Chapter 13

Windows 2000 Server Security

In This Chapter

- Implementing IPsec (Internet Protocol security)
- Discovering Kerberos V5
- Learning about Smart card support
- Implementing Encrypted File System (EFS)

If you feel you already know everything you need to know about Windows 2000 Server security, please proceed to the next chapter. If not, welcome to Chapter 13. Perhaps you've always assumed your security needs ended after you physically secured the server, forced users to log on with passwords, and applied sharing and NTFS-level permissions (see Figure 13-1). Think again. In the four years since NT 4 was released, security has become a major concern for corporations and individual users deploying Microsoft networking technology. From network attacks to disgruntled employees, network systems (including Microsoft’s Windows 2000 Server) have come under continual assault:

- Password-cracking tools are readily available for downloading on the Internet.
- The cost of network monitoring (sniffing) equipment has declined considerably.
- Web sites from Microsoft to the White House have undergone service interruptions and altered content from outside attacks.
- A 21-year-old hacker obtained over 1,700 user IDs and passwords via a sniffer running inside TRW Credit in the mid-90s.
- A United States Senate subcommittee found that as of the mid-1990s, 58% of major US corporations have had break-ins; 18% suffered losses greater than $1 million, and 20% of break-ins were linked to competitors.
Of course, to a great degree, security is a function of common sense. And just like the sport of sailing, which is also based largely on common sense, many of us need to sharpen our common sense skills, right? How many of your users store their passwords in their top desk drawer? Are you checking logs to see who is RASing in at night? Can your firewall alert you to potential attacks in progress? Akin to sailing, this security-oriented behavior is similar to completing the boat’s logbook.

Although the integrity of any system is primarily the result of intelligent planning and vigilant management, some technological advances that can help you improve security have been included in Windows 2000 Server. The security paradigm shift that has occurred between Windows NT Server 4.0 and Windows 2000 Server is two-fold: Internet and Enterprise. To make that point abundantly clear, observe Figure 13-2, in which the traditional network security model is upgraded for Windows 2000 Server.

In this chapter, I will deal with four additions to Windows 2000 Server that improve security: IPsec (Internet Protocol security), Kerberos V5, Smart Card support, and Encrypted File System (EFS). Note these are high-level security issues. In other words, these are security issues that are typically addressed at the 50,000-foot-deep level of strategic planning. Back at sea level, your day-to-day security issues are discussed in Chapters 9 and 10.
Network security is a major issue these days. Some of the most dreaded attacks occur at the physical level, where sniffers and network monitors are capable of capturing and interpreting network packets. Note that I discuss Network Monitor, warts and all, in Chapter 19. Often these intrusions come from within an organization. With the tools currently available, data can be viewed, copied, or modified without a trace.

**IPsec**

It goes without saying, but I'll say it anyway. Network security is a major issue these days. Some of the most dreaded attacks occur at the physical level, where sniffers and network monitors are capable of capturing and interpreting network packets. Note that I discuss Network Monitor, warts and all, in Chapter 19. Often these intrusions come from within an organization. With the tools currently available, data can be viewed, copied, or modified without a trace.
As you will discover in a moment, IPsec will indeed protect your network traffic from being viewed with Network Monitor. The bad news is that it renders Network Monitor useless in trying to resolve network traffic problems involving IPsec. Ouch!

At a very fundamental level, one of the major changes in Windows 2000 Server is the incorporation of Internet Protocol Security, or IPsec. IPsec is the result of an effort by the Internet Engineering Task Force (IETF) to provide network-level authentication, data security, and encryption. Because it is implemented below the transport level, no modifications to existing applications are necessary. And because it is the product of an industry-wide consortium, interoperability with other computing platforms is assured.

The major benefits of IPsec are as follows:

- **Authentication.** Prevents the interception of data via impersonation.
- **Confidentiality.** Prevents unauthorized access to sensitive data when required. This is accomplished by encryption of IP traffic per packet.
- **Data integrity.** Ensures the use of IP authentication headers (see Figure 13-3) and variations of hash message authentication code. Also known as integrity and source authentication per packet.
- **Dynamic rekeying.** By changing the key dynamically, Dynamic rekeying during ongoing communications helps protect against attacks.
- **Transport mode.** Secures links end to end — within the same domain or across any trusted path (see Figure 13-4). Windows 2000 Server L2TP tunnels use IPsec transport mode.
- **Tunnel mode.** Router to Router. Basically, IPsec creates a secure tunnel between two IPsec-compliant hosts (see Figure 13-4). Windows 2000 Server supports IPsec tunnels.
- **Centralized management.** Uses security policies and filters to provide appropriate levels of security, based on user and work group. Filtering is also available per address, subnet, protocol, and port.
- **Flexibility.** Policies can be applied to the entire enterprise or a single workstation.

IPsec uses two components: an authentication header (AH) to provide source authentication and integrity, and an encapsulated security payload (ESP) to provide confidentiality. By using both public keys and shared secret keys, IPsec provides a high degree of security to data communications, both within and between organizations.

One example of IPsec's defense against common attacks is how the sequence number is handled. You may recall from the Windows NT Server Enterprise MCSE course, where you were probably introduced to Network Monitor for the first time, that sequence numbers are very important in reconstructing the order of arriving packets by the receiving hosts. It's as if packets had sequence numbers of 1, 2, 3, 4, and 5 and maybe arrived out of order as 2, 3, 4, 1, and 5. The receiving host uses the sequence number to put the packets back in order as 1, 2, 3, 4, and 5. IPsec encrypts the sequence numbers via cryptohash to foil would-be hackers.
Chapter 13: Windows 2000 Server Security

Figure 13-3: IPsec header

This is added to the packet with IPsec in Windows 2000 server.

TCP/IP Packet

<table>
<thead>
<tr>
<th>Original IP Header</th>
<th>IPsec Header</th>
<th>TCP Header</th>
<th>Data</th>
</tr>
</thead>
</table>

24 Bytes

Figure 13-4: IPsec end-to-end versus tunnel
Implementing IPsec is relatively easy, and I’ve found that just doing it is one of the best ways to learn it. Start with the IP Security Policies MMC (see Figure 13-5). Here you may select from three model policies: Server (Request Security), Secure Server (Require Security), and Client (Respond Only). Double-click one of the policies, say Secure Server (Require Security), to display the properties sheet (see Figure 13-6).

![Figure 13-5: IP Security Policies MMC](image)

You may modify the default policy settings by clicking the Add button to add a new IP Security Rule. Likewise, Edit would enable you to edit an existing IP Security Rule. If you elect to add a new IP Security Rule by clicking the Add button, the Security Rule Wizard will launch (see Figure 13-7).
Figure 13-6: Secure Server (Require Security) properties

Figure 13-7: Security Rule Wizard
STEPS:
To configure Security Wizard

Step 1. Configure the tunnel endpoint (see Figure 13-8) by IP address or DNS name. Click Next.

Step 2. Define a network type (see Figure 13-9) as either local or remote. Click Next.

Step 3. Select an authentication method (see Figure 13-10). Click Next.

Step 4. Create an IP filter list, as shown in Figure 13-11. Click Next.

Step 5. Make a filter action election (see Figure 13-12). Click Next. You will click Finish to close the Security Rule Wizard.

Step 6. Finally, back at the main IP Security Policies MMC screen, right-click the IP security policy you just modified and select Assign from the secondary menu (see Figure 13-13). You have now implemented IPsec via a security policy. Congratulations.

Figure 13-8: Tunnel endpoint
Figure 13-9: Network type

Figure 13-10: Authentication method
Figure 13-11: IP filter list

Figure 13-12: Filter action
Be careful now that you've created an IP security policy. If implemented, it will really work at this point. You might be surprised that you can accomplish nothing more than you've allowed. In short, you might find yourself uttering, "I think I've created a monster." Beware.

Kerberos V5

Ah, you and I have finally reached Windows 2000 Server security Mecca. Time to take in the Zen of Kerberos! It's a key (get it?) component of the Windows 2000 Server security model. That's because the root of the security authentication for Windows 2000 Server has moved from the LAN Manager model (NTLM) to the Kerberos model. And while Kerberos is receiving a lot of press with respect to the Windows 2000 Server security architecture, I think you'll be surprised to see that it really functions in the background.

What is Kerberos?

First, let's answer the question: What was Kerberos? This makes a wonderful technical trivia question. In Greek mythology, Kerberos (or Cerberus) was the three-headed dog that guarded the gates of Hades, or Hell. What better example for a mechanism that provides access security for your resources?

Kerberos was developed as part of Project Athena (more Greek mythology) at the Massachusetts Institute of Technology in the 1980s. It is primarily a
method of verifying that a user is who they say they are. For example, we all know how easy it is to send e-mail purporting to be from someone else. Similarly, in a network situation, how can an application be sure that the request is coming from a user with valid access rights and not an imposter? Kerberos provides this authentication mechanism. It is also bilateral; not only can the application be assured of the identity of the user, the user can be sure that the application being accessed is the authentic one.

As the mythical Kerberos had three heads, the Kerberos security protocol uses three entities — the user, the application or server, and a “trusted third party.” In the case of Windows 2000 Server, the third party is the Key Distribution Center (KDC) using information contained in the Active Directory.

Let’s walk through an example that illustrates how Kerberos functions. As kids, Samantha and Molly used to pass notes back and forth in school. After they were embarrassed by having a note read aloud in class by the teacher, they devised a simple code. As they were the only ones who knew what the key to the code was, their messages were relatively secure — by applying the shared key to decode the message, Molly could be sure that the message actually came from Samantha. Later, however, they wanted to include Cassandra in their note-passing scheme. They had two choices — either each one of them could share a key with one of the others and a different key with the other one, or they could all use the same shared private key. As the circle of friends widened, these choices had obvious limitations: either too many keys to keep track of, or the likelihood that one of them would expose the single key.

Now let’s transpose this situation to a network. Samantha (a client) can’t meet with Molly (a server) in a secure area to tell Molly what the key is — anyone with a sniffer on the network would discover the key. What the network needs is a trusted repository of keys, a secure system that shares keys with each entity on the network that needs authentication. In very simple terms, this is how it works: when a user requests to communicate with a server, the trusted repository, or Key Distribution Center (KDC), sends back to the client two data structures: a session key, encrypted with the client’s key; and a session ticket, which contains the server’s session key and information about the client. The session ticket is encrypted with the server’s key. When the client requests information from the server, it sends the session ticket (still encrypted with the server’s key) to the server, along with an authenticator encrypted with the session key. The server decrypts the ticket, extracts the session key, decrypts the authenticator, matches it with the client information contained in the ticket, and proceeds with the transaction, assured that the client really is who they say they are. If the client has requested authentication of the server, the server then uses the session key to encrypt a part of the client’s authenticator and returns it to the client, who then can trust the server.

Session tickets have a defined lifetime, typically around eight hours, and are thus reusable. This means that the KDC does not have to be contacted for every interaction between the client and the server as long as the session ticket is still valid. As the session keys are only kept in volatile memory on
the client and the server, not on disk, the credentials memory cache is flushed and no record exists of the session when the user logs off.

When the user logs first logs on, the Kerberos client on the user's machine encrypts the password entered by the user and sends it on to the KDC, along with the user name. This encrypted password is referred to as the user's long-term key. The KDC looks up the user name in its database, compares the encrypted password it received with the long-term key, and creates a session key/session ticket as described above. This ticket, because it is used to communicate between the user and the KDC, is referred to as a ticket-granting ticket, or TGT. Likewise, the session key between the user and the KDC is called a logon session key, valid for as long as the user is logged on.

So, to clarify the interaction described above, when a client wants to connect to a server, it searches its cache for a session ticket for that server. If it finds one, it communicates directly with the server. If there is none, the client uses the logon session key and TGT to request a session ticket from the KDC.

Reasons for the move

Kerberos Version 5 is now the default authentication protocol for Windows 2000 Server. The previous default, Windows NT LAN Manager protocol (NTLM), will continue to be used to authenticate NT 4.0 clients.

There were five primary reasons why Kerberos was selected as the authentication protocol:

- **Network efficiency:** Because each client-server or server-server transaction does not need to be authenticated by a domain controller, network bandwidth is conserved.

- **Bilateral authentication:** With NTLM, the server could be sure of the client's identity, but the server was never authenticated to the client. With Kerberos, if the client requests, the server can be easily authenticated.

- **Three-tier authentication:** In both NTLM and Kerberos, the client is impersonated on the server to determine resource access rights. In a three-tier architecture, NTLM had no way of letting the intermediate server authenticate the client to the other server. Kerberos uses a proxy mechanism that allows servers to impersonate a client on another server.

- **Simplified trust relationships:** By default, trust relationships under Kerberos are bilateral and transitive. This means that once credentials for each security authority in an organization are mutually authenticated, two-way trust relationships automatically exist. If Domain A trusts Domain B, Domain B automatically trusts Domain A. In addition, if Domain B trusts Domain C, the transitive nature of Kerberos authentication means that Domain A also trusts Domain C, and vice versa.
Interoperability: As Kerberos Version 5 is a standard incorporated into other operating systems, it is conceivable to have trust relationships between Windows 2000 Server domains and Unix Kerberos realms, for example. Also, individual users with Kerberos clients on other operating systems can be validated and mapped to Windows 2000 Server domain accounts.

Want another take on Kerberos? Take a moment to adhere to the old maxim that a picture is worth a thousand words (or at least a run-on bulleted list). In Figure 13-14, the Kerberos logon flow is displayed as an example of how Kerberos “fits” in the Windows 2000 Server security model.

How is it implemented in Windows 2000 Server?

In Windows 2000 Server, the KDC is implemented as a domain service, using the Active Directory as its account database. There is a KDC located on every domain controller, along with the Active Directory. Both services start automatically and cannot be stopped. Each runs in the process space of the Local Security Authority (LSA). Any domain controller can authenticate and issue tickets.

Policies for Kerberos implementation are determined at the domain level and contained in the Active Directory as part of the domain security policy. Settings for Kerberos policy include:
Maximum user ticket lifetime. This setting determines the life, in hours, of the TGT.

Maximum lifetime that a user ticket can be renewed, in days.

Maximum service ticket lifetime. This setting determines the life of a session ticket, in minutes, between 10 and the value of Maximum user ticket lifetime.

Maximum tolerance for synchronization of computer clocks. As timestamps are part of session keys and session tickets, there must be some accommodation for variations in clocks on the various systems. Settings are in minutes.

Enforced user logon restrictions. This setting determines whether the KDC will validate each request for a session ticket by examining whether the user has the right either to Log On Locally or Access This Computer from the Network. This is an option because the extra lookup involved can slow down the network.

In all likelihood, you are going to bump into Kerberos while undertaking some security task that uses it. For example, in the IP Security Policy Wizard, the Authentication Method screen allows you to select the Kerberos V5 protocol (see Figure 13-15). This type of encounter is most likely going to create your best memories of Kerberos in Windows 2000 Server.

Kerberos extensions in Windows 2000 Server

As I mentioned, Kerberos is a shared secret technology. Many of the other advances in cryptography are based upon public key implementations, where the certifying authority (CA) is an external third party. And within Windows 2000 Server, you have already seen Kerberos in action under the guise of IPsec.

Kerberos also rears its head in other ways within Windows 2000 Server. In particular, smart cards use public key encryption for authentication. In order to support smart cards for logon access, some interaction between public keys and Kerberos is required. Microsoft, in conjunction with IETF, has developed extensions to the Kerberos protocol to substitute the public/private key pair on a smart card for the shared secret key derived from the user’s password. The KDC encrypts the logon session key using the user’s public key; the client uses the private key to decrypt the logon session key.
Smart Card Support

When you play Monopoly, what is the most valuable card you can get? The Get Out of Jail Free card, of course! Well, in the real world, there is another valuable card you can have — a Get Onto the Network Easily card, or smart card. Smart cards have been developed for all sorts of purposes, but in the context of this chapter, smart cards are used to authenticate the identity of the owner to log on.

Smart cards have a number of advantages:

- Making the storage medium for private keys and other personal information tamper-resistant
- Offloading security computations from the rest of the system
- Providing a portable authentication mechanism that insures accurate logon information for many systems

Smart cards use microchip technology to hold the information needed for public key encryption. Earlier in this chapter, I noted that Windows 2000 Server uses the Kerberos protocol, based on shared secret keys, to authenticate users. In order to bridge the differences between public key and shared secret key technologies, Microsoft has, in conjunction with the IETF, developed extensions for Kerberos which enable the logon process to use the
public/private key information on the smart card as a substitute for the shared secret key information derived from the user’s password. The smart card architecture in Windows 2000 Server is shown in Figure 13-16. Note that Server Cache Synchronization Protocol (SCSP) refers to a general approach to solving cache synchronization/cache replication problems in distributed protocol entities. Crypto API/SSCP refers to an encryption application program interface, and common dialog refers to the smart card’s interface.

![Diagram of Smart Card Compliant Applications](image)

**Figure 13-16**: Smart card architecture

Smart cards can also be used for client authorizations using the Secure Sockets Layer (SSL). This is an application level authorization, requiring the holder of the smart card to identify itself before processing. This technology using SSL provides for mutual authentication, meaning that the user is authenticated to the application as well as the application being authenticated to the user.

Finally, smart cards are also useful for remote access. Using third party authentication modules, smart cards can be used to provide credentials for authentication via RAS.

If you would like more information on smart cards, go to the [www.smartcardsys.com](http://www.smartcardsys.com) Web site.
EFS Encryption

One of the security features touted when Windows NT was first launched in 1993 was a new file system, the NT File System or NTFS, which would provide a high degree of security for disk-based data. NTFS uses access control lists for all objects, meaning that permissions could be granted to individual users down to the file level. As the file system could be accessed only from NT, not from DOS, it was thought to be a very secure method to store data.

The Titanic was thought to be unsinkable, too. Microsoft set itself up again to the target of attacks against whatever it said couldn't be attacked. Soon, tools that enabled a user to bypass intricate NTFS security when an NT system was booted using a DOS diskette, became readily available to the public. Files were thus laid wide open for an attack.

Previously, you could bypass local NTFS security, at least before EFS, by simply installing Windows NT Server 4.0 a second time on the same NTFS partition and booting from it. In that scenario, new Security Accounts Manager (SAM) in hand, you could browse the NTFS partition to your heart's delight! Historically, this was a common workaround, known as a parallel installation, for recovering your underlying Windows NT Server operating system when something went astray big time (such as a failed upgrade from a single to multiple processor using `uptomp.exe`). EFS addresses the parallel installation weakness.

But don't forget, war stories such as those presented here suggest you should always have physical security in place for your server, first and foremost, thus preventing unauthorized parallel installations to begin with! Easy to do when the physical security issue applies to a locked-up server. But what about the traveling laptop? There goes the basic physical security premise, eh?

So, given this immediate discussion regarding NTFS security weaknesses, the answer is to encrypt the file. This has been possible in the past with third-party utilities, but there are several drawbacks: the encryption/decryption is manual, not automatic; temporary files may not be encrypted; and the encryption is usually password-based, which third-party password crackers can decrypt.

In Windows 2000 Server, the Encrypting File System (EFS) is available as an integrated file system. EFS can use any symmetric encrypting algorithm, although only DESX is supported in the first release.

EFS generates a public/private key pair and registers it with a certificate authority (CA) if one is configured, or handles the registration itself if no CA is available. Individual files or entire directories can be encrypted, both locally and on remote volumes. Encryption and decryption are automatic and transparent to the user as the bytes are read from or written to disk.

EFS also supports data recovery using a master recovery key. This is helpful, for example, in examining a former employee's files or when the specific encryption keys are lost.
It is easy to use EFS in Windows 2000 Server. EFS is turned on or off for files or directories using the Advanced button on the General tab of the properties for that entity (see Figure 13-17). After making the encryption selection, you will receive an encryption warning (see Figure 13-18) that warns you if you are encrypting a file in an unencrypted folder. You are offered the opportunity to encrypt the folder at this time.

Figure 13-17: File encryption settings

Figure 13-18: Encryption warning
Alternatively, a command line utility, **cipher**, can be used to encrypt or decrypt files or directories. The cipher command contains the following options.

Displays or alters the encryption of directories [files] on NTFS partitions.

```
```

- **/E** Encrypts the specified directories. Directories will be marked so that files added afterward will be encrypted.
- **/D** Decrypts the specified directories. Directories will be marked so that files added afterward will not be encrypted.
- **/S** Performs the specified operation on directories in the given directory and all subdirectories.
- **/I** Continues performing the specified operation even after errors have occurred. By default, CIPHER stops when an error is encountered.
- **/F** Forces the encryption operation on all specified directories, even those which are already encrypted. Already-encrypted directories are skipped by default.
- **/Q** Reports only the most essential information.
- **dirname** Specifies a pattern, or directory.

Used without parameters, CIPHER displays the encryption state of the current directory and any files it contains. You may use multiple directory names and wildcards. You must put spaces between multiple parameters.

**Summary**

It's one thing to revisit the traditional network security model and then learn about the significant security enhancements (such as IPsec) in Windows 2000 Server. But as final parting wisdom on network security, it's often as important to recruit honest people and educate them well on security matters. In short, it's both the security technology and the soft security skills, such as training, that make for a fully secured Windows 2000 Server network.

- Reasons to use IPsec (Internet Protocol security)
- Steps to implementing IPsec
- Defining Kerberos
- Smart card support in Windows 2000 Server
- Role of Encrypting File System (EFS) in Windows 2000 Server