The second installment of your step-by-step guide to setting up and configuring IIS 6.0, as well as understanding each setting, service and component of the program.

IT Influencer Series

IIS 6.0: Step-by-Step Mega-Guide, Part II

by Don Jones

Contents

- FTP Configuration .................................................. 4
- SMTP and NNTP ..................................................... 17
- POP3 Server .......................................................... 47
- The .NET Framework ................................................. 57
- Automated Installation .............................................. 74
- Locking Down the Server ........................................... 84
Need to Secure Multiple Domain or Host Names?

Securing multiple domain or host names need not burden you with unwanted administrative hassles. thawte’s cost-effective Starter PKI Program can streamline management of your SSL and Code Signing certificates putting you in control.

Main program benefits include:

- Fast certificate issuance across multiple domains
- Complete certificate life-cycle management
- Discounted certificate pricing
- Dedicated account management

Get your free guide, “The Starter PKI Program”; to learn more about how this program can benefit you – click here.

More information is available at: www.thawte.com/spki/

thawte SPKI Team

Tel: +27 21 937 8977

E-Mail: spkisales@thawte.com
Table of Contents

Chapter 1: FTP Configuration ................................................................. 4
Installation .......................................................................................... 4
Configuration ..................................................................................... 5
Security ............................................................................................. 15
Summary ........................................................................................... 16

Chapter 2: SMTP and NNTP Configuration ......................................... 17
Installation ......................................................................................... 17
Configuring SMTP ............................................................................. 18
Configuring NNTP ............................................................................ 34
NNTP Virtual Server ......................................................................... 35
Newsgroups ....................................................................................... 40
Expiration Policies ........................................................................... 43
Summary ........................................................................................... 46

Chapter 3: POP3 Server .................................................................... 47
Installation ......................................................................................... 47
Configuration ................................................................................... 48
Domains and Mailboxes ................................................................. 51
Configuring SMTP Support ............................................................ 54
Summary ........................................................................................... 56

Chapter 4: The .NET Framework ...................................................... 57
Chapter 1: FTP Configuration

In addition to being a great Web server, IIS can function as a passable FTP server. It doesn’t include all the bells and whistles of some commercial (and some free) FTP server products, but it does provide reliable, basic FTP functionality that can be made reasonably secure. In this chapter, I’ll explore the options for using FTP and show you how to configure the available settings.

Installation

Remember that FTP is, like just about everything else, an optional component of IIS. You’ll need to install it before you can create FTP sites. To do so:

1. Open Add or Remove Programs from the Control Panel.

2. Click Add/Remove Windows components.

3. Select Application Server and click Details.

4. Select Internet Information Services and click Details. You’ll see the dialog box shown in Figure 8-1, where you’ll select the FTP checkbox.
5. Click OK to close each dialog box and install the FTP component.

You’re ready to start configuring FTP. Once installed, you immediately have one active FTP site that’s set up to use the default configuration. You’ll want to review that configuration to ensure it at least meets your organization’s security requirements.

**Configuration**

As shown in Figure 8-2, you’ll use the IIS Manager console to administer your FTP sites. Just like Web sites, you can have multiple FTP sites running at once. Each is referred to as an FTP virtual server, or simply an FTP site. IIS creates a Default FTP Site for you when FTP is installed.
To modify the properties for the site, simply right-click it and select Properties from the context menu. You’ll start in the dialog box shown in Figure 8-3.
Here, you can configure the site’s description, as well as configuring the IP address and TCP port it uses. The IP address and port combination must be unique; if you have multiple FTP sites on the server, they must all have a unique combination of IP address and port number. This unique combination allows IIS to figure out which FTP site incoming requests are meant for. It’s similar to the way IIS identifies Web sites (which I described in Chapter 5), except that FTP doesn’t offer the additional flexibility of a host header.

You can also configure the maximum number of connections the FTP server will accept and the number of seconds FTP will wait before disconnecting an idle connection. You can also enable or disable logging of FTP activity and select the format of the log. These settings are all similar to those for a Web site.
On the Security Accounts tab, shown in Figure 8-4, you’ll configure the security account FTP uses.

Figure 8-4.

Notice that, by default, anonymous connections to FTP are allowed. Those anonymous users will have all the access privileges and capabilities of the IUSR_machinename account, as shown here. You can change that account to provide different access permissions for anonymous users. You can also configure the Web site to only allow anonymous connections. Doing so will prevent users from logging in by using a local or domain user account’s credentials. Why might you allow only anonymous connections? If you have a Web site used by customers to download product updates, for example. Because FTP transmits user credentials in clear text, forcing anonymous connections will prevent any company employees from trying to
use their company login credentials, which might be compromised by the lack of encryption.

On the Messages tab, shown in Figure 8-5, you can configure the text messages displayed by the FTP server. These can be anything you like; however, note that most users will only see them if they’re using a command-line FTP client like the one included with Windows. Graphical clients like the popular CuteFTP don’t display these FTP text messages in a location where they’re readily apparent to users.

![Figure 8-5. Default FTP Site Properties](image)

The Home Directory tab, shown in Figure 8-6, allows you to configure the properties of the FTP site’s root folder.
As with a Web site, this folder can be local or a shared folder on another computer. You can also configure the folder to allow users to read or write files, although any underlying share or NTFS permissions will also affect those capabilities. Finally, you can configure the directory listing style. The default is MS-DOS, which works for most situations. Use the UNIX style if you have older clients using UNIX-based FTP software, and you discover that they’re unable to display directory lists from their computers.

Finally, the Directory Security tab, shown in Figure 8-7, provides access to FTP’s only real built-in security features.
Figure 8-7.

By default, all computers are allowed access to the FTP site; here, I’ve configured it to deny access to all computers with a 10.10.2.x IP address. This is the only real security that FTP offers, and as you can see it’s not much. I’ll talk more about FTP security in the next section.

Like Web sites, you can create virtual directories under an FTP site. To do so, just right-click the site, select New, and then select Virtual directory. You’ll go through a process much like that for Web sites, starting with the dialog shown in Figure 8-8, where you provide the name of the virtual directory.
Figure 8-8.

Figure 8-9 shows how you enter or select the path for the physical folder that the virtual directory will map to.
Finally, Figure 8-10 shows the permissions available for the virtual directory: Read or Write.
Once the virtual directory is created, you can right-click it to access its own properties. They’re basically a subset of the properties available to an entire FTP site. Figure 8-11 shows the first properties tab, which allows you to set the virtual directory location and permissions—the same things you specified when you created it.
The second tab allows you to specify unique IP address restrictions for the virtual directory.

As you can see, FTP sites don’t offer much in the way of complexity or flexibility.

**Security**

FTP security has long been a sore point. The FTP protocol is old and hasn’t evolved as much as HTTP. Today, FTP’s use is limited mainly to anonymous downloads for software updates, where security isn’t much of a concern. It can also be helpful if you and a business partner need to exchange files that are too large to e-mail. If you’ve set up an FTP site for that purpose, here’s a tip: When you’re not actively using the site to exchange files, stop it. Right-click the site in IIS Manager and select Stop from
the context menu. Doing so disables the site, preventing anyone you don’t know from uploading files. When you need to use it again, just restart the site the same way: Right-click it and select Start.

There’s a standard for SSL-encrypted FTP (RFC 2228), and a number of third-party FTP clients support it. However, IIS doesn’t. If you want to use FTP in a more secure fashion, including using encryption, you’ll need to obtain a third-party FTP server that supports the new specification. It’s unclear why Microsoft—in light of its Trustworthy Computing initiative that delayed the release of Windows Server 2003 to begin with—declined to add encrypted FTP to IIS. One possible explanation is that it sees FTP’s use waning; with technologies like BITS and WebDAV offering functional replacements for FTP as well as better security, perhaps FTP is closer to going the way of the dodo.

Many administrators feel that access to FTP, at least, can be secured by refusing anonymous connections and by creating local user accounts for FTP users. I disagree; because FTP offers only clear-text transmission of those credentials, the use of even local user accounts shouldn’t be an option. Those credentials are to easily compromised, effectively placing you back to anonymous access.

**Summary**

IIS’ FTP server is an adequate way to provide simple, FTP access to files and folders or to exchange files with other businesses or colleagues. With a lack of true security features, FTP should play only a limited role in your organization, ideally as an anonymous file download server or as an occasional-use way to exchange large files.

In the next chapter, I’ll introduce you to IIS’ other major two server components: SMTP and NNTP.
Chapter 2: SMTP and NNTP Configuration

As I’ve already mentioned, IIS includes basic SMTP and NNTP capabilities, allowing it to function as a rudimentary mail server and newsgroup server. Both protocols are prerequisites for newer versions of Exchange Server, which rely on IIS’ built-in functionality rather than providing their own services. In this chapter, I’ll introduce you to both of these services and cover their configuration options and security features.

Installation

Like practically everything else in IIS, SMTP and NNTP services are optional components. To install them:

1. Open Add or Remove Programs from the Control Panel and then click on Add/Remove Windows Components.

2. Select Application Server and click Details.

3. Select Internet Information Services and click Details. You’ll see the dialog box in Figure 9-1, where you’ll select SMTP or NNTP (or both).
4. Click OK on the open dialog boxes to close them and install the new services.

As with FTP, installing SMTP or NNTP creates a Default site for the appropriate service. You’ll want to immediately review the Default site configuration to ensure it meets your organization’s needs, particularly with regard to security.

**Configuring SMTP**

IIS’ SMTP server is a fairly complex piece of software. That shouldn’t be surprising. It has to be robust enough to support Exchange Server, which relies upon it for basic SMTP communications. Keep in mind that IIS can host multiple SMTP virtual servers, allowing you to specify a unique configuration for each one. In most circumstances, though, you’ll just use one SMTP virtual server.

You’ll use the IIS Manager to configure the SMTP server. To display the server’s properties, right-click the server and select Properties from the context menu. You’ll begin in the dialog box shown in Figure 9-2.
Because you can have multiple servers, IIS needs to know how to direct SMTP traffic to each one. A unique IP address for each virtual server is the best way, because other SMTP servers are going to expect to find yours listening on TCP port 25. However, you can specify a different port number by using the Advanced button, which would allow you to host multiple SMTP servers on each IP address (by configuring each to have a unique port number).

You can also use this tab to limit incoming connections, set a timeout interval for idle connections and enable logging. Note that logging is disabled by default; that’s different than in Web or FTP virtual servers. It’s unusual for companies to want to log SMTP traffic, in part because there tends to be so much of it.
The next tab, shown in Figure 9-3, allows you to configure access permissions to the SMTP site.

![Default SMTP Virtual Server Properties](image)

**Figure 9-3.**

The second button, **Certificate**, allows the installation of a digital certificate. This uses the same process as a Web certificate (and, in fact, is the same certificate; if you have a certificate installed for a Web site then the SMTP site can use it, too).

**NOTE:** Refer to **IIS Part I** for more information on installing certificates.

The **Authentication** button brings up the dialog shown in Figure 9-4.
Here, you control how users authenticate to the SMTP server. Anonymous is enabled by default because few Internet SMTP servers are configured to have credentials that your server would recognize; SMTP communications are typically anonymous. However, you can enable basic (which is clear-text and not recommended) or Windows-integrated authentication.

Keep in mind that POP3 and IMAP clients all need to use an SMTP server to send outgoing mail. POP3 and IMAP themselves only handle the process of retrieving stored mail from a mailbox server, not sending mail. One reason to enable authentication on your SMTP server is if e-mail users will be using it to send outbound e-mail.

Clicking Connection on the Access tab displays the dialog box shown in Figure 9-5. Here, you decide what computers—by their IP address—are allowed to connect to the virtual computer. You can also specify by domain name, but I don’t recommend...
it; it’s too easy for an attacker to fake (or spoof) a domain name, defeating your security. Actually, IP addresses can be spoofed as well, although it’s difficult to do so if the attacker intends to maintain an ongoing communication with your server.

![Connection dialog](image)

*Figure 9-5.*

Finally, the Relay button on the Access tab displays the dialog shown in Figure 9-6.
Relaying is an important SMTP concept. Whenever an SMTP server accepts an e-mail, it can do one of two things with it: deliver it to a local mailbox, if one exists, or relay it to another domain. For example, if your company is named company.com, and your mail server receives e-mail for joe@company.com, then the server can reasonably expect to find a local mailbox named Joe. If it does, then the incoming mail is delivered. If it doesn’t, then the server may reject the message, because it knows that the mailbox can’t be found elsewhere. On the other hand, if your server receives an e-mail for tim@organization.org, then it knows that no local mailbox should exist and that the message will have to be relayed to organization.org’s mail server.

Relaying is normal, despite the bad connotations associated with it due to unsolicited commercial e-mail (or spam). For example, if a user in your company needs to send an e-mail to someone at organization.org, then your SMTP server is going to have to relay the message for them. Relaying is only bad when someone
other than your users use your server for relaying—a technique often used by spammers to hide their tracks and place the blame on you for their spam. Allowing anyone to relay mail is called open relaying, and many ISPs now prohibit their customers from having open relays.

IIS allows you to close your relaying and is set up to be closed by default. As shown in Figure 9-6, the default settings allow only authenticated users to relay. However, referring back to Figure 9-4, you’ll see that no authentication protocols are selected. You’ll need to pick one in order for users to be able to relay outgoing e-mail through the server; which authentication protocols you select depends primarily on what protocols your e-mail clients support.

TIP: Note that Exchange Server relies on IIS’ built-in SMTP service, but adds its own administrative interface. You’ll be able to configure more complex and flexible settings for Exchange without using IIS Manager at all.

Figure 9-7 shows the next main properties tab, Messages.
This tab allows you to configure basic size limits for incoming and outgoing messages, to help conserve server resources. You can also specify the folder in which IIS saves bad e-mail messages for your review.

Figure 9-8 shows the Delivery tab. This tab allows you to configure outbound delivery behavior, such as how many times IIS tries to deliver outbound mail and how long it waits between attempts. You can also configure outbound security and connections properties, which allows your SMTP server to authenticate itself to another server. This is particularly helpful if your SMTP server will relay all outgoing e-mail through another SMTP server, such as one maintained by your ISP, which requires authentication.
Figure 9-8.

Figure 9-9 shows the **LDAP Routing** tab. LDAP routing allows the SMTP server to contact LDAP servers, such as Active Directory domain controllers, for recipient addressing and routing information. To use this feature, simply provide the required information about the LDAP directory that IIS should connect to, including authentication credentials.
Finally, Figure 9-10 shows the **Security** tab. This tab specifies the local and domain user accounts that are allowed to manage the SMTP virtual server.
One of the things you can do with an SMTP server is configure it to support various domains. To do so, right-click the server’s Domains subfolder in IIS Manager, select New, and then select Domain. As shown in Figure 9-11, you’ll first need to indicate if this is a remote domain or an alias for an existing domain.

Figure 9-10.
Next, as shown in Figure 9-12, you'll provide the name for the new domain. This can be a regular domain name, such as braincore.net, or a wildcard, such as *.net. The properties of the domain will determine how messages sent to that domain are handled.
After adding the new domain, it will be listed in IIS Manager, as shown in Figure 9-13.
Figure 9-13.

Figure 9-14 shows the domain's properties dialog, which you can access by right-clicking the domain and selecting Properties from the context menu.
Figure 9-14.

On this dialog, you can specify that incoming mail received by your SMTP server is allowed to be relayed to this domain. You can also configure your SMTP server to send the older HELO command when connecting to this domain’s SMTP server, rather than the newer EHLO command. HELO might be required when connecting to a particularly old SMTP server that doesn’t support newer, enhanced SMTP commands.

You can also determine how mail to this domain will be delivered. You have the option of having your SMTP server use DNS to directly contact the receiving SMTP server, or you can have your SMTP server forward all mail bound for this domain to a particular SMTP server, called a smart host. This is typically maintained by your ISP and is often the only way ISPs will permit outbound SMTP traffic.
**TIP:** To configure these properties for all possible outgoing domains, create a domain named "*" and set its properties as desired. All outgoing domains will match the "*" wildcard.

Clicking the **Outbound security** tab displays the dialog in Figure 9-15, where you can configure your SMTP server to connect anonymously or use specified basic or Windows credentials.

![Outbound Security Dialog](image)

Figure 9-15.

Finally, Figure 9-16 shows the **Advanced** tab, where you can configure advanced SMTP properties. It's unusual to use this page, and if you do so you should work with your ISP to obtain the correct settings. The purpose of this tab is to configure ATRN properties, which allows your SMTP server to store, or queue, outbound mail for this domain. The expectation is that the domain's own SMTP server will eventually log in using a user account you specify on this tab and issue an ATRN command. That command signals your server to begin
sending queued messages for the domain. This feature set is used primarily to allow your SMTP server to store mail for another SMTP server, which will connect to yours via dial-up to pull its mail.

![Configuring NNTP](image)

Figure 9-16.

**Configuring NNTP**

IIS’ NNTP service provides a number of configuration options. There are actually three main categories of configuration: The virtual NNTP server itself, newsgroups and expiration policies.
**NNTP Virtual Server**

Figure 9-17 shows the basic properties for the NNTP virtual server. You can access this dialog by right-clicking the virtual server and selecting Properties from the context menu.

![Default NNTP Virtual Server Properties](image)

*Figure 9-17.*

As with most IIS virtual servers, the general properties allow you to configure the virtual server’s identity, using a unique combination of IP address and port number. As with SMTP, changing the port number isn’t recommended since news reader software is usually hard coded to try the standard port. You can, however, click the Advanced button to set a different port number. You can also limit incoming connections, set a timeout for idle connections and enable logging.
Logging is disabled by default since newsgroup access is rarely audited and because a busy NNTP server can generate an incredible amount of log data.

Figure 9-18 shows the **Access** tab, where you can install a server certificate (click the Certificate button), modify authentication properties or modify connection restrictions.

![Default NNTP Virtual Server Properties](image)

*Figure 9-18.*

Figure 9-19 shows the authentication methods. NNTP offers a wide variety, including anonymous access, basic access (in which the credentials are sent in clear text) and Windows-integrated authentication. You can also enable SSL-based client authentication, which uses digital certificates held by clients (such as smart cards) to authenticate incoming connections.
Figure 9-20 shows the connection restrictions dialog box, which, like SMTP’s connection restrictions, allows you to control incoming connections by IP address, IP address range or domain name. Again, domain names aren’t recommended because they can be more readily spoofed than IP addresses.
On the **Settings** tab, shown in Figure 9-21, you’ll configure the main settings for this NNTP virtual server. You can allow clients to post to the newsgroups and limit their post sizes. You can also allow the posting of feeds from other NNTP servers and limit those sizes. You can decide whether or not other NNTP servers will be allowed to pull newsgroup messages from your server.
Control messages allow users with the right knowledge to send newsgroup messages that create, modify or remove newsgroups. These are disabled by default, and you can use IIS Manager to perform these tasks.

You can also configure the SMTP server used for moderated newsgroups, specify a default moderator domain and specify the e-mail address of the server’s administrator. Moderated newsgroups allow posts from users, but require a moderator to approve each post before it is actually placed into the newsgroup. Communications to the moderator are conducted by SMTP, meaning the moderator receives an e-mail whenever a post is available for his or her review.

Figure 9-21.
Shown in Figure 9-22, the **Security** tab allows you to configure the user accounts that are granted operator privileges for this NNTP virtual server.

![Security tab in IIS Manager](default.png)

**Figure 9-22.**

**Newsgroups**

*Newsgroups* are collections of messages related to a particular topic. These can be moderated or open, allowing users to freely post messages. Newsgroups are also often referred to as discussion groups, since they facilitate a written form of group communication. A single NNTP virtual server can host a nearly unlimited number of newsgroups. The IIS Manager lists the newsgroups currently available, as shown in Figure 9-23.
To create a new newsgroup, simply right-click Newsgroups and select New Newsgroup from the context menu. You’ll see the dialog box shown in Figure 9-24, where you’ll provide the new newsgroup’s name.
Next, as shown in Figure 9-25, you'll provide a description and a display, or
*pretty*, name for the newsgroup.
Once a newsgroup has been created, it will appear in the list in IIS Manager. You can right-click any newsgroup and select Properties from the context menu to modify its properties. Figure 9-26 shows the properties dialog box for a newsgroup.

Other than modifying the description and pretty name, your only real options with a newsgroup are to make it read-only or moderated and to select a moderator.

Expiration Policies

The NNTP virtual server will retain newsgroup messages until they expire. You determine how long that is by creating an expiration policy. Note that IIS doesn’t
create a default expiration policy, meaning that newsgroup messages will be retained until the server’s hard drive has no more room for them.

Expiration policies are listed in IIS Manager, along with newsgroups. To create a new policy, right-click Expiration policies and select New expiration policy from the context menu. You’ll see the dialog shown in Figure 9-27, where you specify a name for the new policy.

![New NNTP Expiration Policy Wizard](image)

**Figure 9-27.**

Next, as shown in Figure 9-28, you’ll specify the newsgroups that this policy will apply to. In this example, the policy applies to all newsgroups, thanks to the "*" wildcard. You can, however, create multiple policies, allowing different newsgroups to maintain a different number of messages.
Figure 9-28.

Figure 9-29 shows the final dialog, where you specify the maximum age, in hours, for each message in the newsgroup. IIS will automatically remove messages older than this time period.
Summary

IIS’ SMTP and NNTP servers add excellent communication capabilities to the IIS platform. Combined with Windows Server 2003’s POP3 server, which I’ll cover in the next chapter, Windows Server 2003 can be used as a basic e-mail server, suitable for small shops with only a few mailboxes. The SMTP and NNTP virtual servers also provide security adequate to protect you against the most common SMTP- and NNTP-based attacks. Configuring SMTP relaying restrictions is especially important, so that your server can’t be used to relay spam on the Internet.
Chapter 3: POP3 Server

Windows Server 2003 includes a POP3 server which, while not technically part of IIS, integrates with IIS SMTP service to provide basic e-mail server capabilities. Remember that SMTP is used to send and receive mail; POP3 is used to allow e-mail clients to retrieve their stored e-mail. Essentially, IIS' SMTP service allows incoming mail to be received and then stored on disk. POP3 allows users to access their e-mail from mailboxes, and the SMTP service acts as a relay for users' outgoing e-mail.

Installation

Installing the POP3 server is much like installing IIS components:

1. Open the Windows Components Wizard, shown in Figure 10-1, and select E-mail Services. Click the Details button.

![E-mail Services dialog box]

Figure 10-1.
2. As shown in Figure 10-2, select the POP3 Service item. I usually don’t include the service’s Web administration component, since it’s often easier to administer it through the normal MMC snap-in.

3. Click OK to close the dialog boxes and the POP3 service will be ready to go.

**Configuration**

Figure 10-3 shows the POP3 service after you’ve just installed it. As you can see, it doesn’t include any domains or user mailboxes. You’ll need to create a domain for
each e-mail domain—like braincore.net or mcpmag.com—that you want the server to handle e-mail for. Within each domain, you’ll create mailboxes.

![SERVER1 Properties dialog box](image)

**Figure 10-3.**

First, however, you’ll want to check the server’s basic properties. Simply right-click the server and select Properties from the context menu. You’ll see the dialog box shown in Figure 10-4. Here, you can specify a number of server-wide properties.
• The Authentication method you choose determines how users will log on to their POP3 mailboxes. As shown, the Local Windows Accounts option allows users to log on by using an account that’s defined on the local server; other options allow users to use a domain user account.

• The Server Port option is set to 110 by default. This is the port that clients will use to connect to the POP3 service; 110 is the standard port number, and you shouldn’t change it unless you’re willing to also change all of your clients’ e-mail applications.
• **Logging Level** is set to Minimum by default, which means only critical errors are written to the log. You can raise this logging level if you need to troubleshoot the service and want to see additional details about what’s going on under the hood.

• The **Root Mail Directory** is the folder where users’ mailboxes are stored. Each mailbox has its own subfolder; the folder you specify here is the top-level storage location for all mailboxes.

• **Require SPA for all client connections** requires that your clients use a Microsoft e-mail client. SPA is a variation of the NTLM authentication protocol, and is proprietary to Microsoft. Outlook and Outlook Express both support SPA.

• **Always create an associated user for new mailboxes** is great when you’re using the Local Windows Accounts authentication method. When you create a new mailbox in the POP3 console, you’ll also be creating an associated user account that can use the new mailbox.

## Domains and Mailboxes

The POP3 service needs to know which domains it will store e-mail for. To create a new domain, simply click the New Domain link in the POP3 Service console. As shown in Figure 10-5, domain are listed in both the main console window and the left-hand tree view. A single POP3 service can receive e-mail for many different domains. However, mailboxes can only belong to one domain.
Figure 10-5.

That’s a bit different than full-fledged e-mail systems like Exchange. For example, a single Exchange mailbox can be configured to have multiple SMTP addresses, such as braincore.com, braincore.net and braincore.org. Incoming e-mail is delivered to the mailbox with the matching e-mail address. In the POP3 Service, however, e-mail is first routed to a domain and only then to a mailbox. So if you own three domain names—company.org, company.com and company.net—you will probably need to pick one to use for incoming e-mail. Otherwise, your users will have to maintain a separate mailbox for each domain.

To add a mailbox to a domain, simply right-click the domain and select **New Mailbox** from the context menu. You only have to provide the name of the user—without the domain name—in order to create a mailbox. You also have the option, as shown in Figure 10-6, to create an associated local user account. If you choose to do so, you’ll need to specify an initial password for the account.
As shown in Figure 10-7, user mailboxes are all listed in the console. Also shown is the total size of the mailbox. Incoming messages are stored as single files under the mailbox's folder, and the Size of Mailbox column simply totals the size of all the files in the mailbox.
Because the POP3 service is so basic, it doesn’t provide much in the way of management options. You can control users’ maximum mailbox size by applying a regular Windows disk quota to the volume containing the mailboxes; messages are owned by the user account associated with the mailbox. However, there’s no autocleanup, deleted item retention or any of the advanced features you might associate with a product like Exchange Server. Encourage users to delete their messages from the server after retrieving them—an option provided by POP3 client software—to keep their mailboxes from growing too large.

**Configuring SMTP Support**

As I’ve mentioned before, POP3 is only half the battle of creating an e-mail server. POP3 simply provides a way for users to retrieve their e-mail from mailboxes; SMTP is required to actually receive that e-mail and put it in the mailbox and to send
outgoing e-mail. I covered SMTP in the previous chapter; you’ll simply need to create a matching SMTP domain for each domain that your POP3 server supports. Figure 10-8 shows the SMTP service configured with a domain to match what I set up in the POP3 service.

![Figure 10-8.](image)

The SMTP domain must be configured as a local domain, meaning delivery will take place locally and that incoming messages for the domain won’t be forwarded somewhere else.

You’ll also need to configure the SMTP service to handle outgoing e-mail, by enabling it as a relay. As I described in the previous chapter, I don’t recommend
allowing relaying by unauthenticated users, since doing so leaves you open for bulk e-mailers.

**Summary**

Windows Server 2003’s POP3 service, while not strictly a part of IIS 6.0, integrates with IIS to provide a complete basic e-mail server. By using the POP3 service to allow users to access their mailboxes, and IIS’ SMTP service to handle incoming and outgoing mail traffic, you can quickly create a rudimentary mail server for small offices, internal projects or for testing. In fact, I’ve often used the built-in POP3 and SMTP services to create “external” mail domains in a lab environment, allowing me to more easily test and troubleshoot Exchange Server and other corporate e-mail platforms.
Chapter 4: The .NET Framework

You may wonder why a book on deploying IIS, clearly written for administrators, would include a chapter on the .NET Framework. Isn’t the Framework for software developers?

It’s true that developers do use the Framework to develop applications, including applications written for ASP.NET. Those ASP.NET applications, by the way, run under IIS 6.0, which is one reason you as an administrator need to understand the Framework. More importantly, there’s an entire administrative component to the Framework that allows you to configure security and other settings to ensure a safer, more reliable application server. That’s the big reason or this chapter: to introduce you to the .NET Framework’s concepts and show you how to effectively administer the Framework on your IIS servers.

A Framework Crash Course

While the Framework includes a lot of cool concepts and capabilities, they tend to be of real interest only to software developers. Here’s what you really need to know as an administrator.

The various .NET languages—VB.NET, C#, Managed C++, and so forth—are all used to write .NET applications. However, when a developer finally compiles his or her application into an EXE or DLL or whatever, what’s produced isn’t true, binary Windows code. The .NET compilers basically just translate the VB.NET (or whatever) into Microsoft Intermediate Language, or MSIL. That’s the language inside a .NET EXE or DLL.

When someone tries to execute a .NET EXE or DLL, the .NET Common Language Runtime (CLR) loads into memory first. It then loads a special component called the Just-in-Time (JIT) Compiler, which compiles the MSIL into binary code that Windows can execute. The binary code actually runs under the CLR, in much the same way that a Java applet runs inside a Java Virtual Machine (JVM). In fact, in
everything but name, the CLR is simply a Microsoft-centric JVM. Instead of running Java, the CLR runs MSIL. The CLR gets to control everything an application can do.

Application files—EXEs and DLLs, mainly—are called assemblies. Unlike old-style EXEs and DLLs, assemblies don’t require registry entries, COM registration, and so forth. Everything the CLR needs to know about an assembly is included right within the assembly itself. This information includes other assemblies that are required to run, the assembly’s version, and the like. The practical upshot of this is that you can deploy .NET applications by simply copying their files. You don’t really have to run a special Setup program to register DLLs and do all the other stuff we usually have to worry about as administrators.

The Framework also seeks to address the “DLL Hell” issue that you’re probably sick and tired of dealing with. “DLL Hell” is what happens when multiple applications install their own versions of a particular common DLL, and then wind up breaking each other because the various DLL versions aren’t cross-compatible as much as they should be. The Framework solves this by simply spending disk space and keeping all versions of all assemblies.

Here’s how it works: When an assembly is compiled, a list of the assemblies it uses, along with their version numbers, is saved into the assembly, in a special data structure called a manifest. The assembly actually forms something called a contract, which says, “These are the support assemblies I need, and these are the specific versions I need.” The Framework uses this information to make sure the right assemblies are provided when an application is run.

Common assemblies—those provided by Microsoft, primarily, and used by many applications—are stored in the global assembly cache. That’s really just a fancy name for the folder where the assemblies’ files are stored, under the System32 folder. Very few applications install additional global assemblies; most of the time, any assemblies that come with an application are stored in its local assembly cache. Again, that’s just a fancy name for a folder, in this case the folder where the application is installed.
As an administrator, you have a high level of control over applications’ abilities to work with assemblies and to interact with the operating system. Remember: The CLR controls what a .NET application can actually do; you tell the CLR what you’ll allow. A proper security configuration can prevent applications—even ones that are compromised by a hacker—from hurting your system. Even if an application has an exploitable security flaw, it won’t be able to do anything more than the CLR lets it.

.NET Framework Administration

Windows comes with a .NET Configuration console, which you’ll use to configure the Framework. Do you absolutely need to do this for every Web server you’re running? No, of course not; if you’re not using ASP.NET or any .NET applications in the Web site, then you can completely ignore this chapter. However, more and more Web sites are beginning to incorporate .NET elements, so you should take the time to understand what’s going on inside the Framework.

Figure 11-1 shows the .NET Configuration console and the list of assemblies in the global assembly cache (GAC). This is useful mainly for informative and troubleshooting purposes, as this screen displays assemblies’ version numbers alongside their names.
Figure 11-1.

The heart of the Framework’s security infrastructure is in the Runtime Security Policy (RSP). The RSP consists of two main components: Permission sets, which define specific sets of permissions that an application can have, and Code groups, which define groups of applications. By applying permission sets to code groups, you control what applications can do.

Figure 11-2 shows the default code groups: FullTrust, SkipVerification, Execution, Nothing, LocalIntranet, Internet, and Everything. You can select any one of these to view the actual permissions they contain, or click Create New Permission Set to create a new set of your own.
Figure 11-2.

As shown in Figure 11-3, you’ll define a name and provide a description for your new code set.
Figure 11-4 shows how you add permissions to the permission set. By default, no permissions are available, meaning any code group assigned to this permission set won’t be able to do anything. To assign a permission to the set, select the permission and click Add. As shown, I’ve selected File IO, giving this permission set the ability to work with the file system.
Most permissions will require further definition. As shown in Figure 11-5, my File IO permission can be scoped to allow just read, write, append and path discovery permissions for a specific file path. Or, I can configure it to allow unrestricted access to the file system.
Figure 11-5.

Figure 11-6 shows the detail for the Directory Services permission, where I’ve specified that assemblies be able to do anything they want to all directory services paths. Note that none of these permissions overrides native permissions like AD permissions or NTFS permissions; the permissions I’m defining here simply tell the CLR what the application is allowed to try. For example, unrestricted directory services access does not mean an application will suddenly be a Domain Admin; Active Directory’s underlying permissions will still govern what the application can accomplish. What I’m specifying here simply tells the CLR that it’s OK for the application to make the attempt.
Figure 11-6.

Figure 11-7 shows my completed permission set with two assigned permissions. As you can see from the list of available permissions, control is pretty granular. For example, when assigning the Event Log permission, you can specify which event logs you’re talking about. When specifying Registry access, you can specify particular keys within the registry. The Framework allows you to be detailed on what applications are allowed to do.
Figure 11-7.

Figure 11-8 shows my completed permission set, listed with the others that come with the Framework by default. You might now be asking yourself, “What happens if I assign a permission set to an application, but it's less than the application needs? Will the application crash?” Possibly.
The .NET Framework will not let an application do anything that violates the permission set you assign. If an application tries, an error is generated and the application can choose to deal with it. One of three situations can occur.

- Applications can check in advance by telling the Framework what permissions they need to run. The Framework will compare that list to the permission set, and return errors if insufficient permissions are available. The application can then choose to continue in a degraded state (possibly providing fewer features), or to exit gracefully.

- Applications can check for errors when performing actions that might be restricted. If an error occurs, the application can handle it any way the developer chose.
• Applications might not check at all, meaning they’ll receive an unhandled error if they try to perform a restricted operation. This will generally result in the application exiting unexpectedly—or, in other words, crashing.

Well written applications will obviously take the first approach. Poorly written applications will take the last approach and may crash when they aren’t allowed to do what they want. So what? You don’t want poorly written applications. They’re more likely to have exploitable security flaws, and if you give them the permissions they need to run without crashing, the application might be compromised at a later time and wind up doing more damage than you thought possible.

Getting back to configuring the Framework, the next step is to create and configure code groups. By default, all .NET applications belong to a code group named “All Code,” which includes, as the name implies, all code. You can create your own code groups. A code group doesn’t contain a list of applications, exactly; it contains a set of rules for identifying applications. For example, in Figure 11-9, I’m creating a new code group named MyCodeGroup.
Figure 11-9.

In Figure 11-10, I’ve indicated that this code group will include all code meeting an application directory rule. I could also specify other rules that would determine whether or not an application fell into this group.
• The **Application Directory** rule includes all assemblies in an application's folder.

• A **Hash** is a cryptographic checksum taken from an assembly that uniquely identifies that assembly. If an assembly is changed, the hash will no longer be valid and so the assembly won't meet the rule.

• A **Publisher** condition checks the publisher of an assembly when the assembly is signed with a digital signature.

• A **Site** rule checks the site that an assembly came from.

• A **Strong Name** is a combination of several components, including a type of digital signature.
• A URL checks the URL that a Web-based assembly came from.

• A Zone applies the Internet Explorer security zones to determine where an application came from.

Note that code groups are in a hierarchy. For example, all code falls into the “All Code” code group; any code groups you create will fall underneath that and will narrow down the included code. Assemblies can “belong” to more than one code group by meeting the selection criteria for each; by default, assemblies will receive the permission sets assigned to each code group that the assemblies “belong” to.

Figure 11-11 shows how a code group is assigned to a specific permission set. In this example, I’m using the permission set I created earlier. Any assembly meeting the criteria for this code group will receive the permissions in the MyPermissions permission set.
**Figure 11-11.**

**TIP:** The entire RSP can be applied at the Enterprise, Machine, and User level. It can be deployed through Group Policy to centrally control an entire enterprise, and you can create specific exceptions, if necessary, for individual users or computers.

By carefully creating code groups and permission sets, you can determine what .NET applications are allowed to do in your enterprise. For example, suppose you have internal developers creating custom applications in VB.NET. If your developers digitally sign their assemblies, then you can specify a code group that looks for that signature and allows full permissions for those assemblies. All unknown assemblies can then be restricted to very minimal permissions, ensuring that outside assemblies downloaded by users or otherwise brought into the environment can’t do much harm.
Summary

While the .NET Framework might not seem particularly exciting, it’s an important component of IIS 6 because of ASP.NET (which is itself a .NET application, written in C#). If your IIS servers will include ASP.NET applications, then you can configure the Framework’s security infrastructure to ensure that the assemblies included in the Web site don’t have any more permissions than they need. Then, even if the assemblies include security flaws and vulnerabilities, the Framework itself will help ensure that assemblies don’t have any ability to create serious harm. Similarly, if you have assemblies that require potentially dangerous permissions, you’ll know to be extra careful when testing those assemblies to ensure that they don’t have any vulnerabilities that can be exploited to harm your servers.
Chapter 5: Automated Installation

IIS doesn’t support its own automated installation routines. However, IIS is completely integrated into Windows’ own unattended setup routines, meaning you can easily deploy IIS servers automatically if you’re prepared to do so as part of the main operating system installation. In this chapter, I’ll explain how to create the necessary files.

Setup Manager

You’ll begin with Setup Manager, which is included on the Windows Server 2003 CD or DVD, under \Support\Tools\Deploy.cab. Setupmgr.exe is designed to create unattended installation answer files, which provide the answers that Windows Setup would normally collect through its user interface. Figure 12-1 shows the first screen, where you’ll select what type of setup you’d like to create. An Unattended setup uses the normal Windows Setup routines, while providing an answer file rather than requiring you to manually input setup information.
Figure 12-2 shows the next screen, where you’ll select the operating system you want to install. I’ve selected Windows Server 2003, Web Edition, which is a common choice when deploying Web servers. Remember, Web Edition costs significantly less than Standard or Enterprise Edition and is designed for servers that will primarily serve as Web servers.
Next, as shown in Figure 12-3, you’ll select the type of setup. I’ve selected Fully automated, which will make for the most streamlined installation. You’ll need to carefully construct the answer file so that all of the Windows components you want—including IIS—are installed automatically.
Finally, as shown in Figure 12-4, you’ll select how to distribute Windows. You can use a normal CD-based setup, or have Setup Manager create a network-based distribution share for you.
After you’re done, Setup Manager will allow you to start providing answers. The information you’ll provide is the same as you’d provide in an ordinary Windows installation, including a product key, administrator password, networking components, and so forth. Upon completion, Setup Manager will produce an answer file with contents similar to the following:

```plaintext
UnattendedInstall="Yes"
[Unattended]
    UnattendMode=FullUnattended
    OemSkipEula=Yes
    OemPreinstall=No
    TargetPath=\WINDOWS
[GuiUnattended]

AdminPassword=e52cac67419a9a224a3b108f3fa6cb6d8846f
7eae8fb117ad06bdd830b7586c
```
Unfortunately, this answer file is just enough to create a default Windows Server 2003 installation. As you’ve learned, the default installation doesn’t include IIS, so you’d still need to install those components manually. However, you can add to the answer file to include IIS and its optional components.

**NOTE:** I won’t be covering what each optional component does; refer to the appropriate chapters in this book for a description of each component and how it works, along with its potential security implications.

## The Answer File

The first section you need to add to you answer file is [Components]. This section specifies additional optional components. For example, to install everything that IIS includes, along with the POP3 service, you might add the following:

```
[Components]
  appsrv_console = On
  aspnet = On
  BitsServerExtensionsISAPI = On
  BitsServerExtensionsManager = On
  complusnetwork = On
  dtcnetwork = On
  fp_extensions = On
  fp_vdir_deploy = On
  iis_asp = On
  iis_common = On
```

iis_ftp = On
iis_inetmgr = On
iis_internetdataconnector = On
iis_nntp = On
iis_serversideincludes = On
iis_smtp = On
iis_webadmin = On
iis_webdav = On
iis_www = On
iis_www_vdir_scripts = On
inetprint = On
Pop3Admin = On
Pop3Service = On
Pop3Srv = On
TSWebClient = On

Here’s what each one does:

• **appsrv_console** = On. Installs the Application Server management console. You don’t need this on an IIS server if you plan to manage it from a client computer that has the console installed.

• **aspnet** = On. Installs ASP.NET.

• **BitsServerExtensionsISAPI** = On. Installs the BITS server extensions.

• **BitsServerExtensionsManager** = On. Installs the BITS server extensions manager, which appears in the Application Server management console.

• **complusnetwork** = On. Installs COM+ network access.

• **dtcnetwork** = On. Installs the Distributed Transaction Coordinator network access.

• **fp_extensions** = On. Installs the FrontPage Server Extensions.

• **fp_vdir_deploy** = On. Installs the Visual InterDev rapid deployment extensions.

• **iis_asp** = On. Installs Active Server Pages.
• **iis_common = On.** Installs the IIS core components, which are required for IIS to function.

• **iis_ftp = On.** Installs the FTP service.

• **iis_inetmgr = On.** Installs the Internet Information Services management console. Again, you don’t need the console on the server if your management workstation has the console.

• **iis_internetdataconnector = On.** Installs the Internet Data Connector.

• **iis_nntp = On.** Installs the NNTP service.

• **iis_serversideincludes = On.** Installs support for server-side includes.

• **iis_smtp = On.** Installs the SMTP service.

• **iis_webadmin = On.** Installs the IIS Web-based administration interface.

• **iis_webdav = On.** Installs the WebDAV component.

• **iis_www = On.** Installs the WWW publishing service.

• **iis_www_vdir_scripts = On.** Installs the optional Scripts virtual directory under IIS.

• **inetprint = On.** Installs Internet Printing services.

• **Pop3Admin = On.** Installs the POP3 Service console.

• **Pop3Service = On.** Installs the POP3 service files.

• **Pop3Srv = On.** Sets up the default POP3 service.

• **TSWebClient = On.** Installs the Web-based Remote Desktop Connection client.

*TIP: For a complete list of available components, refer to the Ref.chm help file located in the Deploy.cab file on the Windows Server 2003 CD or DVD.*
**TIP:** The default for all components is Off, so if you don’t want to install a particular component, simply exclude it from the [Components] section.

You’ll also need to add an [InternetServer] section to your answer file. It might look like this:

```plaintext
[InternetServer]
   ApplicationDependency=ASPNET20;ASP60
   DisableWebServiceOnUpgrade=False
   PathWWWRoot=C:\Inetpub\Wwwroot
   PathFTPRoot=C:\Inetpub\Ftproot
   ServiceManualStart=ftp,www
```

Here’s what that does:

- **ApplicationDependency** line specifies which Web extensions are required; my example above specifies ASP.NET and ASP. For a complete list of available extension names, refer to the Ref.chm file included with Setup Manager.

- **DisableWebServiceOnUpgrade** is an option you’ll normally set to False. When set to True, Setup will, as a security precaution, install but disable the World Wide Web service.

- **PathFTPRoot** and **PathWWWRoot** specify the root paths for the FTP and WWW services, respectively. The default values are shown in my example.

- **SvcManualStart** specifies what services should be set to manual startup. As a security precaution, I always specify *ftp,www*, which allows me to install IIS but keep those services turned off until I’m sure the server is configured to be as secure as possible. Then I’ll set those services for automatic startup before placing the server into production.

**TIP:** Remember that servers are especially vulnerable when they’re first installed because they haven’t yet received the latest service packs and security patches available from Microsoft. Try to keep servers on a protected network segment, preferably one that only contains a Software Update Services (SUS) server, until the latest patches are installed and the server is configured to have the appropriate level of security.
Summary

Automatically installing IIS along with Windows is a great idea when you’re deploying Web servers and need to override Windows’ default behavior of *not* installing IIS by default. Keep in mind that IIS can also be deployed along with Windows in other ways, such as through Remote Installation Services (RIS), drive imaging, and so forth. However you decide to deploy IIS, automating the process will not only help save you time and effort, it will also help ensure a more consistent configuration across the Web servers in your organization.
Chapter 6: Locking Down the Server

One of the most important things you can do with any Web server—especially Internet-facing servers—is to reduce your exposure to attacks. Security specialists refer to this as reducing your attack surface, meaning you try to remove product functionality that could potentially be compromised by an as-yet-undiscovered security vulnerability. That’s a theme I’ve stressed throughout this book with regard to IIS: Only install the components that you need. The reason Microsoft chose to make each individual component of IIS a separate installation option was so that you can install only the bits of functionality you need, eliminating unnecessary functionality as a potential attack point.

Looking for Ports

One way that attackers begin looking for weak points on a server is port scanning. Freely-available port scanning tools probe each potential TCP and UDP port to see if a server is listening to those ports; or, in other words, if the server is running some service that accepts connections on those ports.

Figure 13-1 shows the output of the netstat –a command run on a Windows Server 2003 computer.
Each line in the output is a TCP or UDP port that is open for service on the server. Many of the ports are listed by their service names, such as ftp and http; others simply include a port number, such as 3245, one of the ports used by the MSN Messenger service. This port list is an excellent way to see where an attacker would start, since it’s pretty much exactly what a port scanner would list. Examine each

---

**Figure 13-1.**
open port and decide if there’s a Windows or IIS service that is behind that port, but isn’t really necessary. Disabling or uninstalling the service should close the port.

For example, the output in Figure 13-1 shows that the server is listening to the FTP port. If I know that the server isn’t supposed to be functioning as an FTP server, I can uninstall IIS’ FTP component and shut that port down, eliminating one potential attack point on the server.

Another way to lock down ports is to use Windows’ port filtering functionality. To do so, open the TCP/IP properties of any network adapter. Click Advanced, and on the Options tab select TCP/IP filtering and click Properties. You’ll see a dialog like the one shown in Figure 13-2.

As shown, this server is configured only to accept connections on TCP ports 80 and 443—for HTTP and HTTPS traffic, respectively—so it doesn’t matter what other services are running. Only traffic on 80 and 443 will be allowed through to the server.
Here’s the major downside to TCP/IP filtering: It’s a global setting for all network adapters in the server. You can’t, for example, enable it for one outward-facing network adapter while leaving filtering off for an intranet-connected adapter. Even though you configure these settings from within a specific network adapter, it’s effective for all adapters.

**NOTE:** You can use Windows’ built-in IPSec features to perform more granular port filtering, if desired. However, a discussion of IPSec is beyond the scope of this book.

Why is that a problem? If your Web server isn’t a domain member, it might not be a problem; you can restrict the server to the necessary ports and ensure that an attacker can’t hit anything else. However, domain members require more than a dozen open ports to communicate with various domain services, making it almost impossible to create a TCP/IP filter list that both protects the server and allows it to function properly.

**NOTE:** For obvious reasons, you should never place a Windows Server 2003 computer right on the Internet unless it’s running firewall software. Use a firewall to protect your Web servers and other servers, allowing only the necessary traffic to reach your servers.

**Protecting Servers**

Whether you use a firewall or not, you should still lock down as many of a server’s open TCP and UDP ports as possible, simply to reduce the possibility of attack in the event that the firewall itself is compromised. However, a bigger point of concern when locking down a server is more subtle than simply shutting off ports.

Most IIS servers will always be listening on port 80, since that’s what Web servers use. Even if you lock out every other possible port, port 80 still offers plenty of opportunities for attackers. In the past, IIS has been a major source of security vulnerabilities; while Microsoft’s Trustworthy Computing initiative has supposedly made IIS 6 less prone to security problems, the very nature of computer software assures us that IIS 6 will eventually have a security bug or two. That’s why part of locking down a server is preventative maintenance.
Microsoft has created a tool called the Microsoft Baseline Security Analyzer, or MBSA. Version 1.1.1 added support for Windows Server 2003, and it's an excellent tool to run against all of your servers, especially Web servers running IIS 6. MBSA can be run locally on a server, or remotely from a workstation. You can download the tool for free from http://www.microsoft.com/mbsa.

Figure 13-3 shows the tool when you first run it. It's appearance is similar to the Windows Update Web site, and you simply click a link to initiate a scan. Note that you can initiate a scan against multiple computers at once, making it easier to scan all of your IIS servers in one operation.

![Microsoft Baseline Security Analyzer](image-url)
Figure 13-4 shows the main scanning options, where you specify the server to scan and indicate what type of scans you want to perform.

Figure 13-4.

MBSA can:

- Check for known Windows vulnerabilities that you can fix either with a configuration change or a software update.
- Check for poor passwords or a weak password policy.
- Check for IIS-specific vulnerabilities.
• Check for SQL Server vulnerabilities.

• Check for security updates. This option will use the Windows Update service by default, but you can point it to an internal Software Update Services (SUS) server instead, if you have one.

Figure 13-5 shows the results of an MBSA scan.

![Microsoft Baseline Security Analyzer](image-url)

**Figure 13-5.**

As you can see, problem areas are highlighted for immediate review. In this case, several Windows security updates may or may not be out of date; I’ll cover the “may or may not be” part a bit later. Figure 13-6 shows that the tool also highlights areas that are configured correctly, such as the text for local accounts with blank or simple passwords. A green checkmark indicates that the item passed the test and doesn’t need further attention.
Figure 13-6.

Figure 13-7 shows what you get if you click the Result details link under the Windows Security Updates item. Several security updates are listed, along with the problem MBSA had with them. In the first case, MS02-055 is probably not a problem; MBSA was checking for version 5.2.3735.0, and version 5.2.3669.0 was present. That means the version present is later than the one MBSA was checking for, so there’s no problem. You’ll often find that to be the case on security updates: MBSA checks for a specific version based on the last security update that affected the file. However, since that security update came out, the file may have been included in other patches that weren’t necessarily security-related. So the version you’ve got will be later, and will include the fixes MBSA is looking for.
Figure 13-7.

Patch Management

Nowadays, the most important thing you can do to lock down your servers is to implement a rigorous, ongoing program of patch management. As security vulnerabilities are discovered—not if, but when—and patched, you need to apply those patches. Doing so will “lock down” additional attack points that can be used to compromise your systems.

Windows Server 2003 provides an easy way to keep the latest critical and security updates applied to your servers: Automatic Updates. To configure it, right-click My Computer and select Properties from the context menu. Figure 13-8 shows the Automatic Updates tab.
Figure 13-8.

You can configure Automatic Updates to download security and critical patches automatically. This will not automatically apply service packs, recommended updates, device driver updates, or other software; only those patches that Microsoft marks as security-related or critical will be included.

By default, Automatic Updates uses the Windows Update service to obtain patches. However, you can configure a Group Policy object that centrally configures servers to check an internal SUS server, instead. SUS allows you to obtain updates from Windows Update, hold them for administrator approval, and then centrally distribute them to Windows 2000, Windows XP and Windows Server 2003 computers on your network. You can think of SUS as a kind of “in-house Windows
Update” service, and it’s a great way to manage the patches that are applied to your computers.

Permissions

Attackers can do the most damage to IIS is they gain control over the server’s configuration. IIS stores its configuration in a special, XML-formatted metabase, which you normally modify through the IIS management console. There are also command-line tools, including Iisback, Iisftp, Iisweb, and others, which can modify the configuration. These tools present a potential danger, because it might be possible for an attacker to run them by exploiting a poorly-written dynamic (ASP or APS.NET) Web page. Fortunately, that’s difficult; IIS is hardcoded to not run its own configuration tools, helping to close that potential vulnerability.

Still, access to the metabase by an attacker is a real security concern. Only local administrators—which, on member servers also includes the members of the Domain Admins group—can modify the metabase. You might consider making a separate “IIS Admins” group and placing it in the local Administrators group on IIS servers, while removing Domain Admins. Strictly limit the membership of this new group and ensure that the accounts belonging to the group aren’t used as regular logon accounts.

TIP: If an administrative account is used for everyday tasks, such as checking e-mail, then it’s possible for a virus to execute under those administrative credentials and compromise IIS’ security. The best solution is to use Windows’ Runas feature to run the IIS manager console and other administrative tools with your IIS administrative account, while using a regular, non-administrative user account for daily tasks.

Summary

Locking down an IIS server requires attention to several areas: removing unnecessary services and reducing your attack surface; maintaining a close watch over patches and other configuration parameters; keeping the latest patches applied; and carefully configuring server permissions to protect configuration information.
IIS 6 provides some built-in features that help make it more secure, but it’s up to you as the IIS administrator to ensure the server remains locked down and secure.