Securing Windows NT/2000 Servers for the Internet

A Checklist for System Administrators

By Stefan Norberg
November 2000
ISBN 1-56592-768-0
216 pages

Chapter 1
Windows NT/2000 Security

In this chapter:

The use of Windows systems as Internet servers presents security challenges. In contrast to most internal systems, systems connected to the Internet are directly exposed to security attacks from both unsophisticated and highly skilled attackers. The typical Windows NT 4.0 (and, more recently, Windows 2000) installation makes a Windows server an easy target for such attacks. Securing the Windows NT or the Windows 2000 operating system for Internet use is a complex task. The purpose of this book is to offer a strategy for making your Windows-based server configuration as secure as possible. This strategy has two basic parts:

1. Secure or "harden" any Windows server that will be exposed to potential attacks from the Internet so it is as secure as it possibly can be. An exposed system of this kind is typically known as a *bastion host*.

2. Provide extra security protection for such exposed systems by installing an additional network -- typically known as a *perimeter network* -- that separates the outside network (usually the Internet) from your organization's internal networks.

Later chapters of this book describe specifically how to harden your Windows NT or

Windows 2000 system so it can function on your perimeter network as a secure bastion host. Before I present the step-by-step security details, this chapter sets the scene by describing briefly the security threats your system will face, the architecture of the Windows NT and Windows 2000 operating systems, and the recommended placement of Windows servers on your perimeter network.

Internet Threats

An Internet server faces many different kinds of threats. The most common include:

Intrusion
: An intrusion occurs when an unauthorized person gains access to your system. These days, intrusions most often result in web page defacement: an attacker alters the contents of your web site. Such attacks are growing in popularity. Attrition (http://www.attrition.org/mirror/attrition/) maintains a daily updated list of defaced web sites. The current record is 56 reported defacements in one day (November 21, 1999). About 60% of the defacements recorded at Attrition between October 1999 and April 2000 have occurred on Windows NT systems.

Denial of service
: The goal of a denial of service (DOS) attack is to sabotage operation by consuming all of your computing resources (CPU time, network bandwidth, etc.). This effectively stops authorized users from using the system.

Information theft
: This type of attack occurs when an unauthorized person obtains private information. The most popular targets are login/password information, credit card information, and software source code.

Many intrusions are made possible by improperly configured software. Looking at a concrete example may help underscore this point. Recently, the Apache web server site (http://www.apache.org/) was hacked.[1] In this particular case, the attackers uploaded a PHP[2] script to a world-writeable FTP directory. The web server root directory was the same as the FTP server root directory. This allowed the attackers to launch Unix commands using the uploaded script. They uploaded and executed a shell binary that bound to a high port and enabled them to initiate a connection to that port. The attackers now had interactive shell access on the system. The next step was to gain root access. This was accomplished by using a database process that was running as root to indirectly create a *setuid* root shell.

Fortunately, these attackers (so-called "gray-hats"[3]) were not out to thrash the site; they only replaced the "powered by Apache" logo with a Microsoft Back Office logo and alerted the site administrators.

The following configuration errors made the Apache break-in possible:

1. The web server and the FTP server had the same root directory. This allowed the attackers to upload the software that was used to launch the attack. The uploaded software could be executed because the web server software used the same filesystem hierarchy.

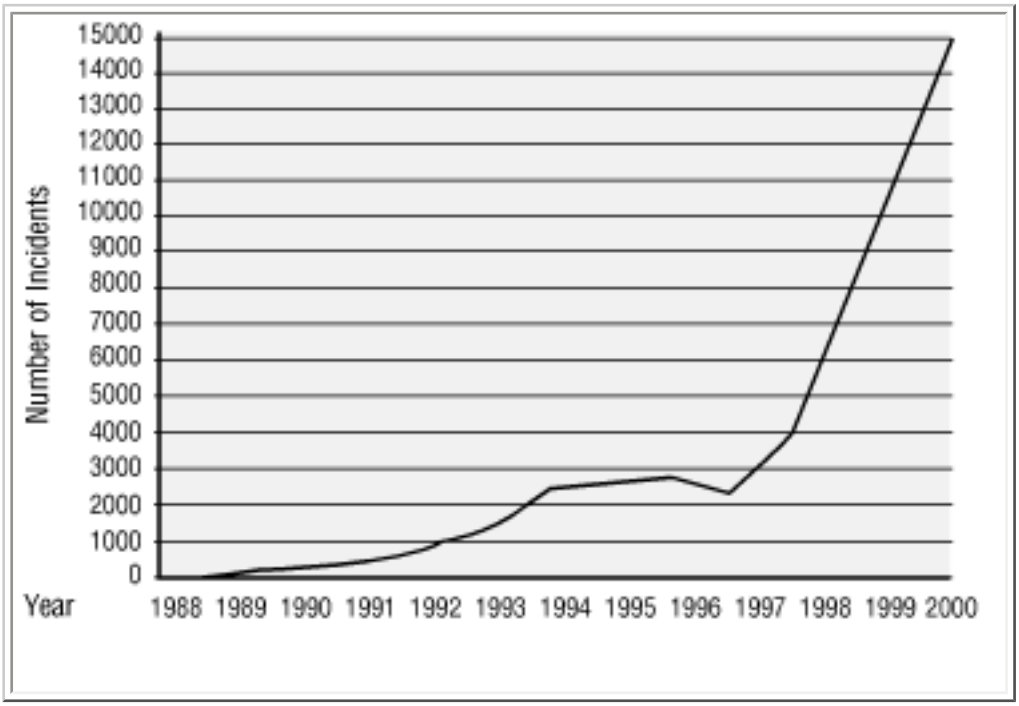
2. There was no (or an improperly configured) firewall system protecting the web server. It was possible for the attackers to connect to any port on the system. This made the attack much easier.

3. The database software was running as root. This is the reason why the attackers were eventually able to gain root access.

Windows NT systems present many vulnerabilities, which attackers are only too happy to take advantage of. For example, there are cases where an attacker has been able to connect directly to a system using Windows file sharing. Those systems are an even easier target than the Apache site was. Just start guessing passwords and try to connect as Administrator!

The number of security incidents reported to the Computer Emergency Response Team Coordination Center (CERT-CC)[4] has grown at an alarming rate in recent years. Figure 1-1 illustrates this development; note how steeply incidents have increased since 1997. (Incidents include, but are not limited to, attempts to gain unauthorized access to a system or its data, and disruption or denial of service.) The real security picture is far worse than these statistics show; it's safe to assume that only a small number of all incidents are reported to CERT-CC.

Figure 1-1.Number of incidents reported to CERT-CC



If you already have a presence on the Internet, you probably know that attempts are made to compromise your site's security mechanisms every day. And the stakes are high. Imagine how you'd feel if an online store that you use was hacked and the attacker managed to steal your credit card information. Would you feel comfortable shopping there again? Would you use an Internet bank that was successfully attacked last year? I wouldn't.

As a result, it's of great importance to have and maintain a high level of security for your site. This is a complex task, but after reading this book I hope you will find it somewhat less troublesome.

Keep in mind that even if you're not running a large online bank or shopping site, you still need to take steps to protect your servers. The attackers are out there; they may be after your intellectual property; they may want to use your computing resources; or they may just want to have some fun defacing your web site.

Building a Secure Site on the Internet

Building and maintaining a secure site on the Internet includes many more tasks than simply installing your operating system, however securely you may do so. Overall security is a combination of secure software and careful human planning and administration. You will need to be concerned with all of the following tasks:

Planning
: Securing an Internet site must be a carefully planned and coordinated process. It's not just a matter of clicking on screens and working it out as you go. Figure out the goals and tactics ahead of time, and then implement security, step-by-step. It's also important to understand that you need one encompassing plan that includes all aspects of the process, rather than several small and uncoordinated planning efforts.

Policies
: In order to achieve a high level of security, you need policies that define the main aspects of running an Internet site. This is not a book on policies, but keep in mind that before you start building a secure system, you need to have the appropriate policies in place. Start by reading the *Site Security Handbook* (RFC 2196); it's an excellent introduction to this topic.

Access control
: Access control protects systems from unauthorized use; there are several different types:

Physical access control
: Physical access control[5] is often overlooked, but it's an extremely important

outer level of protection. Large organizations often have big computer rooms that are both bomb-proof and earthquake-proof, which is good. In many cases, however, pretty much everyone in the organization has access to these rooms, which makes it possible for anyone in the building to sabotage operations.

System access control
: Only the people involved in the daily operation of your systems should have access to these systems. Those who are granted access should have only the amount of privilege required to do their jobs. For example, not everyone needs to be a member of the Administrator group.

Network access control
: Network access to your systems needs to be restricted by a firewall system. A *firewall system* consists of a number of components that act in concert to enforce your network access policy; it's typically not just one single gateway with firewall software installed. The perimeter network (discussed later in this chapter) is a type of firewall system.

Operation
: Once your system is up and running, you need to manage its operation in a careful and secure manner. System management includes:

Auditing
: Watch your systems carefully. Set up an audit policy that keeps you informed of any access policy violations. *Chapter 6, Auditing and Monitoring Your Perimeter Network,* deals with the different aspects of setting up auditing on a Windows NT/2000 bastion host.

Backups
: Make frequent backups. Always back up before and after changing the configuration of your systems. The flip side of backup is restore; you must attempt to restore your system from backups at regular intervals to make sure you'll be able to do so if there is a disaster. *Chapter 5, Backing Up and Restoring Your Bastion Host,* serves as an introduction to backing up and restoring bastion hosts.

Log management
: Collect logs in real time on a separate secured logging host and carefully review this information. Chapter 6 suggests a strategy using *syslog* as a transport mechanism for log collection.

Peer reviews
: Ask your colleagues or a third party to review your work periodically. See *Chapter 7, Maintaining Your Perimeter Network,* for details.

Encryption

Use encryption to secure communication and sensitive data stored on disk. You will find references to various types of encryption methods and algorithms throughout this book. The "Cryptography Basics" section later in this chapter provides a brief introduction.

It's important to understand that site security is a very big and complex subject and that this book's focus on the practical aspects of building and managing secure bastion hosts based on Windows NT/2000 is a very narrow aspect of site security.

Hardening the Bastion Host

Microsoft's success in the network operating system market is largely because its products are so easy to use. The Windows server version has the familiar user interface that almost all office workers use every day. It's easy to get started, and you don't need in-depth knowledge of the operating system to install a Windows NT/2000 server. Most components are configured and started automatically, just as they are in the consumer Windows 95/Windows 98 operating system. These characteristics are attractive for an internal file and print server that isn't exposed to direct attack. However, you want something quite different for an external web server that serves the organization's customers and partners over the Internet. A system exposed in this way should provide a minimum of services and needs to be properly configured to ensure a higher level of security. As I mentioned earlier in this chapter, a system configured in this manner is referred to as a bastion host.

Basically, a bastion host is a computer system that is a critical component in a network security system, and one that is exposed to attack. Examples of bastion hosts are firewall gateways, web servers, FTP servers, and Domain Name Service (DNS) servers. Because bastion hosts are so important--and so vulnerable--such systems must be highly fortified. You must pay special attention to fortifying (i.e., establishing the maximum possible security for) the bastion host during both initial construction and ongoing operation.

Why are such systems called bastion hosts? The *American Heritage Dictionary* defines a bastion as:

1. A projecting part of a rampart or other fortification.

2. A well-fortified position or area.

3. Something regarded as a defensive stronghold.

Marcus J. Ranum is generally credited with applying the term *bastion* to hosts that are exposed to attack, and with the popularization of the term in the firewall community. In

"Thinking About Firewalls V2.0: Beyond Perimeter Security"[6] he wrote:

> Bastions are the highly fortified parts of a medieval castle; points that overlook critical areas of defense, usually having stronger walls, room for extra troops, and the occasional useful tub of boiling hot oil for discouraging attackers. A bastion host is a system identified by the firewall administrator as a critical strong point in the network's security. Generally, bastion hosts will have some degree of extra attention paid to their security, may undergo regular audits, and may have modified software.

Bastion hosts are not general-purpose computing resources. They differ in both their intent and their specific configuration. The process of configuring or constructing a bastion host is often referred to as *hardening*.

The effectiveness of a specific bastion host configuration can usually be judged by answering two questions:

- How does the bastion host protect itself from attack?
- How does the bastion host protect the network behind it from attack?

Chapter 2, Building a Windows NT Bastion Host, and *Chapter 3, Building a Windows 2000 Bastion Host*, provide detailed instructions for building a bastion host, using Windows NT and Windows 2000 respectively.

Exercise extreme caution when installing software on bastion hosts. Very few software products have been designed and tested to run safely on these exposed systems. For a thorough treatment of bastion hosts, and on firewalls in general, I recommend reading *Building Internet Firewalls, Second Edition*.

Configuring the Perimeter Network

No matter how carefully you configure your bastion host to withstand direct attacks, you can't be entirely confident about its security. Most software code has bugs in it, and therefore all systems potentially have undiscovered security vulnerabilities. For this reason, it's important to provide extra layers of security for systems that are as exposed and as vulnerable as bastion hosts.

A common way to protect exposed servers on the Internet is to implement some kind of network-based access control mechanism that serves as extra protection for the bastion hosts. One such very effective mechanism is provided by a perimeter network. A perimeter network is a network that connects your private internal network to the public Internet or another untrusted network. This makes the perimeter network very important

from a security standpoint. The purpose of this network is to serve as a single point of access control. All components in a perimeter must act in concert to implement a site's firewall policy. In other words, the perimeter network is a firewall system.

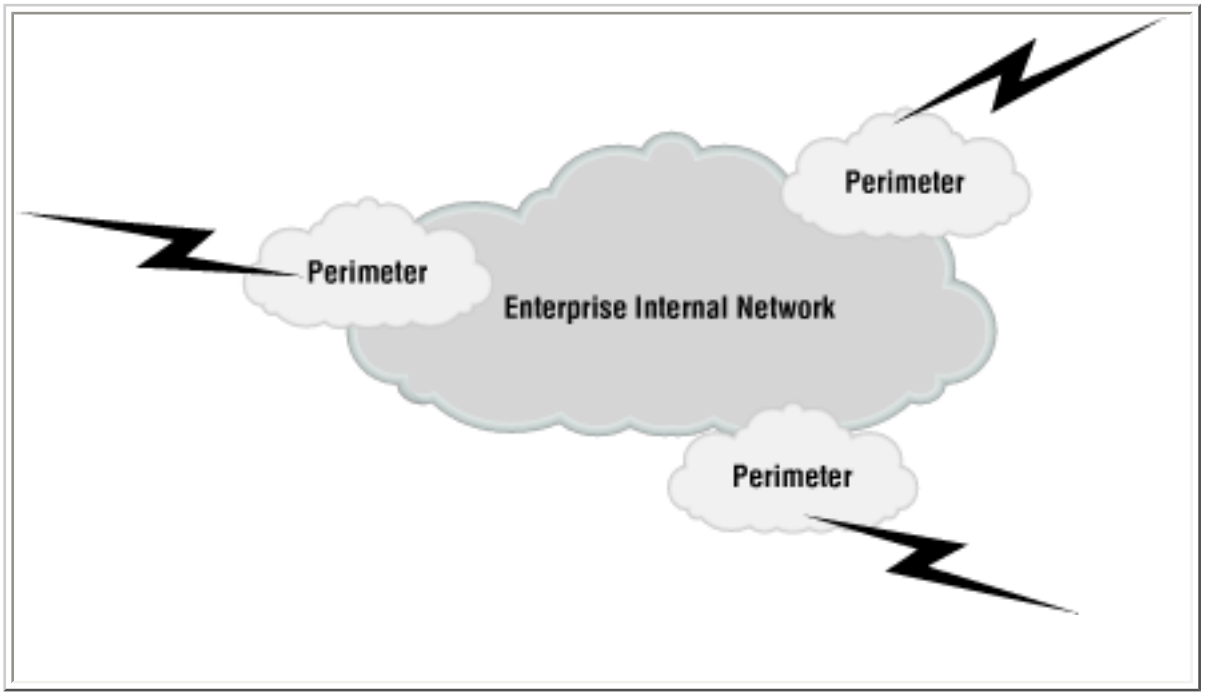
The perimeter network is a key part of the architecture of many current Internet sites. The reasons are partly historical. When the Internet took off commercially, many companies wanted to get on the Net to do business. The first step was often simply to publish product information on a web server. These web servers typically contained only static information, and thus didn't need to be connected to the internal network. With the advent of e-commerce, such web servers had to be connected in some way both to the clients on the Internet and to the legacy systems on the internal network -- for example, to process orders and check the availability of products.

Many companies now faced the requirement to connect their internal networks to the Internet--and to the accompanying security risks. Since the Internet could not be trusted for obvious reasons, there was an increasing need for company-controlled networks that could act as secured perimeters.

The perimeter network architecture

A perimeter network is an untrusted part of an enterprise network that resides on the outskirts of the private network. The perimeter network is often also referred to as the *demilitarized zone*, or *DMZ*, named after the region separating North Korea and South Korea. An example of a perimeter network is where the Internet connection and the web servers are located. A company might have several perimeter networks, as illustrated in Figure 1-2.

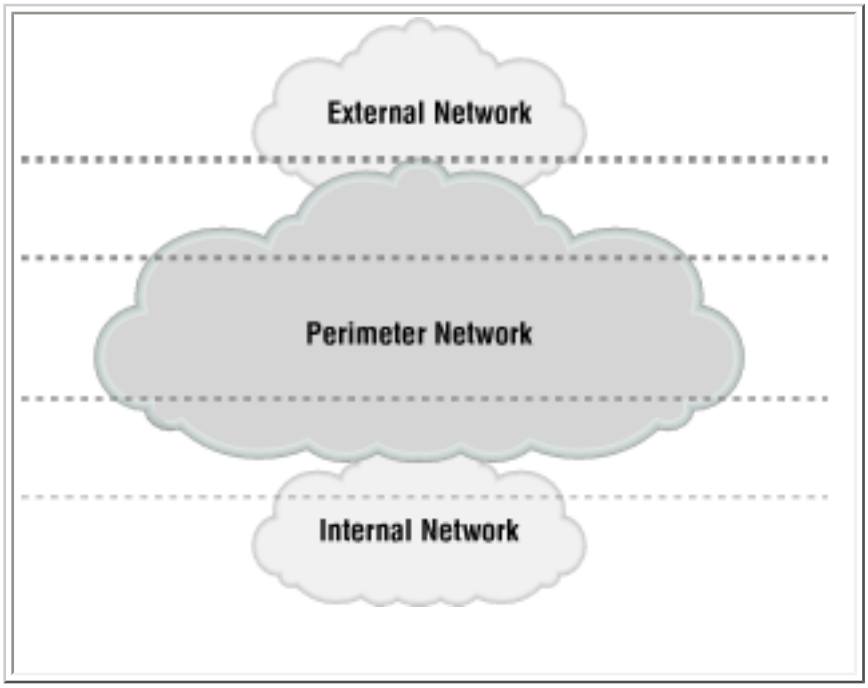
Figure 1-2.An enterprise with three perimeter networks



All external communication from the internal network has to pass the perimeter before it can reach an external host, and no communication is allowed directly from an external network to the internal network.

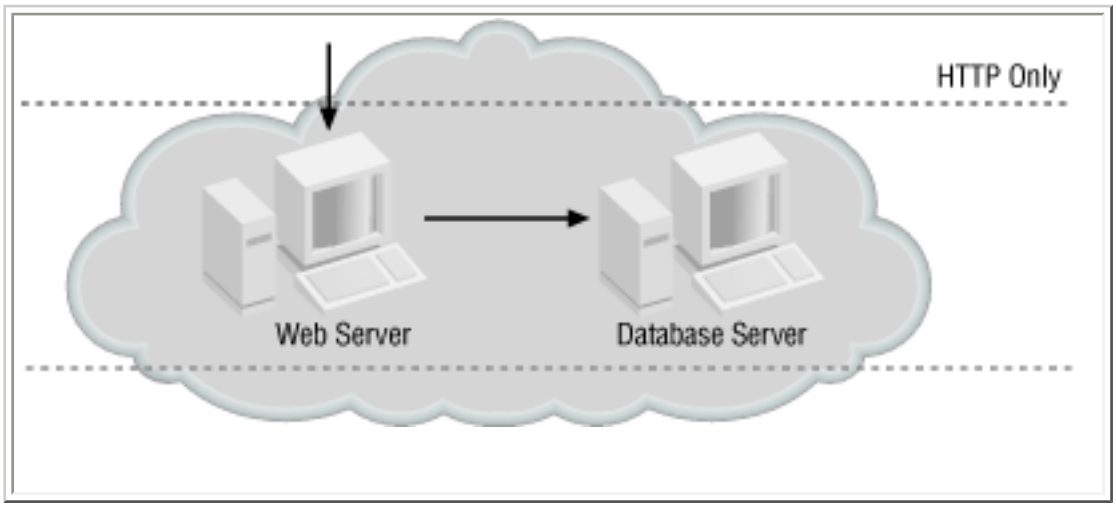
A good approach to building the perimeter network is to build it in compartments, so that the perimeter is able to protect itself and the internal network even if one compartment is compromised. This compartmentalization is illustrated in Figure 1-3.

Figure 1-3.Security zones in the perimeter



Because each compartment has access control mechanisms, the farther in from the external network a host is placed, the better it is protected. It's good security practice to block as much traffic as possible in each compartment layer--I recommend that you take a *default-deny* stance regarding network traffic. With a default-deny stance, everything that isn't explicitly allowed is denied, in contrast to a *default-allow* stance, where everything that isn't explicitly denied is allowed. Consider the example in Figure 1-4.

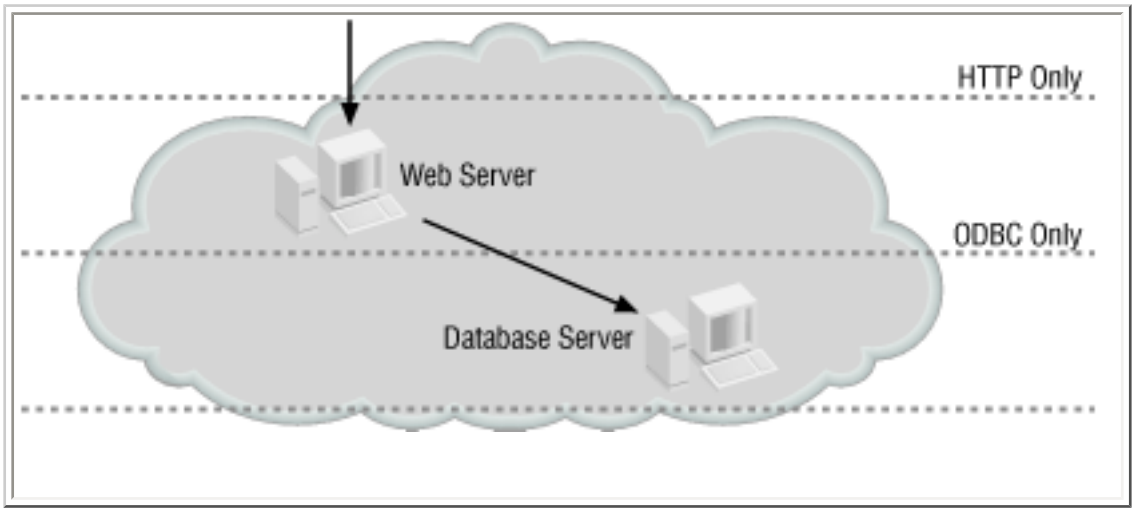
Figure 1-4.A perimeter with one security zone



In this example, if the web server is compromised, it's easy for an intruder to attack any service provided by the database server. This is because there is no network access control between these two servers.

On the other hand, in the topology shown in Figure 1-5, if the web server is compromised, the access control layer between the compartments will block unneeded traffic to the database system. As a result, the intruder may be able to attack the database process on the server, but not be able to attack anything else.

Figure 1-5.Perimeter with two security zones



Components in the Perimeter

It takes a number of different components to build a perimeter network, and some architectures are quite complex. This section does not attempt to describe all of the issues or possible combinations. It simply introduces the components and explains how they interact so you will have enough background to be able to understand subsequent chapters.

Routers

Routers are the traffic police of the network. They decide what route a datagram should take at each router or "network intersection." Like the police, routers can also choose to stop certain types of traffic. Traffic is controlled by rules called router Access Control Lists (ACLs). Example 1-1 shows a router ACL for a Cisco router.

Example 1-1: Cisco IOS Router ACL Example

```
ip access-list extended example_access_list
permit tcp any 192.168.1.0 0.0.0.255 eq http
permit icmp any any
deny ip any any log
```

A Cisco IOS ACL is applied from top to bottom. An incoming datagram is tested against each line in the ACL. This example allows HTTP from anywhere to the 192.168.1/24 network. It also allows any type of ICMP anywhere. All other datagrams[7] are blocked and logged.

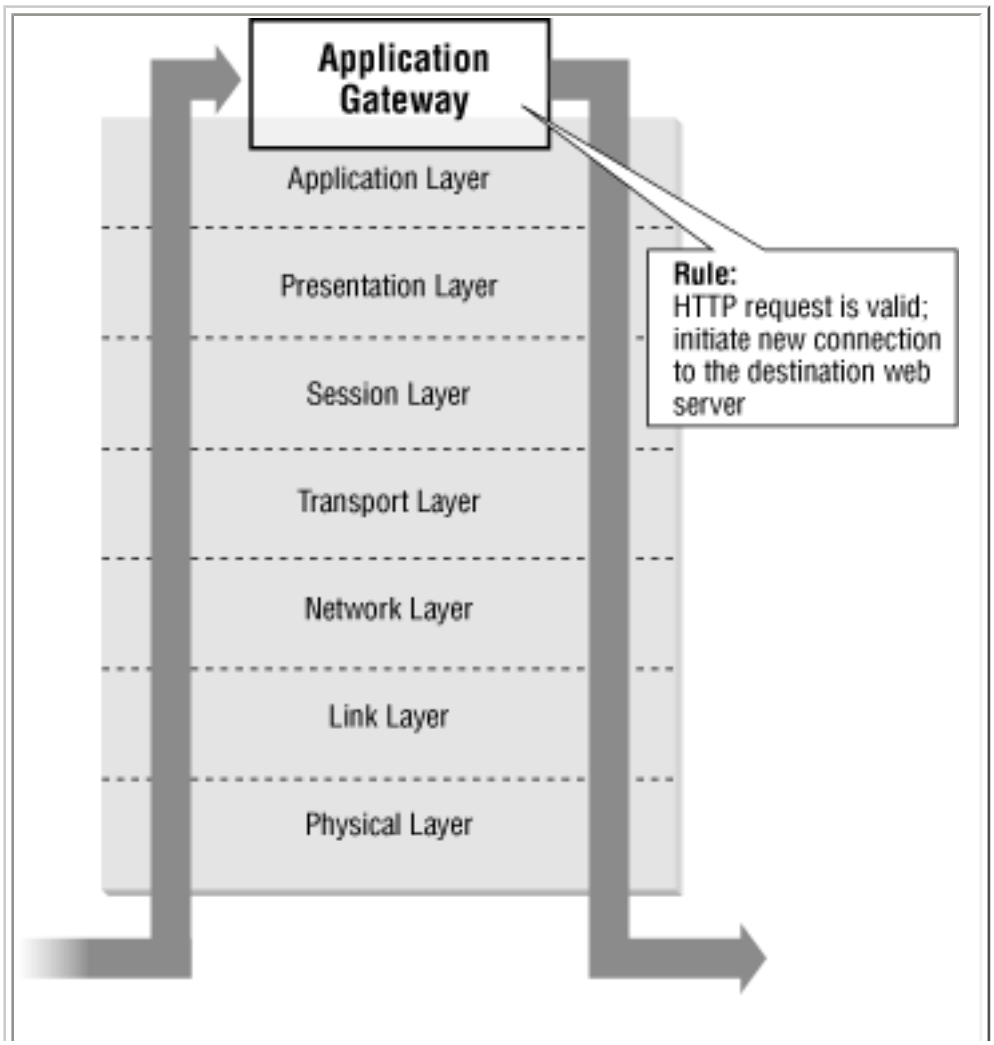
Using router ACLs in this manner provides us with a useful network access control mechanism in the perimeter. A router that implements access control in this manner is generally referred to as a *screening router*.

Firewall gateways

Certain components in the perimeter typically have firewall software installed, and these machines are referred to as *firewall gateways*.

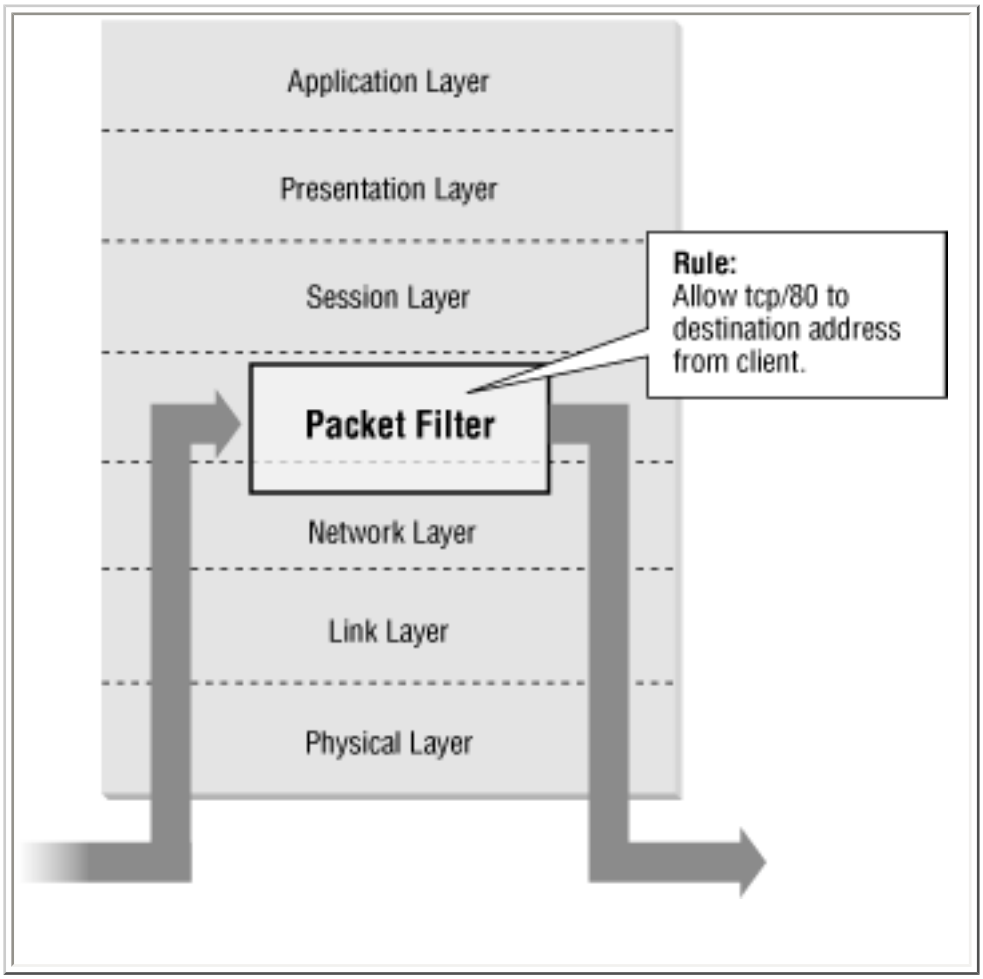
There are two common techniques that a firewall gateway can use. One method, shown in Figure 1-6, is to act as an application-level gateway; the gateway serves as a middleman that intercepts traffic at the application level, and it initiates a new connection to the target system on behalf of the client. Examples of application-level protocols are File Transfer Protocol (FTP), HyperText Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), and Post Office Protocol (POP).

Figure 1-6.An application-level gateway



The other technique, illustrated in Figure 1-7, is to inspect the traffic on the Internet Protocol (IP) level. This is called *packet filtering*. A more sophisticated form of packet filtering, called *stateful inspection*, is used by products such as Checkpoint's Firewall-1. A state-aware firewall gateway keeps track of the state of the connections that are going through it. If an outgoing HTTP request from a client is allowed through the gateway, the response to that request also has to be allowed through. The firewall software adds a temporary rule in its rulebase to allow the response from the destination web server to the client. The firewall gateway also understands some types of application data (HTTP, SMTP, FTP, etc.) in the IP datagrams, and for this reason, it may be able to make better security decisions than a screening router can.

Figure 1-7.A packet-filtering device



In theory, an application-level proxy can make more sophisticated decisions, but it's usually slower than inspecting the IP datagrams. As "inspection" technology gets more sophisticated (it's now becoming possible to keep track of the state of many application-level protocols), the gap between the two approaches lessens. Many firewall products provide both application-level proxies and IP-level inspection or filtering. These products are referred to as *hybrids*.

Bastion hosts

The bastion hosts are the application servers in the perimeter. A bastion host usually runs one specific piece of software, such as a mail gateway or some web server software. A bastion host has no unnecessary services running, and it is installed and configured in a highly secured manner, as described in Chapters and .

Switches and hubs

As with any other network, you need switches or hubs to build the network infrastructure in the perimeter. A network segment that uses a *hub* is a shared media, where all traffic is visible from all network stations (hosts). On the other hand, a *switch* connects the sender directly to the receiver for every Ethernet packet. This provides improved performance for unicast (one-to-one) traffic, but also some additional security. If one of the hosts on a network segment is compromised, an intruder may be able to install a network sniffer to spy on traffic on that segment to get information. However, if a switch is used, the intruder may not be able to see the traffic between other hosts.

I recommend that you use "dumb" switches and hubs without management software if possible. If the switch or hub device has its own IP stack, it may be vulnerable to attacks, and for this reason, it will have to be secured in the same manner as your bastion hosts.

A perimeter network example

The best way to describe how all the components fit together is to present an example perimeter network design. The example network I describe in this section is very general and simplified; don't use it as a ready-to-run implementation blueprint.

What are the objectives of this design?

A design always has to meet some core objectives. These are usually determined by specific business requirements. Let's assume that our example company has the following needs:

- It must allow access to the web servers from the Internet.
- It must accept incoming mail.
- It must allow outbound web and FTP from the internal network.
- It must allow outgoing mail.

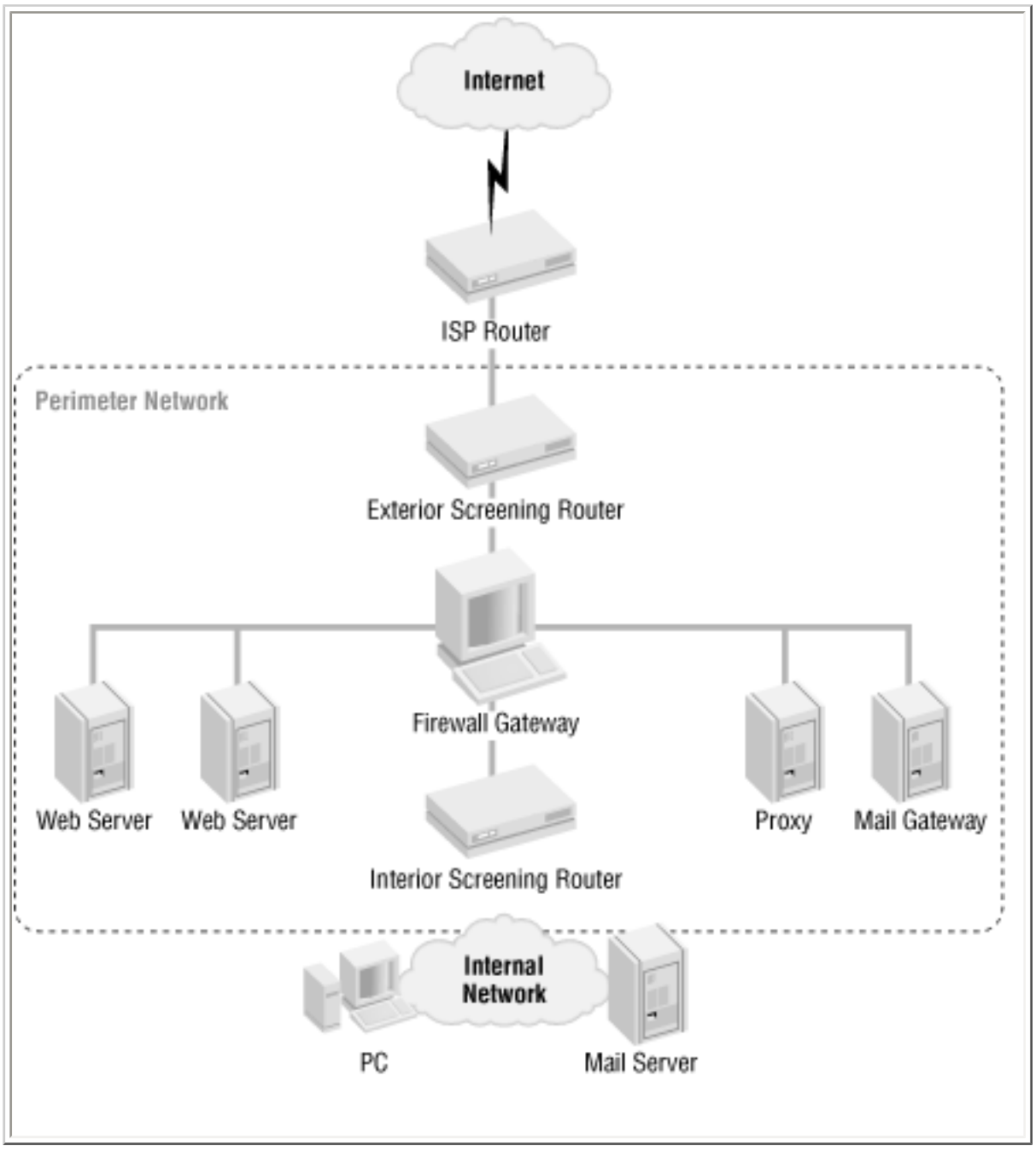
The example company must solve these business objectives with regard to two key network security needs:

- No direct traffic can be allowed between the Internet and the internal network.
- If one component in the perimeter is compromised, it should not result in a compromise of the entire perimeter or the internal network.

What's a possible solution to these problems and objectives?

The solution shown in Figure 1-8 meets all the objectives of this example and protects the perimeter both from external and internal threats. The solution implements a perimeter network with the web servers, a firewall gateway, a mail gateway, and an HTTP proxy server.

Figure 1-8.Example perimeter network



In this example, the following services are allowed:

- Only inbound HTTP to the web servers and inbound SMTP to the mail gateway are allowed from the Internet.
- The proxy server and mail gateway are allowed to do DNS queries (udp/53).
- The proxy is allowed outbound HTTP, HTTPS, and FTP.
- The mail gateway is allowed to send mail (SMTP) to the Internet and to relay incoming mail to the internal mail server.
- The internal mail server is allowed to relay outgoing mail to the mail gateway.
- The internal network is allowed to use the proxy server in the perimeter.

The screening routers protect the perimeter from the Internet and the internal network from the perimeter in case there is a problem with the firewall gateway. You'll notice that no direct traffic is allowed between the internal network and the Internet and vice versa.

Rules of Thumb for Perimeter Networks

Use the following rules when designing a perimeter network:

Default-deny
: Don't allow traffic that isn't absolutely necessary. This limits the number of network-based attacks to which you are susceptible.

Defense in depth
: Don't rely on a single mechanism to enforce your security policy. Build well-fortified compartments in your perimeter.

Keep it simple
: Complicated technology and policies are hard to implement and hard to understand. And if you have trouble understanding, it's easy to make mistakes. Avoid "bleeding-edge" technology. Use products that are proven.

Take it slow
: Try to not let the business side of your organization set deadlines that are impossible to meet. Take a phased approach. All bells and whistles don't have to be in place on Day One.

Planning is crucial

> Plan for security, capacity, redundancy, and manageability. If you don't have a good plan, you will probably end up redesigning everything in six months.

The proxy and mail servers are placed on a different network from the web servers; doing so separates outgoing web surfing from published web services. In the future, the company might consider a separate Internet connection for outgoing web traffic to guarantee bandwidth to its public web servers.

This design has four separate security zones:

- Two zones between the firewall and the screening routers
- One zone for the web servers
- One zone for the mail gateway and proxy server

As a result, the perimeter is well compartmentalized; if one security zone is compromised, the others remain intact. Note that if the firewall gateway is compromised, multiple security zones are also compromised. However, the interior screening router still protects the internal network.

All the components in the perimeter must be hardened to a very high level. This implies removing all unneeded or unsecure services that are provided by default. An easy thing to do is to list the active network services with a command (`netstat -an` on most operating systems), and to scan and probe the host for available services to identify which services you need and which ones you can turn off or remove.

The Windows NT/2000 Architectures

This section provides a very basic summary of the architecture of Windows NT and Windows 2000 systems. You'll need at least this background information for understanding the instructions in subsequent chapters. For more detailed information, see a good Windows operating system book. I particularly recommend *Inside Windows NT, Third Edition* by James D. Murray (O'Reilly & Associates, 1999).

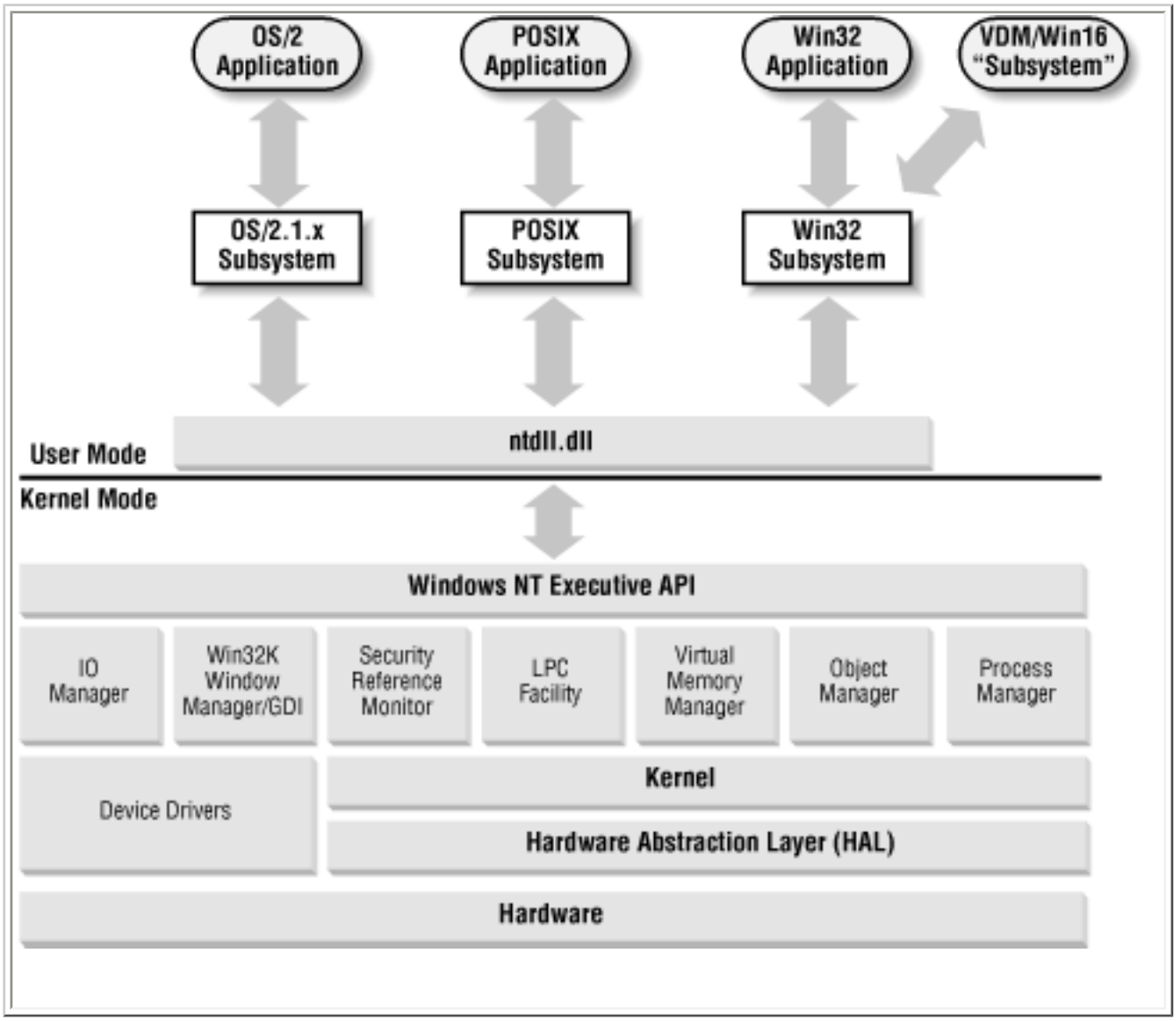
> **TIP:**
>
> In the discussion that follows, most of the details are the same for Windows NT and Windows 2000. If there is an architectural difference, I will note it.

Windows NT is a multithreaded, micro-kernel-based[8] operating system. The term *micro-kernel* implies that the kernel component is very small, and provides only basic functions such as thread dispatching and hardware exception handling. Hardware-specific code is kept in a separate layer called the *Hardware Abstraction Layer* (HAL). The HAL simplifies porting of the operating system to new processor architectures like the IA-64.

The core operating system code runs in *privileged processor mode*. This mode is also known as *protected mode* (when referring to the CPU), or *kernel mode* (when referring to a process or thread). Protected mode provides direct access to system memory and other hardware. Applications run in a nonprivileged processor mode known as *user mode* and have no direct hardware access. Applications have to use the system calls -- the API (Application Programming Interface) -- in the underlying operating system to perform tasks such as reading or writing to memory or to the screen.

The basic Windows NT architecture is shown in Figure 1-9.

Figure 1-9.The Windows NT architecture



Windows NT/2000 Subsystems and Services

Operating system services are kept in discrete subsystems, some running in user mode and others in kernel mode. There are several kernel mode subsystems in Windows NT. They provide NT's native functionality for user mode subsystems through *ntdll.dll* (running in user mode). The kernel mode subsystems make up the *Windows NT Executive*, and consist of the following*:*

Object Manager
: The Windows NT architecture is not strictly object-oriented, but internal structures such as shared memory segments, processes, and threads are represented as objects to provide a uniform method for handling things like access control. The Object Manager creates, manages, and deletes Windows NT Executive objects. Objects are represented in a hierarchical namespace much like a filesystem.

Process Manager
: Responsible for creating and terminating processes and threads using underlying kernel functions.

Virtual Memory Manager
: Implements the virtual memory used to allocate a private address space to each process.

I/O Manager
: Provides a device-independent I/O system to processes. It dispatches I/O requests to the appropriate device driver.

Local Procedure Call (LPC) Facility
: Implements a fast, lightweight version of Remote Procedure Call (RPC) for communication between components within a computer.

Security Reference Monitor (SRM)
: Enforces the access and audit policies in the system. The Security Reference Monitor provides access validation, privilege checking, and audit message generation at runtime for both user and kernel mode processes.[9]

Window Manager and Graphical Device Interface (GDI)
: These components make up the kernel mode part of the Win32 subsystem. They handle user input and screen output. All of the Win32 subsystem originally ran in user mode; however, for performance reasons, a part of it was moved to kernel mode as of NT 4.0.

The subsystems running in user mode are called the *environment subsystems*. There are three environment subsystems:

Win32 subsystem
: The part of the Win32 subsystem running in user mode. The Win32 subsystem is a required part of the operating system and is loaded as a part of the boot sequence. The subsystem consists of the Win32 API DLLs (*kernel32.dll, user32.dll, gdi32.dll*) and the Win32 subsystem process (*csrss.exe*).

POSIX subsystem
: Provides support for POSIX.1 applications. It's an optional component that is loaded on demand.

OS/2 1.x subsystem
: Provides support for OS/2 1.x console applications. It's an optional component that is loaded on demand.

TIP:

Windows NT also provides support for MS-DOS and 16-bit Windows 3.x applications through its Virtual DOS Machine. This is not a native NT subsystem. It's a Win32 application that emulates a DOS environment.

Windows NT Networking

The first version of Windows NT (Windows NT 3.1[10]) was released in 1993. It was positioned as a successor to the LAN Manager products from Microsoft and IBM. To interoperate and provide backward compatibility with these products, it had to support some established networking standards, such as NetBIOS and SMB. It's important you understand what these protocols are and how they are used in Windows NT. They still provide the foundation for most Windows NT network communication in both Windows NT 4.0 and Windows 2000.

NetBIOS

NetBIOS (Network Basic Input/Output System) is a standard for transmitting data between computers over the network. The NetBIOS specification was developed for IBM back in 1983 to allow network communication between applications. NetBIOS provides three key services:

Name service
: Locates NetBIOS names on the network.

Session service
: Provides a connection between two computers.

Datagram service
: Provides a connectionless communication channel between computers.

The first implementations of NetBIOS didn't separate the software interface from the network protocol. Later on, the network-level part of the standard was named NetBEUI (NetBIOS Extended User Interface). The Windows NT version of NetBEUI is also referred to as NBF (NetBIOS Frame). Nowadays, NetBIOS can use transports other than the nonroutable[11] NetBEUI protocol, such as TCP/IP (NetBIOS over TCP/IP--NetBT) or IPX/SPX.

Server Message Block (SMB)

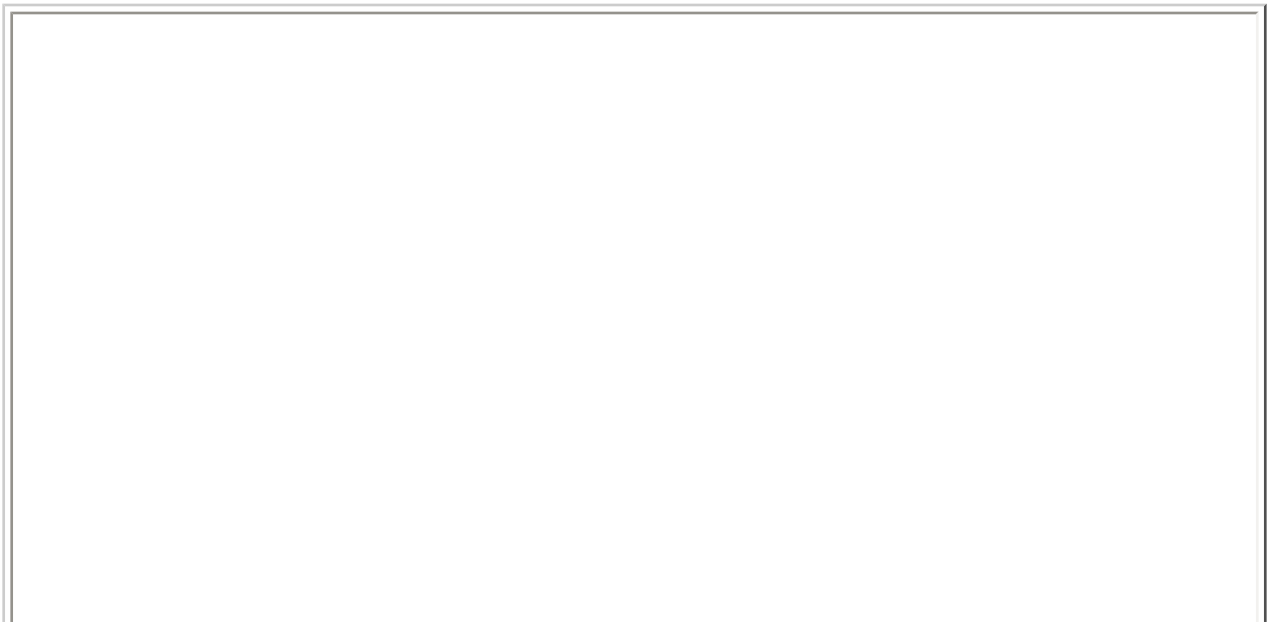
Remember that NetBIOS is merely a standard for finding resources and transmitting bits. A higher-level protocol is required on top of NetBIOS for it to be of any real use. Here's where SMB comes in. Server Message Block (SMB)[12] is a standard for sending commands and data. SMB is mostly used for file and print sharing, but it can also be used for Inter-Process Communication (IPC) to communicate with processes on other systems.

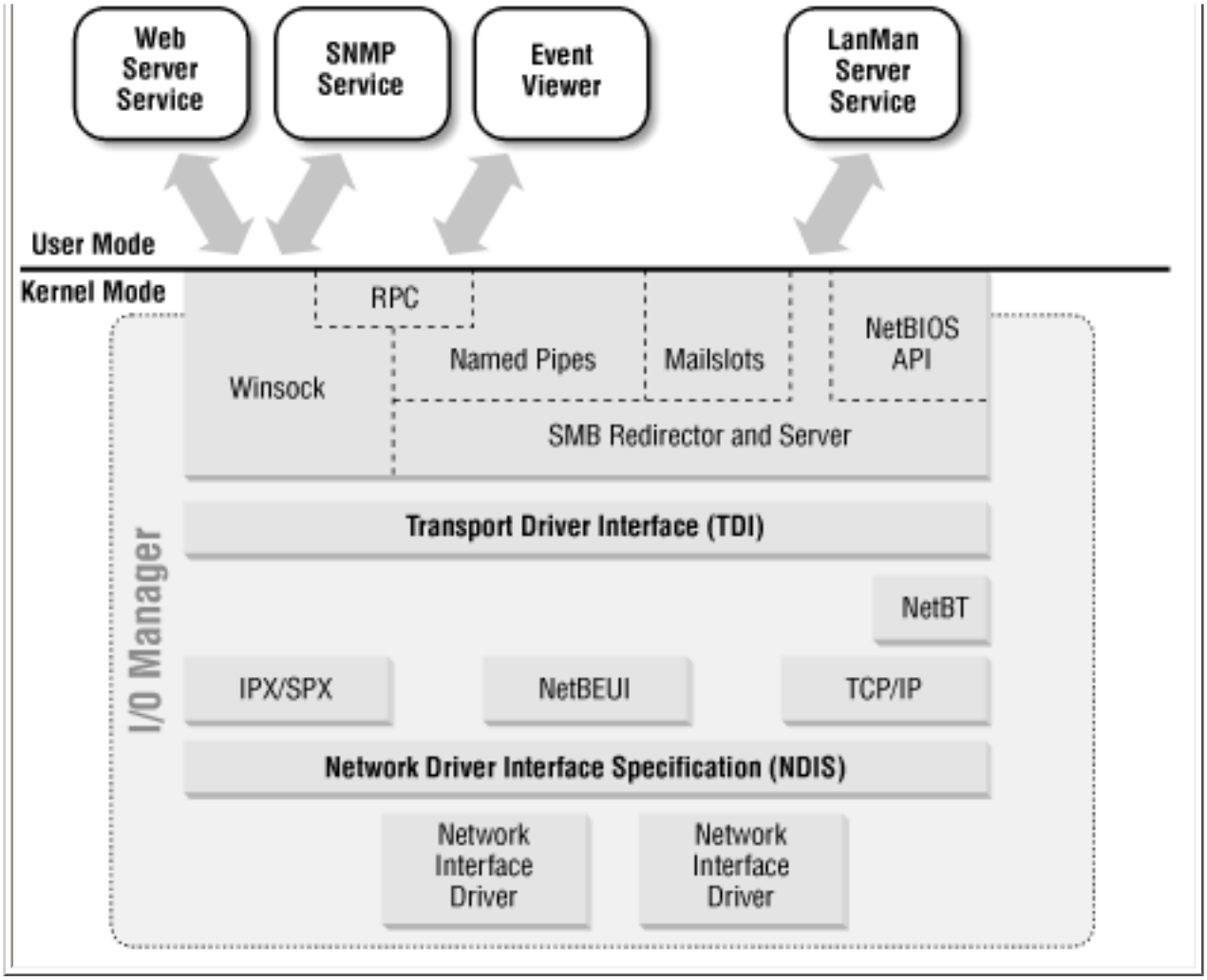
SMB over NetBIOS uses ports udp/137 (NetBIOS name service) and udp/138 (NetBIOS datagram service) or tcp/139 (NetBIOS session service). Windows 2000 includes support for running SMB without NetBIOS over tcp/445. This support is referred to as *Direct Host*.

NT networking architecture

The I/O Manager in the NT Executive is responsible for most I/O processing, including disk and network I/O. Figure 1-10 illustrates some of the networking components in the I/O Manager and shows how user mode services interact with these components.

Figure 1-10.The Windows NT networking architecture





Like all subsystems in the Executive, the I/O Manager exposes a number of APIs to user mode processes. These APIs include the following:

Windows Sockets (Winsock)
: The Windows NT implementation of the widely used Sockets API. Applications that use Winsock include Internet Explorer, IIS, Telnet, and FTP.

SMB Named Pipes
: One-way or duplex communications channels between the pipe server and one or more pipe clients.

SMB Mailslots
: A simple IPC mechanism that can be used to send or receive small (less than 425 bytes) datagram broadcast messages.

Remote Procedure Call (RPC)
: Microsoft's Remote Procedure Call (MS-RPC) provides a mechanism for using ordinary function calls to communicate with processes on another computer. Distributed Components Object Model (DCOM) components use MS-RPC.

  There are two ways of performing RPC communication between two hosts:

MS-RPC over SMB
: Uses SMB-named pipes as transport for the RPC calls. Administrative tools such as Server Manager, User Manager, Performance Monitor, and Event Viewer all use MS-RPC over SMB to connect to remote hosts. Windows NT domains also rely on MS-RPC over SMB.

MS-RPC using Windows Sockets
: Communication is established by using dynamically assigned high ports (>1023) and the RPC portmapper services tcp/135 and udp/135. This RPC method is often used by DCOM applications.[13]

SMB filesystem drivers
: There are two components that enable SMB file sharing:

SMB Redirector
: The redirector is a filesystem driver that communicates with the SMB server driver component on a remote system. The Workstation service uses the SMB redirector.

SMB Server
: The Server filesystem driver and the Server service work for the connections requested by client-side redirectors, forwarding them to the appropriate local filesystem driver, such as NTFS.

NetBIOS Interface API
: The NetBIOS interface API is provided primarily for existing applications that use IBM NetBIOS 3.0 and need to be ported to the Win32 API.

There are two boundary layers in the network architecture:

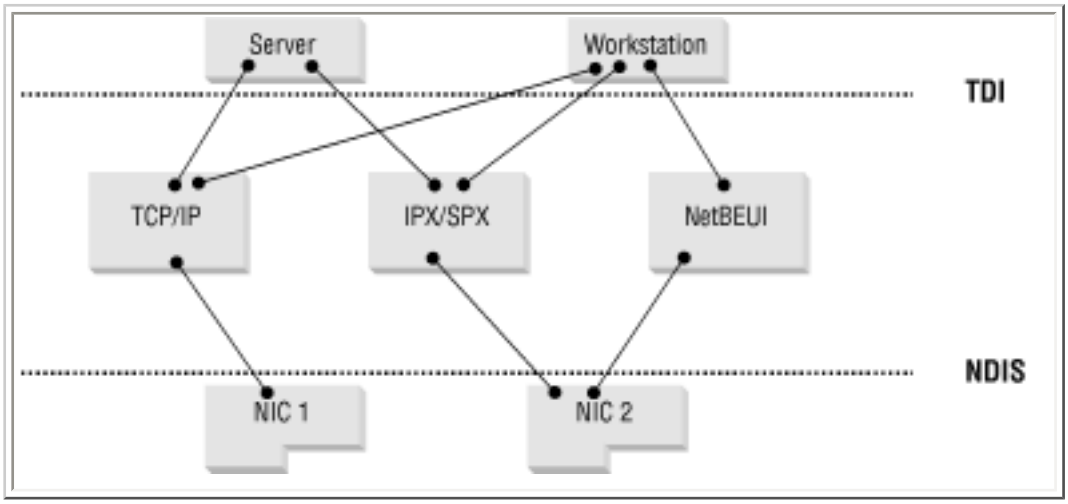
Transport Driver Interface (TDI)
: The TDI provides developers with a protocol-independent network API for network services. Developers need only to program against the TDI to support all available network protocols.

Network Driver Interface Specification (NDIS)
: The NDIS wrapper driver communicates with the network protocols. The wrapper driver provides a uniform interface between protocol drivers and NDIS device drivers.

Bindings enable communication between two components on adjacent layers in the protocol stack. For example, bindings can be configured to limit a service to one network protocol or to allow only a network protocol on one of the network adapters in the system.

In the example shown in Figure 1-11, the binding between the Server service and the NetBEUI protocol has been removed. This means the Server is not able to service requests from NetBEUI clients. TCP/IP is bound only to the NIC1 network interface card. IPX/SPX and NetBEUI are bound only to NIC2. As a result, the system will only use TCP/IP on NIC1, and both IPX/SPX and NetBEUI on NIC2.

Figure 1-11.Example of bindings in the Windows network architecture



Windows NT/2000 in the Perimeter Network

Features like discretionary access control, security auditing, and memory protection place the Windows NT core operating system on par (or better) with many Unix systems in terms of local host security. So why do many people claim that Windows is less secure than Unix?

The problem is not really Windows itself; rather it's the services and applications built on top of the operating system that are the weakest links.

The following sections describe some fundamental principles of secure system design, as well as examine how some of Windows NT/2000's services and applications stack up to these principles.

Least Privilege

A very important principle is that of *least privilege*. The least privilege philosophy dictates that an application should be designed to run only with the privilege level it needs to execute properly--and no more.

Consider the following question: what privilege level do you need to grant to a web server application? The simplified answer is that the application needs the right to read the data files it serves. Now, take a look at the Internet Information Server's (IIS)

WWW service. By design, it has to run as Local System, the highest privilege level in Windows. IIS does run the actual worker threads with lower rights, but if an attacker manages to break IIS before the security context switch is made, he'll be able to do *anything*, including deleting filesystems, starting up a back door,[14] and so on--you get the idea!

Microsoft designed IIS in this way to be able to integrate the web server with the NT security architecture. There's not much specific security code in IIS; instead, it uses the same access control mechanisms as any other NT process would. The IIS authentication mechanisms use the NT account database. Access to individual files and directories is controlled by NTFS DACLs, just like on a file server. To achieve this level of functionality, IIS needs to be able to start a process or a thread in the security context of the connecting user,[15] and to call the required Win32 APIs it needs to run at very high privilege level.

It's unfortunate[16] there's no option to install IIS without this functionality, since most Internet web servers don't need user authentication, and because of this, could run IIS at a much lower privilege level.

Unfortunately, many Windows applications and services run as Local System. Some of them may not need that privilege level, but it's the default. Most Windows software vendors don't seem to be aware of the least privilege approach, or at least they don't reflect such awareness in their code. As a result, a bug or back door in these programs can compromise the security of your entire machine. If an application is exposed in the same way it is in a perimeter network, it needs to be designed with security in mind. In fact, the top priority in the perimeter is often not functionality or speed--it's security.

Exploiting Buffer Overflow

The reality is that there are bugs in all software. It's usually not a problem if an application breaks or misbehaves under some rare circumstances--let's say 0.01% of the time. However, if you know how to force an application to misbehave or to crash, you can use this knowledge to force 100% failure. As a result, a malicious user can cause a crash (denial of service) and even execute arbitrary code on the system in the security context of the broken application. For this reason, it's very important that exposed services in the perimeter run with the least privilege possible.

A common way to break poorly written code in an application is to overwrite code segments in the stack by feeding the application long strings (URLs, for example) or other data. To keep such attacks from breaking your applications, make sure that all programs that read input always perform bounds checking (checking the length of input and trimming the strings according to the length of the input buffer). If an

application doesn't do bounds checking, it can crash. If an attacker succeeds in inserting machine-level instructions using an overflow attack, he can execute this code in the security context of the application.

The pointers of the C programming language are mainly responsible for buffer overrun problems. Some programming languages, such as Java? and Perl, prevent developers from making these types of programming errors.

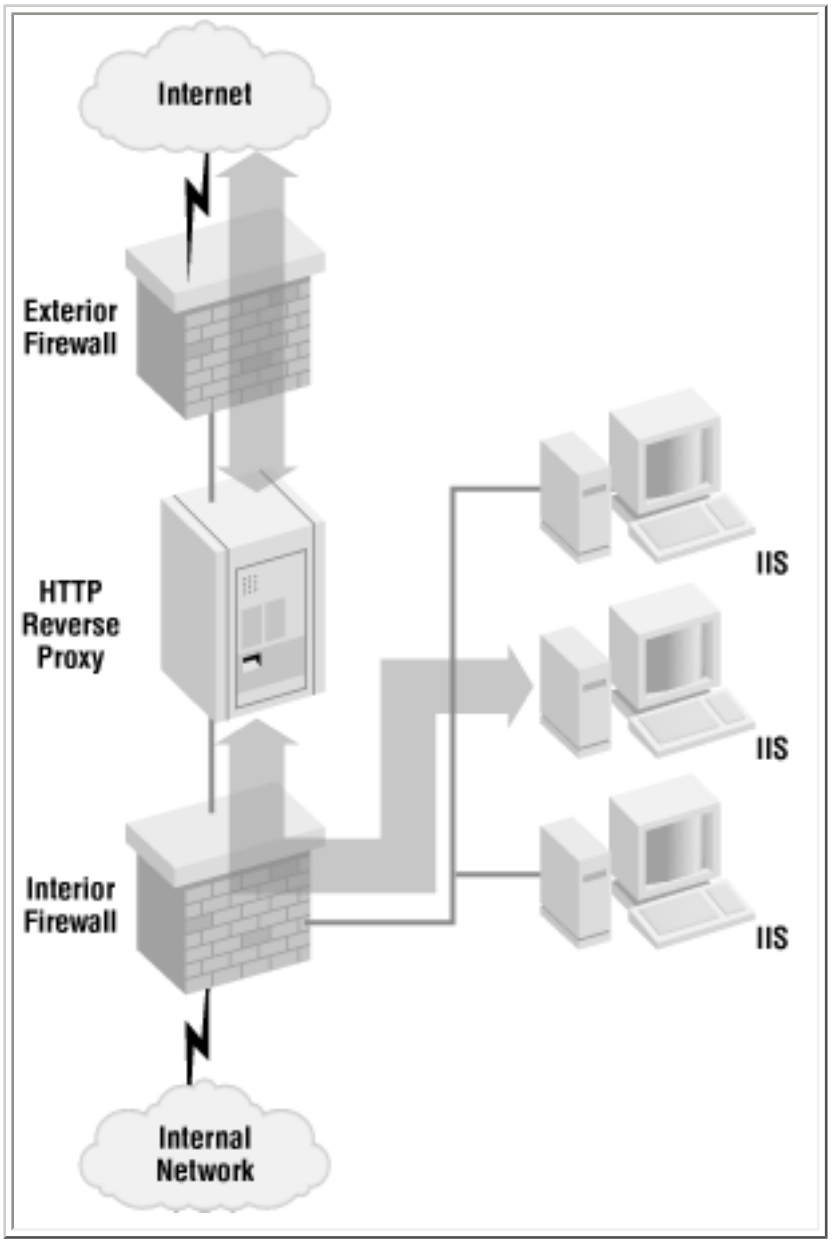
According to a recent report, buffer overflows are by far the most common type of network-based attack (because most network servers are written in C). For this reason, it's of utmost importance to keep your systems up-to-date with application and operating system patches and service packs.

"Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade," by Crispin Cowan, Perry Wagle, Calton Pu, Steve Beattie, and Jonathan Walpole, Department of Computer Science and Engineering, Oregon Graduate Institute of Science & Technology (http://immunix.org/StackGuard/discex00.pdf).

Any application that must run as Local System is potentially a major security hazard. It's a sitting duck waiting for a new buffer overflow attack (described in the sidebar later in this section) to happen. If you're running web servers like IIS, one possible solution is to place an application-level proxy in front of those servers. The proxy should be able to verify that any requests to the IIS WWW service conform to the HTTP standards. Any malformed HTTP requests will be blocked in the proxy. As a result, the web server is protected from many forms of attack. The disadvantage of having a *reverse proxy* as an additional layer of security is that it will impact performance to some extent. You also need to make sure that the proxy solution isn't a single point of failure if you want to build a highly available site.

The choice of proxy server product depends on your security needs. It may be sufficient to use an intelligent and configurable application-level (or hybrid) firewall. A better solution may be a combination of one or several dedicated firewalls and a reverse proxy server, as shown in Figure 1-12; such a configuration provides additional layers of security.

Figure 1-12.Dual firewall systems and reverse proxy solution



Separate Ports

Another important principle of secure design is to use one (or a few) fixed TCP/IP port (TCP or UDP) per application. It's good practice to use different ports for logging, viewing performance data, network logon, and so on. This separation makes it possible to implement granular network access control in the perimeter. It's often a good idea to design an application using this method.

So how does Windows behave on the network? Windows NT is terrible -- just about every service uses SMB over the NetBIOS session port (tcp/139). Windows 2000 is somewhat better. Logon authentication is Kerberos (tcp/88 and udp/88). There's also LDAP (tcp/389) to the Active Directory database. Everything else uses SMB just as in Windows NT 4.0 -- either over NetBIOS or over the new Direct Host protocol port (tcp/445).

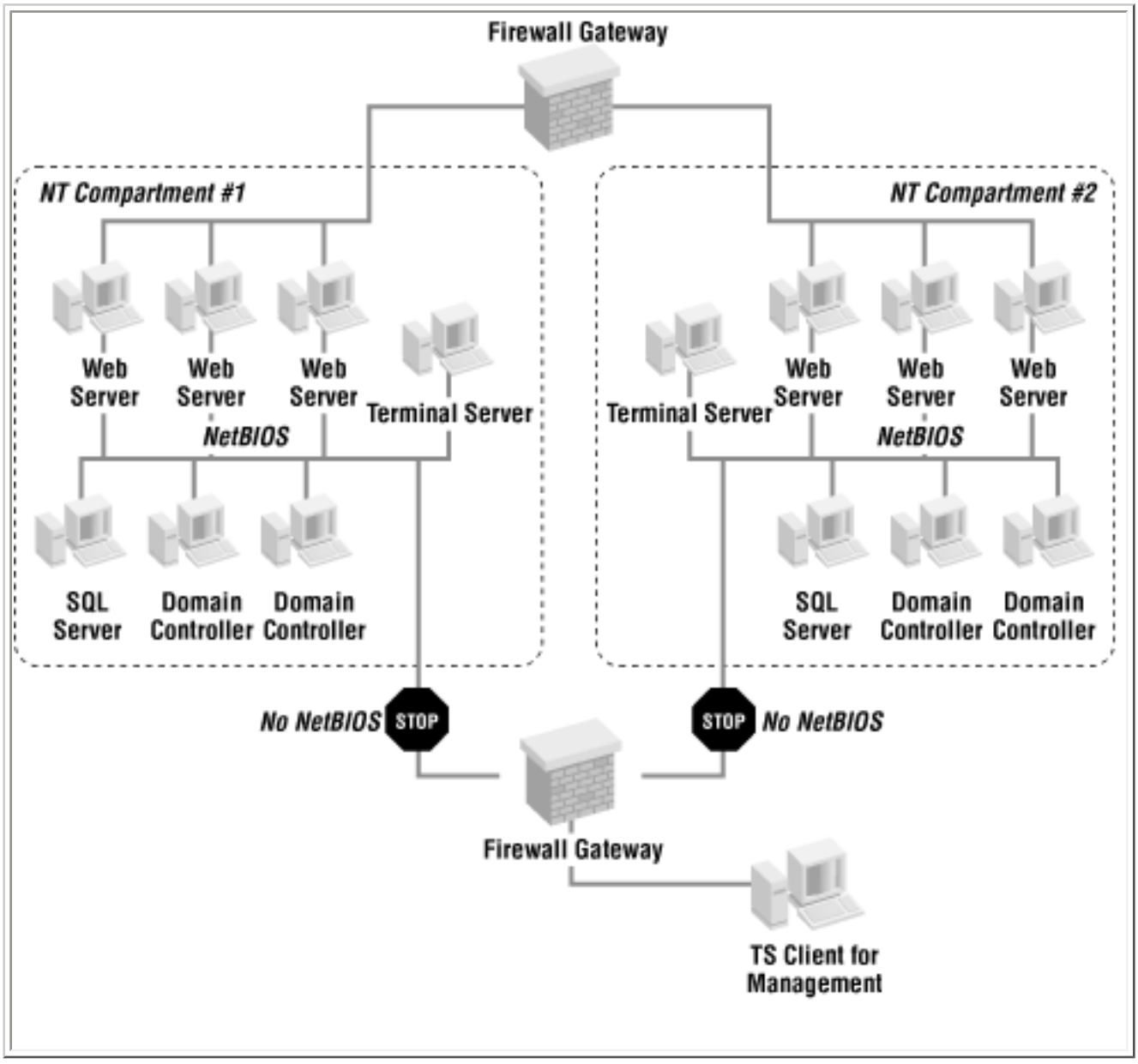
Microsoft could have done a better job here. It's virtually impossible to have Windows servers that need to communicate using NetBIOS or Direct Host in different

compartments in the perimeter.

If you plan to support a large number of Windows NT 4.0 or Windows 2000 servers, you may find that it's difficult to manage them as separate hosts. It is tricky to have both security and a management platform that scales up to hundreds of servers. It is tempting to use a centralized accounts database (using NT domains) and other NetBIOS-based tools like Event Viewer and Server Manager.

At very large sites (those with 50 or more NT servers in the perimeter), a common setup is to have dual-homed NT/2000 systems with NetBIOS/SMB unbound (deactivated) from the external network interface. The internal network interface is connected to a kind of management network so that the servers can be managed using the standard RPC-based tools in a domain environment. A remote console solution such as Terminal Server is often used to gain remote access to the management network. Figure 1-13 shows such a solution.

Figure 1-13.NT domains in the perimeter



If you must run NetBIOS in this manner, you should be aware that it will be extremely difficult to build a well-compartmentalized perimeter. You can't set up network access control to allow only a certain type of NetBIOS traffic. As a result, you have to consider all hosts within an NT domain as one security zone. In such a configuration, if one server is broken into, there are no security mechanisms that can protect your domain controllers and other servers in the same administrative domain.

NetBIOS was not designed with security in mind and I recommend *against* using it in a perimeter. However, if you choose to implement NetBIOS anyway, I urge you to implement an extremely secure perimeter using multiple firewalls and a reverse proxy solution. Do not allow any direct connections from the Internet to your exposed services.

Cryptography Basics

You will find references to various types of encryption methods and algorithms throughout this book. This section is a very brief summary of some of the terms and algorithms relevant to discussions in later chapters. For more detailed information on this complex topic, consult a good cryptography reference.[17]

Public Key Cryptography

With public key cryptography, each party has a key-pair consisting of a private key and a public key. The public key is published while the private key is kept secret. Data encrypted with the public keys can only be decrypted using the private key and vice versa. Public key cryptography can also be used for authentication (through the use of digital signatures).

An important advantage of public key cryptography is that there are less complicated key distribution problems. All parties that want to be able to communicate using public key cryptography need to publish their public key in some kind of directory. When Alice wants to send an encrypted message to Bob, she uses Bob's public key to encrypt the data. Alice can also digitally sign a message by encrypting it with her private key. Bob can then decrypt the message using his private key and verify Alice's signature by decrypting using her public key.

The main disadvantage is that public key cryptography is slow compared to symmetric key cryptography.

Two common public key systems are:

Rivest-Shamir-Adleman (RSA)
: The RSA cryptosystem is the most widely used public key cryptosystem. It's designed for doing both digital signatures and data encryption. RSA is used in

Internet standards and drafts like IPSec, S/MIME, and TLS (the successor to SSL). RSA was patented in the U.S. until September 20, 2000.

Digital Signature Algorithm (DSA)
: The Digital Signature Algorithm is an unpatented alternative to RSA. DSA was intended for performing digital signatures only, but it can be adapted for encryption as well. DSA is described in the Digital Signature Standard (DSS).[18] DSA was designed by the National Security Agency (NSA) based on the ElGamal algorithm, which is unpatented.

Symmetric Key Cryptography

With symmetric key cryptography, a single key is used for both encryption and decryption. Symmetric key cryptography is also known as secret key cryptography. The two most common methods that use symmetric key cryptography are stream ciphers and block ciphers, described in the following sections.

An important advantage of symmetric cryptography is that it's generally faster than public key cryptography. The main disadvantage is that it's hard to manage and distribute the keys to all parties in a secure manner.

Stream ciphers

Stream ciphers use a key stream that is the same length as the cleartext (unencrypted) data to produce the ciphertext (encrypted) data. The key stream can be independent of the data, or it can be generated based on it. Stream ciphers can be designed to be extremely fast. The most commonly used stream cipher is probably RC4 (Rivest Cipher 4), which is used in SSL-enabled browsers, for example.

Block ciphers

Block ciphers use a fixed-size encryption key to encrypt fixed blocks (generally 64 bits) of data. The following are some commonly used block ciphers:

Data Encryption Standard (DES)
: The U.S. Government's Data Encryption Standard is a block cipher, originally created by IBM, that operates on 64-bit blocks using a 56-bit key. This version is known as Single DES. The latest revision of DES now incorporates the Triple DES (3-DES) algorithm. This is also referred to as the TDEA (Triple Data Encryption Algorithm). 3-DES is simply Single DES performed three times (encrypt, decrypt, encrypt) with three different keys (3 × 56-bit keys) on the same 64-bit block -- hence the name 3-DES. DES is described in FIPS 46-3, available from http://csrc.nist.gov/fips/fips46-3.pdf.

Advanced Encryption Standard (AES)

The Advanced Encryption Standard, currently under development, will eventually replace DES as the U.S. government encryption standard. The AES development effort is led by the U.S. National Institute for Standards and Technology (NIST). NIST initiated the AES effort in 1997 as a "call for algorithms" in which the cryptographic community was invited to submit AES candidates. On October 2, 2000, the Rijndael (pronounced Rhine-doll) algorithm was declared the winner. It will eventually become the official AES in 2001. Follow the progress at http://www.nist.gov/aes/.

Blowfish

Bruce Schneier's Blowfish algorithm is another block cipher that operates on 64-bit blocks. Blowfish uses variable-length keys (32 to 448 bits) and offers very good performance. Blowfish is unpatented and free. You can get it from http://www.counterpane.com/blowfish.html.

International Data Encryption Algorithm (IDEA)

The International Data Encryption Algorithm is yet another block cipher that operates on 64-bit blocks. Many consider IDEA the best block cipher algorithm to date. It is used in several products and protocols such as PGP and some SSH implementations. Ascom Systec Ltd. (http://www.ascom.ch/infosec/idea.html) holds the rights to the IDEA algorithm. The licensing cost is about $10 per end user.

Hash Algorithms

Hash algorithms provide a method for reducing variable-length data to a small fixed-length *hash*. Hashes are also called *message digests* or *fingerprints*. Hash algorithms are often used to produce message integrity checksums or to store passwords in a secure manner. A hash algorithm has the following requirements:

- It should not be possible to deduce the data from the hash.
- No sets of data should produce the same hash.
- It should not be possible to generate a given hash.

Two of the more common hash algorithms are:

Message Digest (MD5)

MD5 is available from RSA Data Security. Ronald Rivest (the "R" in RSA) published the MD5 hash algorithm in 1992 as RFC 1321. MD5 was an improvement over the previously published MD4 algorithm, which has some design weaknesses. MD5 produces a 128-bit hash from an input of any length.

Secure Hash Algorithm (SHA-1)

Available from the U.S. National Institute of Standards and Technology, SHA-1 is the NIST Secure Hash Standard.[19] It's considered to be the most secure hash algorithm today. It produces a 160-bit hash from a variable-length input.

---

1. This attack was described in detail on the BugTraq mailing list on May 4, 2000, in a message entitled "How we defaced www.apache.org."

2. PHP is a free server-side scripting language (http://www.php.net/) similar to Microsoft's Active Server Pages (ASP).

3. Tom O'Donnell of the IEEE Ethics Committee describes gray-hats as "self-styled Robin Hoods who make it their business to expose security flaws in software in a very public way" (http://www.spectrum.ieee.org/INST/dec99/ethics.html).

4. CERT-CC originally was established in response to the first major Internet security incident: the release of the Internet worm back in 1988.

5. *Practical Unix and Internet Security, Second Edition*, by Simson Garfinkel and Gene Spafford (O'Reilly, 1996) has an excellent chapter on physical security. You might consider checking it out even if you don't need the Unix details.

6. Available at http://pubweb.nfr.net/~mjr/pubs/think/index.htm.

7. Non-first TCP fragments are permitted. See RFC 1858, "Security Considerations for IP Fragment Filtering," for details.

8. The NT architecture is not a true micro-kernel architecture. Most true micro-kernel implementations have exhibited poor performance. Many compromises in the micro-kernel design have been made in the NT architecture to achieve better performance.

9. SRM doesn't check for object permissions if the calling thread or process is running in kernel mode. For performance reasons, SRM assumes that the caller has permission on all objects, since it is running in kernel mode and is therefore a part of the Trusted Computing Base (TCB).

10. It was dubbed Version 3.1 mainly because it reported version number 3.1 for backward compatibility with Windows 3.1 applications.

11. All hosts have to be on the same network segment to be able to communicate using NetBEUI.

12. Microsoft has renamed its SMB protocol implementation "CIFS" (Common Internet File System) in a marketing effort to make it an "open" protocol.

13. You can configure DCOM to use a specific port range on a per-application basis using the DCOM Configuration Properties application (*dcomcnfg.exe*).

14. An example of a back door is a shell process (like *cmd.exe*) that runs with Local System privilege and that can be accessed over the network.

15. Unauthenticated "anonymous" connection threads run as the IUSR_MACHINENAME account.

16. If you consider the security implications, you'll realize that an experienced Unix system administrator would never run a web server as root. It's just not a very bright thing to do!

17. I particularly recommend Bruce Schneier's *Applied Cryptography, Second Edition* (John Wiley & Sons, 1996). You can also find a very readable summary of fundamental cryptography terms and algorithms in Appendix C of *Building Internet Firewalls, Second Edition*, referenced earlier in this book. There's a good online FAQ at RSA Labs as well (http://www.rsasecurity.com/rsalabs/faq/).

18. Digital Signature Standard, Federal Information Processing Standard (FIPS) 186-2 (http://csrc.nist.gov/fips/fips186-2.pdf).

19. The Secure Hash Standard, FIPS 180-1 (http://csrc.nist.gov/fips/fip180-1.pdf).

Back to: Securing Windows NT/2000 Servers for the Internet

© 2000, O'Reilly & Associates, Inc.
webmaster@oreilly.com