize them. The OSI model presented us with a systematic view of how data is transformed as it is sent from one computer to the other across a network. The discussion then

turned to the services and protocols in the Windows NT networking environment. Windows NT networking is, in fact, no simple matter; it encompasses a large range of protocols and services, most if not all of which translate to challenges when it comes to attempting to implement security. This chapter covered the SMB and NetBIOS layers of networking that Windows NT shares and other network applications utilize. SMB/CIFS's method of establishing connections leaves quite a bit to be desired from the perspective of security. We delved into null sessions and discovered still another cause for concern. Finally, this chapter analyzed the major types of vulnerabilities in, and a few solutions with respect to, the SMB and NetBIOS layers of networking.

Chapter 4, "Basic Windows NT Security Exposures," examines other types of vulnerabilities. In the remaining chapters of this book, we once again look at Windows NT networking mechanisms, but from a different perspective. We will increasingly consider options for dealing with the many vulnerabilities inherent in the way Windows NT networking transpires.

Checklist for System Administrators

1. If you need high levels of security, change the Registry of critical servers to restrict null session access, or at least to limit access via shares and named pipes. Test these changes in a nonproduction environment first to ensure that they do not disrupt your operational environment.
2. Observing the same constraints as in step 1, consider disabling the NetBIOS bindings if security needs so warrant.
3. Ensuring first in a nonproduction environment that no application breaks, consider disabling Administrative shares if security needs so warrant.
4. Install the most recent SP in all your Windows NT hosts, minimally SP5.
5. Ensure that access to devices and programs that capture network traffic is properly limited—only a few of the most trusted system and network administrators should be given such access.