Indian Computer Emergency Response Team

Enhancing Cyber Security in India

Securing IIS 7.0 Web Server

Department of Information Technology
Ministry of Communications and Information Technology
Government of India

Version: 3.1 Issue Date: 18 November 2010
Disclaimer:

This document is provided for informational purposes only, and is provided entirely “AS IS” basis.

Information in this document, including URL and other Internet Web Site references, is subject to change without notice.

The products mentioned herein are the trademarks of their respective owners.
Contents

1. Introduction .......................................................................................................................... 7
   1.1 Purpose and Scope ......................................................................................................... 7
   1.2 Audience and Assumptions ......................................................................................... 8

2. Background .......................................................................................................................... 8
   2.1 Web Site Security Issues ............................................................................................ 9
   2.2 Security of a Web Server ........................................................................................... 9
   2.3 Steps required for securing any public web server ..................................................... 9

3. Planning and Managing Web Servers .............................................................................. 10
   3.1 Web Server Platforms ............................................................................................... 10

4. Security and IIS 7.0 ......................................................................................................... 12
   4.1 IIS 7.0 Design Principles .......................................................................................... 12
   4.2 IIS 7.0 Design Principle 1: Secure by default design ............................................... 12
       4.2.1 Key Features introduced in IIS7.0 ....................................................................... 12
       4.2.2 Security Changes in IIS 7.0 .............................................................................. 13

5. Securing the Web Server Operating System ................................................................. 15
   5.1 Managing Windows Security ...................................................................................... 15
       5.1.1 Working with User and Group Accounts ......................................................... 16
       5.1.2 Managing the IIS Service Logon Accounts ...................................................... 17
       5.1.3 Managing the Internet Guest Account ............................................................. 18
       5.2 Working with File and Folder Permissions ............................................................ 19
       5.2.1 File and Folder Permission Essentials .............................................................. 19
       5.2.2 Viewing File and Folder permissions ................................................................. 20
       5.2.3 Setting File and Folder Permissions ................................................................... 21
       5.3 Enforcement of Security Configurations through Policies .................................... 22
       5.3.1 Local Security Policy ....................................................................................... 22
       5.3.2 Group Policy ...................................................................................................... 22
           5.3.2.1 Group Policy Editor ................................................................................... 23
           5.3.2.2 Group Policy Management Console (GPMC) ........................................... 24
       5.3.3 Local Group Policies ....................................................................................... 24
       5.3.4 Setting Account Policies for IIS Servers ........................................................... 25
       5.3.5 Setting Auditing Policies .................................................................................. 26

6. Securing the Web Server ................................................................................................. 27
   6.1 IIS 7.0 Design Principle 2: Reducing Attack Surface Area ....................................... 27
       6.1.1 Install the minimal required set of Web Server features .................................... 28
       6.1.2 Enable only the required Internet Server Application Programming Interface (ISAPI) filters .......................................................... 30
       6.1.3 Enable only the required ISAPI extensions ....................................................... 31
       6.1.4 Enable only the required Common Gateway Interface (CGI) applications .... 33
       6.1.5 Enable only the required FastCGI applications ............................................... 34

7. Securing the Web Application ......................................................................................... 35
   7.1 Reduce the attack surface area of the application ..................................................... 35
       7.1.1 Enable only the required modules ..................................................................... 35
       7.1.2 Configure the minimal set of application handler mappings ................................ 35
       7.1.3 Set Web Site permissions .................................................................................. 36
       7.1.4 Configure a minimal set of MIME types ......................................................... 38
   7.2 Configuring Applications for Least Privilege ............................................................ 39
7.2.1 Use a low privilege application pool identity........................................ 39
7.2.2 Set NTFS permissions to grant minimal access...................................... 41
7.2.3 Reduce trust of ASP.NET applications.................................................. 43
7.2.4 Isolate applications................................................................................ 44
7.3 Implementing Access Control.................................................................... 45
  7.3.1 IP and Domain Restrictions................................................................. 46
7.3.2 Request Filtering.................................................................................... 48
  7.3.2.1 Setting Request Limits..................................................................... 49
  7.3.2.2 Configuring Allowed Extensions.................................................... 50
  7.3.2.3 Configuring Hidden URL Segments............................................... 51
  7.3.2.4 Configuring Denied URL Sequences.............................................. 52
7.3.3 Authorization........................................................................................ 52
  7.3.3.1 NTFS ACL-based Authorization...................................................... 53
  7.3.3.2 URL Authorization......................................................................... 54
  7.3.3.3 Using URL Authorization to Restrict Access.................................... 54
  7.3.3.4 Creating URL Authorization Rules................................................ 55
  7.3.3.5 Using ASP.NET Roles with URL Authorization............................... 58
7.3.4 Authentication....................................................................................... 58
  7.3.4.1 Anonymous Authentication............................................................ 59
  7.3.4.2 Basic Authentication....................................................................... 61
  7.3.4.3 Digest Authentication....................................................................... 62
  7.3.4.4 Windows Authentication.................................................................. 63
  7.3.4.5 Configuring Windows Authentication.............................................. 64
  7.3.4.6 Client Certificate Mapping Authentication........................................ 67
  7.3.4.7 IIS Client Certificate Mapping Authentication.................................... 68
  7.3.4.8 Creating One-to-One Certificate Mappings....................................... 69
  7.3.4.9 Creating Many-to-One Certificate Mappings..................................... 70
  7.3.4.10 UNC Authentication...................................................................... 72
  7.3.4.11 Understanding Authentication Delegation...................................... 73
7.4 Securing Web Content............................................................................. 74
  7.4.1 Securing Active Content and Content Generation Technologies.......... 75
8. Using Encryption Technologies.................................................................... 77
  8.1 Securing Communications with Secure Socket Layer (SSL)....................... 77
  8.1.1 Configuring SSL................................................................................. 77
  8.1.2 Requiring SSL..................................................................................... 78
  8.1.3 Client Certificates................................................................................. 80
  8.2 Securing Configuration........................................................................... 80
  8.2.1 Restricting Access to Configuration..................................................... 81
   8.2.1.1 Setting Permissions on Configuration Files..................................... 82
   8.2.1.2 Understanding Configuration Isolation......................................... 82
   8.2.1.3 Setting Permissions for Shared Configuration................................. 83
  8.2.2 Securing Sensitive Configuration......................................................... 84
   8.2.2.1 Using Configuration Encryption to Store Configuration Secrets........ 84
   8.2.2.2 Selecting Encryption Providers.................................................... 85
   8.2.2.3 Limitations of Storing Secrets in Configuration............................... 87
   8.2.2.4 Limitations Access to Configuration from Managed Code in Partial Trust Environments................................................................. 87
  8.2.3 Controlling Configuration Delegation.................................................. 88
   8.2.3.1 Controlling which Configuration is Delegated............................... 88
9. Implementing a Secure Network Infrastructure........................................ 91
  9.1 Threat Modeling.................................................................................. 94
    9.1.1 Application.................................................................................. 94
    9.1.2 IT Infrastructure.......................................................................... 94

10. IIS 7.0 logging......................................................................................... 96
   10.1 IIS Manager..................................................................................... 96
   10.2 The XML-Based Logging Schema.................................................. 97
   10.3 Centralized Logging Configuration Options..................................... 99
   10.4 SiteDefaults Configuration Options................................................ 99
   10.5 Disable HTTP Logging Configuration Options.................................. 99
   10.6 Default Log File Location............................................................... 100
   10.7 Default UTF-8 Encoding............................................................... 100
   10.8 New Status Codes.......................................................................... 100
   10.9 Management Service....................................................................... 100
   10.10 Log File Formats That Have Not Changed..................................... 101
   10.11 Centralized Logging...................................................................... 101
     10.11.1 W3C Centralized Logging Format............................................ 101
     10.11.2 Centralized Binary Logging Format........................................ 101
   10.12 Remote Logging............................................................................. 102
     10.12.1 Setting Up Remote Logging by Using the IIS Manager............... 102
     10.12.2 Setting Up Remote Logging by Using Appcmd.......................... 103
     10.12.3 Remote Logging Using the FTP 7.0 Publishing Service................ 103
     10.12.4 Custom Logging..................................................................... 104
   10.13 Configuring IIS Logging................................................................. 104
     10.13.1 IIS Manager.......................................................................... 104
     10.13.2 Appcmd................................................................................. 107
   10.14 HTTP.sys Logging.......................................................................... 108
   10.15 Application Logging....................................................................... 109
     10.15.1 Process Recycling Logging..................................................... 109
     10.15.2 ASP....................................................................................... 109
     10.15.3 ASP.NET............................................................................... 110
     10.15.4 IIS Events............................................................................. 110
   10.16 Folder Compression Option............................................................ 110

11. Administering the Web Server.............................................................. 111
   11.1 Analyzing the log files................................................................. 111
   11.2 Web Server Backup....................................................................... 113
   11.3 Maintain a Test Web Server......................................................... 113
   11.4 Maintain an Authoritative copy of Organizational Web Content.... 113
   11.5 Recovering from a Security Compromise.................................... 114
   11.6 Security Testing Web Servers...................................................... 114
   11.7 Remotely Administering a Web Server........................................ 114

12. References............................................................................................ 116

Appendix A: Web Server Security Checklist............................................. 117
  i. Planning and Managing Web Servers............................................... 117
  ii. Securing the Web Server Operating System.................................... 118
  iii. Securing the Web Server................................................................ 119
  iv. Securing Web Content.................................................................. 121
  v. Using Authentication and Encrypting Technologies for Web Servers.. 123
vi. Implementing a Secure Network Infrastructure........................................................... 124
vii. Administering the Web Server.................................................................................. 125

Appendix B: Function of Built-In Modules with Security Implications............................... 127
1. Introduction

This guide is a collaborative effort of Indian Computer Emergency Response Team (CERT-In) and Microsoft, India.

1.1 Purpose and Scope

The purpose of this guide is to recommend security practices for designing, implementing and operating publicly accessible Microsoft Internet Information Services (IIS) 7.0 Web servers, including related network infrastructure issues.

The practices recommended in this document are designed to help mitigate the risks associated with public Web Servers.

This document may be used by the organizations interested in enhancing security on existing and future Web server systems to reduce web-related security incidents. The recommended settings in this guide are indicative and may change according to the specific requirements in which the web server is running. Thus some organizations might need to beyond these recommendations or adapt them in other ways to meet their unique requirements. It should be noted that the recommended settings are applicable when genuine software is used.

1.2 Audience and Assumptions

This document, while technical in nature, provides the background information to help readers understand the topics that are discussed. The intended audience for this document includes but not limited to System Engineers and Architects, Web and System Administrators, Webmasters, Security consultants, Information Technology (IT) security officers.

The practices recommended here are designed to help Web server administrators configure and deploy Web servers that satisfy their organizations’ security requirements. Web server administrators managing existing Web servers should confirm that their systems address the issues discussed.

As stated earlier, the purpose of this guide is to recommend security practices for designing, implementing and operating publicly accessible Microsoft Internet Information Services (IIS) 7.0 Web servers. It is presumed that the readers already know the installation process and administration practices; as these topics are beyond the scope of this guide.
2. Background

The Web server is a public face of any organization and one of the most important ways for an organization to publish information. However, if an organization is not rigorous in configuring and operating its public Web site, it may be vulnerable to a variety of security threats. Organizations may face monetary losses, damage to reputation or legal action if an intruder compromised their Web server. In addition, an organization finds itself in an embarrassing situation resulting from malicious intruders changing the content of the organization’s Web pages.

2.1 Web Site Security Issues

Kossakowski and Allen [Ref-05] identified three main security issues related to the operation of a publicly accessible Web site:

1. Misconfiguration or other improper operation of the Web server, which may result, for example, in the disclosure or alteration of proprietary or sensitive information. This information can include items such as the following:
   - Assets of the organization
   - Configuration of the server or network that could be exploited for subsequent attacks
   - Information regarding the users or administrator(s) of the Web server, including their passwords.

2. Vulnerabilities within the Web server that might allow, for example, attackers to compromise the security of the server and other hosts on the organization’s network by taking actions such as the following:
   - Defacing the Web site or otherwise affect information integrity
   - Executing unauthorized commands or programs on the host operating system, including ones that the intruder has installed
   - Gaining unauthorized access to resources elsewhere in the organization’s computer network
   - Launching attacks on external sites from the Web server, thus concealing the intruders’ identities, and perhaps making the organization liable for damages
   - Using the server as a distribution point for illegally copied software, attack tools, or pornography, perhaps making the organization liable for damages
   - Using the server to deliver attacks against vulnerable Web clients to compromise them.

3. Inadequate or unavailable defense mechanisms for the Web server to prevent certain classes of attacks, such as DoS attacks, which disrupt the availability of the Web server and prevent authorized users from accessing the Web site when required.

In addition, poorly written software applications and scripts allow attackers to compromise the security of the Web server or collect data from backend databases have become the targets of attacks. Many dynamic Web applications do not perform sufficient validation of user input, allowing attackers to submit commands that are run on the server. Common examples of this form of attack are structured query language (SQL) injection and cross-site scripting.
2.2 Security of a Web Server

A secure Web Server configuration plays a critical role in the overall security of an organization’s hosting environment. The security of a Web Server depends on the following:

- Security of underlying Operating System of Web Server
- Security of Web Server software
- Security of Web Server Applications/Content
- Security of the network where Web Server is operating

2.3 Steps required for securing any public Web Server

As a prerequisite for taking any step, it is essential that the organization should have a security policy in place. Taking the following steps within the context of the organization’s security policy should prove effective:

1. Installing, configuring, and securing the underlying operating system (OS)
2. Installing, configuring, and securing Web server software
3. Employing appropriate network protection mechanisms (e.g., firewall, packet filtering router, and proxy)
4. Ensuring that any applications developed specifically for the Web server are coded following secure programming practices
5. Maintaining the secure configuration through application of appropriate patches and upgrades, security testing, monitoring of logs, and backups of data and OS
6. Using, publicizing, and protecting information and data in a careful and systemic manner
7. Employing secure administration and maintenance processes (including server/application updating and log reviews)
8. Conducting initial and periodic vulnerability scans of each public Web server and supporting network infrastructure (e.g., firewalls, routers).
3. Planning and Managing Web Servers

Careful planning prior to installation, configuration and deployment is one of the most critical aspects of securing a Web Server. This will ensure that Web Server is as secure as possible and in compliance with all relevant organizational policies.

Security should be considered from the initial planning stage at the beginning of the systems deployment. It is much more difficult and expensive to address security after deployment and implementation.

It is utmost important to have a security policy of an organization in place. The security policy of an organization outlines and directs the implementation and management of network security. In addition, it confirms that organizations make efforts to be in compliance with relevant laws and regulations.

The basic steps should be followed in the planning stages of a Web Server are as follows:

- Identify the purpose(s) of the Web Server.
- Identify the network services that will be provided on the Web Server.
- Identify any network service software, both client and server, to be installed on the Web Server and any other support servers.
- Identify security requirements of information.
- Identify how the Web Server will be managed.
- Identify appropriate Operating System for Web Server.
- Develop secure programming practices for Web applications.
- Identify configuration/change control and management for Web Server.
- Identify contingency, continuity of operations and Disaster Recovery planning.
- Identify responsible personnel for managing Web Server.

Note: For comprehensive checklist, see “Appendix-A: Web Server Security Checklist” provided by NIST

3.1 Web Server Platforms

One of the most important stages in ‘Planning and Management of Web Server’ is the selection of Web Server platform. Although many organizations manage Web servers that operate over general-purpose Operating Systems, there are instances in which an organization may wish to use alternatives. These alternatives are based on sound technologies and have started to see broader use in the Web server environment. These alternatives are discussed in following sections.

- **Trusted Operating Systems (TOS):** These are security-modified or enhanced Operating Systems that include additional security mechanisms not found in most general-purpose Operating Systems. They are generally used in applications for which security is paramount. They can securely control all aspects of computing environment, including networking resources, users, processes and memory. Using TOS will generally produce a very secure Web server; however, a major drawback is that configuring and administering a TOS requires knowledge of each protected subsystem and its access needs. It may also require significant planning and administrative overhead to design and support a complex Web site on a TOS. Nevertheless, even with these limitations, organizations that have very high security requirements should consider using TOS on their Web servers. They were originally created to meet a particular set of

---

1 NIST – National Institute of Standards and Technology, US Department of Commerce
government requirements. Examples of certified Trusted Operating Systems are: HP-UX 10.26, PitBull for AIX 5L, Trusted Solaris, Trusted UNICOS 8.0 (Rated B1) and XTS-400.

- **Web Server Appliances:** A Web server appliance is a software/hardware combination that is designed to be a “plug-and-play” Web server. These appliances employ the use of a simplified OS that is optimized to support a Web server. The simplified OS improves security by minimizing unnecessary features, services, and options. The Web server application on these systems is often pre-hardened and pre-configured for security. Performance is often enhanced because the system (i.e., OS, Web server application, and hardware) is designed and built specifically to operate as a Web server. Cost is often reduced because hardware and software not specifically required by a Web server are not included. The greatest weakness in these systems is that they may not be suitable for large, complex, and multi-layered Web sites. They may be limited in what types of active content they support (e.g., J2EE, .NET, PHP Hypertext Preprocessor [PHP]), potentially reducing the options available to an organization. An appliance may host the back-end database as well as the front-end Web interface, potentially preventing organizations from having separate servers for each. Finally, it may be difficult to configure appliances from different manufacturers to work together. Nevertheless, because they offer a secure environment and an easy-to-configure interface, small- to medium-size organizations may find appliances an attractive option requiring less administrative effort. Web server appliances are available from most major hardware manufacturers and from various specialized manufacturers that concentrate solely on Web server appliances. Examples of such appliances are: Sun Cobalt RaQ, Zeus Web Server Appliance, Strongbolt server appliance etc.

- **Pre-Hardened Operating Systems and Web Servers:** There are some software distributions that include modified Operating System and Web Server application which are pre-configured to provide high security. These packages often include a greater number of security options and are designed to be easier to configure through the use of precompiled scripts and graphical user interfaces (GUI). Since the underlying Operating System is pre-configured specifically to run Web Server application, additional services cannot be configured. For example, Microsoft Windows 2003 Web Edition, Windows Web Server 2008 and Windows Web Server 2008 R2.

- **Virtualized Platforms:** Virtual machine technology is being used increasingly for Web servers. Through virtualization, a single physical host computer can run multiple virtual machines, each with a distinct guest OS and associated applications. New versions of mainstream OSs are being designed with virtualization in mind and new x86 64-bit processors provide hardware-level support for virtualization. Virtualization allows organizations to reduce costs by running multiple Web servers on a single host computer and by providing a mechanism for quickly responding to attacks against a Web server. Organization have to choose between three main types of virtual machine technology that exists; namely Full Virtualization, Native Virtualization and Paravirtualization. It is important to note that some virtualization software may be a hybrid implementation, depending on the hardware and guest OSs. Virtualization adds another layer of complexity to the Web server setup. Both the host OS and guest OS need to be secured. There is an overhead while using virtualization technology; each guest OS and associated Web server software should be configured and maintained individually. The Web server and its guest OS, the host OS, and the virtualization software should all be patched in a timely manner. If the guest OS or applications become compromised, the guest virtual machine can infect other hosts on the network as if it were a standalone physical host. Examples of virtualization software are: VMWare, Virtual PC, Hyper-V, Virtual Box and Xen etc.

---

4. Security and IIS 7.0

4.1 IIS 7.0 Design Principles

Following design principles have been followed to improve the security of IIS:

1. **Secure by default design**: IIS 7.0 is the outcome of Security Development Lifecycle (SDL)\(^3\) software development process which has been adopted by Microsoft to reduce the number of security-related design and coding defects, and to reduce the severity of any defects that are left. This follows Microsoft-cited motto, "Secure by Design, Secure by Default, Secure in Deployment and Communication" (also known as SD3+C).

2. **Reduce attack surface area**: IIS 7.0 gives an ability to reduce the attack surface area of the Web server through its modular architecture by enabling to remove all functionality other than what is absolutely necessary to host the web application. By leveraging this ability, organizations can deploy low-footprint Web servers with minimal possible surface area. After installing the minimal set of features, organizations can further reduce the surface area of the Web server by configuring the web application to operate with the minimal functionality, for example, configuring which application resources should be served.

3. **Use least privileges**: Practicing this principle minimizes the risk of exposed application vulnerabilities being successfully exploited by an attacker.

4.2 IIS 7.0 Design Principle 1: Secure by default design

As discussed earlier, IIS 7.0 is the outcome of Security Development Lifecycle (SDL) software development process which has been adopted by Microsoft to reduce the number of security-related design and coding defects, and to reduce the severity of any defects that are left. Following sections will discuss some of key features and security changes introduced in IIS 7.0.

4.2.1 Key Features introduced in IIS 7.0

IIS 7.0 delivers many new features and functionality based on the following key enhancements:

- **Modularity**: IIS 7.0 architecture is fully componentized. It enables administrators to customize which features are installed and running on the Web server. With more than 40 feature modules that can be independently installed, administrators can reduce the potential attack surface and lower the footprint requirements of the server.

- **Extensibility**: The core Web server features of IIS 7.0 have been built using a new set of comprehensive public APIs that developers can use to extend, replace, or add functionality to a Web server. These APIs are available as native Win32 APIs as well as managed .NET Framework APIs. Developers can also extend IIS configuration and build IIS Manager Extensions that plug in seamlessly to the management console.

- **Unified distributed configuration system**: IIS 7.0 provides a unified distributed file-based configuration system for storing all IIS and ASP.NET settings in a single clear-text XML format in a configuration files hierarchy where configuration files are stored together with Web site and application content. This

---

configuration system enables xcopy deployment of configuration alongside application code and content, and it also provides an easy way to share a configuration across a Web farm.

- **New administration tools**: IIS 7.0 offers a set of administration tools that simplify managing Web infrastructure and allow administrators to delegate administrative control for sites and applications to developers and content owners. IIS 7.0 includes a new GUI management console, IIS Manager; a new command line utility, Appcmd.exe; a new WMI provider for automating administration tasks; and a new managed API. All of these tools provide unified support for managing IIS and ASP.NET settings together. Administrators and developers can also use Windows PowerShell for scripting access to configuration information for the entire Web platform.

- **Integrated diagnostics**: IIS 7.0 enables administrators and developers to minimize downtime by using new diagnostics and troubleshooting capabilities. IIS 7.0 exposes run-time diagnostic information including currently executing requests. IIS 7.0 can also be configured to automatically log detailed trace events for failed requests for errant Web sites and applications.

### 4.2.2 Security Changes in IIS 7.0

IIS 7.0 builds on much of the IIS 6.0 feature codebase and secure practices formulated during the IIS 6.0 development life cycle. The majority of the core security principles and features established in IIS 6.0 are still in use today. However, IIS 7.0 does introduce improvements to help enhance the security of the Web server:

- **The anonymous user configured by default for anonymous authentication is the new built-in IUSR account**. This account is built-in and does not require a password that needs to be renewed and synchronized between servers. Additionally, permissions set for IUSR accounts are effective when copied to another IIS 7.0 server because the IUSR account has a well-known Security Identifier (SID) that is the same on every computer.

- **The IIS_WPG group has been replaced with the built-in IIS_IUSRS group**. This group is built-in and enables permissions set for IIS_IUSRS to remain effective when copied to another IIS 7.0 server because it has a well-known Security Identifier (SID). In addition, this SID is automatically injected into the worker process token for each IIS worker process, eliminating the need for manual group membership for any custom application pool identities.

- **Anonymous authentication can be configured to use the application pool identity**. This enables the content to require permissions only for the application pool identity when using anonymous authentication, simplifying permission management.

- **IIS worker processes automatically receive a unique application pool Security Identifier (SID) that administrator can use to grant access to the specific application pool to enable application isolation**.

- **Configuration isolation automatically isolates server-level configuration for each application pool**. The global server-level configuration contained in ‘applicationHost.config’ is automatically isolated by creating filtered copies of this configuration for each application pool and preventing other applications pools from being able to read this configuration.

- **Virtual directories can specify fixed credentials regardless of whether they point to Universal Naming Convention (UNC) shares or a local file system**. Unlike IIS 6.0, which supports fixed credentials for specifying access to UNC shares only, IIS 7.0 enables fixed credentials to be used for any virtual directory.

- **Windows Authentication is performed in the kernel by default**. This improves the configurability of the Kerberos protocol on the server. It also improves the performance of Windows Authentication. However, it may affect some applications that have custom clients that pre-send authentication credentials on the first request. This behavior can be turned off in the configuration.

- **The new Request Filtering feature provides extended URL Scan functionality**. Administrators can use the new Request Filtering feature to protect their Web server against nonstandard or malicious request patterns and additionally protect specific resources and directories from being accessed.
The new URL Authorization feature enables applications to control access to resources through configuration-based rules. The new URL Authorization feature provides flexible configuration-based rules to control access to application resources in terms of users and roles, and it supports the use of the ASP.NET Roles service.

Additionally, IIS 7.0 introduces several changes to existing security features and removes several deprecated security features that could impact web application. These changes to security-related features are listed here:

- **IIS 6.0 Digest Authentication is no longer supported.** It is being replaced by Advanced Digest Authentication (now simply referred to as Digest Authentication), which does not require the application pool to run with LocalSystem privileges.

- **.NET Passport Authentication is no longer supported.** The .NET Passport Authentication support is not included in Windows Vista and Windows Server 2008, and therefore IIS 7.0 does not support it.

- **IIS 6.0 URL Authorization is no longer supported.** The IIS 6.0 URL Authorization was overly complex and not often used. It has been replaced by the new configuration-based URL Authorization feature.

- **IIS 6.0 Sub-Authentication is no longer supported.** The Sub-Authentication feature enabled IIS 6.0 Digest Authentication (which has been removed) and synchronized anonymous account passwords (the anonymous account now uses the new built-in IUSR account that does not have a password). It is no longer needed in IIS 7.0 and therefore has been retired.

- **IIS Manager no longer provides support for configuring IIS Client Certificate Mapping Authentication.** Use Appcmd from the command line, or use another configuration application programming interface (API) to configure this feature.

- **Several authentication and impersonation differences exist in ASP.NET applications when running in the default Integrated mode.** This includes an inability to use both Forms authentication and an IIS authentication method simultaneously, and an inability to impersonate the authenticated user in certain stages of request processing.

- **Metabase access control lists (ACLs) are no longer supported.** With the new configuration system, the permissions are not set on individual configuration settings. IIS 7.0 provides built-in support for delegating configuration settings to Web site and application owners, replacing metabase ACLs as a mechanism for configuration delegation.

- **Metabase auditing is no longer supported.** The ability to audit changes to specific configuration settings is not supported out of the box. This is a consequence of IIS 7.0 not supporting metabase ACLs.
5. Securing the Web Server Operating System

Many security issues can be avoided if the Operating System underlying the Web Servers is hardened properly. Default hardware and software configurations are typically set by manufacturers to emphasize features, functions, and ease of use. Because manufacturers are unaware of each organization’s security needs, each Web server administrator must configure new servers to reflect their organization’s security requirements and reconfigure them as those requirements change.

Security configuration guides, guidelines and checklists for different Operating Systems are publicly available by their respective OEMs. These documents typically contain recommendations for settings that improve the default level of security, and they may also contain step-by-step instructions for securing systems.

Note: Readers are encouraged to follow the security guides/guidelines for specific platforms either provided by respective OEMs or CERT-In. It is also recommended that the organizations should maintain their own guidelines, specific to their requirements and security policies in-line with industry’s security best practices.

The techniques for hardening different Operating Systems vary greatly; therefore, this section includes the generic procedures common in securing most Operating Systems. [For specific configurations, refer to the respective OEMs’ documentation]

Following five basic steps are necessary to maintain basic OS security (it should be in-line with organizational security policy):

- Planning the installation and deployment of the host Operating System and other components for the Web server
- Patching and updating the host Operating System as required
- Hardening and configuring the host Operating System to address security adequately
- Installing and configuring additional security controls
- Testing the host Operating System to ensure that the previous four steps adequately addressed all security issues

Note: For comprehensive checklist, see “Appendix-A: Web Server Security Checklist”

Some automated tools and security templates also exist for hardening Operating Systems, and their use is strongly recommended.

5.1 Managing Windows Security

In purview of IIS; Windows security, IIS security, and Web Application security can be completely integrated. The integrated security model allows using authentication based on user and group membership in addition to standard Internet-based authentication. It also allows using a layered permission model to determine access rights and permissions for applications and content. Before users can access files and directories, ensure that the appropriate users and groups have access at the Operating System level. Then set IIS security permissions that grant permissions for content that IIS controls. Finally, use Web Application Profile, Users, and Roles to manage top-level access to managed code applications.
5.1.1 Working with User and Group Accounts

Administrators can set user and group accounts either at the local computer level or at the domain level. Local accounts are specific to an individual computer and aren't valid on other machines or in a domain unless specifically grant permissions. Domain accounts, on the other hand, are valid throughout a domain, which makes resources in the domain available to the account.

Typically, use specific accounts for specific purposes:

- Use local accounts when IIS servers aren't part of a domain or to limit access to a specific computer.
- Use domain accounts when the servers are part of a Windows domain and users to be able to access resources throughout that domain.

User accounts that are important on IIS servers include:

- **Local System**: By default, all standard IIS services log on using the local system account. This account is part of the Administrators group on the Web server and has all user rights on the Web server. If application pools are configured to use this account, the related worker processes have full access to the server system, which may present a serious security risk.
- **Local Service**: A limited-privilege account that grants access to the local system only. The account is part of the Users group on the Web server and has the same rights as the Network Service account, except that it is limited to the local computer. Configure application pools to use this account when worker processes don't need to access other servers.
- **Network Service**: By default, all applications log on using the network service account. When IIS is using out-of-process session state management, the ASP.NET State Service also uses this account by default. This account is part of the Users group on the Web server and provides fewer permissions and privileges than the Local System account (but more than the Local Service account). Specifically, processes running under this account can interact throughout a network by using the credentials of the computer account.
- **IUSR_ComputerName**: Internet guest account used by anonymous users to access Internet sites. The account grants anonymous users limited user rights and is also known as the anonymous user identity.
- **IIS_IUSRS group**: When IIS 7.0 is installed, the IIS_IUSRS group is also created. If a specific user identity for an application pool is used, Web Administrator must make that identity a member of the IIS_IUSRS group to ensure that the account has appropriate access to resources.

Following table provide details of key user rights assigned to IIS user and group accounts by default. Administrators can make changes to these accounts if necessary. For added security, administrator can configure IIS to use different accounts from the standard accounts provided or can also create additional accounts.
### Table 5.1: Important User Rights Assigned by Default to IIS User and Group Accounts

<table>
<thead>
<tr>
<th>Default User Right</th>
<th>Local Service</th>
<th>Network Service</th>
<th>IUSR</th>
<th>IIS_IUSRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access This Computer From The Network</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Adjust Memory Quotas For A Process</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Allow Log On Locally</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Bypass Traverse Checking</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Change The System Time</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change The Time Zone</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create Global Objects</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generate Security Audits</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Impersonate A Client After Authentication</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Increase Process Working Set</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Log On As A Batch Job</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Log On As A Service</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Replace A Process-Level Token</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 5.1.2 Managing the IIS Service Logon Accounts

The standard IIS services use the local system account to log on to the server. Using the local system account allows the services to run system processes and perform system-level tasks. It is advisable that this configuration should not be changed unless have very specific needs or want to have strict control over the IIS logon account's privileges and rights.

Following steps should be carried out to re-configure the logon account for an IIS service.

1. In the Computer Management console, in the left pane, connect to the IIS server whose services to be managed.
2. Expand the Services And Applications node by clicking the plus sign (+) next to it, and then choose Services.
3. In the right pane, right-click the service to be configured, and then choose Properties.
4. Click the Log On tab, as shown in Figure 5.1
5. Choose one of the following:
   - If the service should log on using the system account (the default for most services), select Local System Account.
   - If the service should log on using a specific user account, select This Account. Be sure to type an account name and password in the appropriate fields. Click the Browse button to search for a user account if necessary. [Note: If a specific user identity be used for a service, assign privileges and logon rights to the account to be used].
6. Click OK.
5.1.3 Managing the Internet Guest Account

Manage the Internet Guest account at the IIS security level and at the Windows security level. At the IIS security level, specify the user account to use for anonymous access. Normally, anonymous access is managed at the server or site level, and all related files and directories inherit the settings used. This behavior can be changed for individual files and directories as necessary.

To change the configuration of the anonymous user account for an entire server or another configuration level, complete the following steps:

1. In IIS Manager, navigate to the level of the configuration hierarchy to be managed, and then double-click Authentication.
2. On the Authentication page, in the main pane, click Anonymous Authentication, and then in the Actions pane, click Edit.
3. The IUSR_ComputerName account is the default Internet guest account. Choose one of the following based on the user account to specify:
   - If a different user account is to specify, select Specific User, and then click Set. In the Set Credentials dialog box, type the user name for the account. Type and then confirm the account password, and then click OK twice.
• If the application pool identity is to be used rather than a specific user account, select Application Pool Identity, and then click OK.

Note: When Anonymous Access is enabled, users don’t have to log on using a user name and password. IIS automatically logs the user on using the anonymous account information provided for the resource. If Anonymous Authentication isn’t listed as Enabled on the Authentication page, the resource is configured for named account access only. To enable anonymous access, click Anonymous Authentication, and then in the Actions pane, click Enable. However, this should be done when sure that the resource doesn’t need to be protected.

5.2 Working with File and Folder Permissions

Every folder and file used by IIS can have different access permissions which can be set at the Windows security level.

5.2.1 File and Folder Permission Essentials

The basic permissions which can be assigned to files and folders are summarized in Table 5.2. The basic permissions are created by combining special permissions, such as Traverse Folder and Execute File, into a single easily managed permission. If granular control is needed over file or folder access, use advanced permissions to assign special permissions individually.

<table>
<thead>
<tr>
<th>Permission</th>
<th>Significance to Folders</th>
<th>Significance to Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>Permits viewing and listing files and subfolders</td>
<td>Permits viewing or accessing the file's contents</td>
</tr>
<tr>
<td>Write</td>
<td>Permits adding files and subfolders</td>
<td>Permits writing to a file</td>
</tr>
<tr>
<td>Read And Execute</td>
<td>Permits viewing and listing files and subfolders and executing files; inherited by files and folders</td>
<td>Permits viewing and accessing the file's contents and executing the file</td>
</tr>
<tr>
<td>List Folder Contents</td>
<td>Permits viewing and listing files and subfolders and executing files; inherited by folders only</td>
<td>N/A</td>
</tr>
<tr>
<td>Modify</td>
<td>Permits reading and writing of files and subfolders; allows deletion of the folder</td>
<td>Permits reading and writing of the file; allows deletion of the file</td>
</tr>
<tr>
<td>Full Control</td>
<td>Permits reading, writing, changing, and deleting files and subfolders</td>
<td>Permits reading, writing, changing, and deleting the file</td>
</tr>
</tbody>
</table>

Table 5.2: File and Folder Permissions Used by Windows Server

Keep the following in mind when working with file and folder permissions

• Read is the only permission needed to run scripts. Execute permission applies only to executables.
• Read access is required to access a shortcut and its target.
• Giving a user permission to write to a file but not to delete it doesn't prevent the user from deleting the file's contents. A user can still delete the contents.
• If a user has full control over a folder, the user can delete files in the folder regardless of the permission of the files.
IIS uses the following users and groups to configure file and folder access:

- **Administrators**: Allows administrators to access IIS resources.
- **Creator Owner**: Allows the account that created a resource to access the resource.
- **System**: Allows the local system to access the resource.
- **Users**: Allows named accounts to access the resource (including the Local Service and Network Service accounts, which are user accounts).
- **IIS_IUSRS**: Allows setting specific permission for special identities that are members of the IIS_IUSRS group. To prevent malicious users from gaining access to files and modifying them, deny this account Full Control, Modify, and Write permission on important directories.

When Read permission is granted to these users and groups, anyone who has access to Internet or intranet Web site will be able to access the files and folders. If access to certain files and folders is restricted, then set specific user and group permissions and use authenticated access rather than anonymous access. With authenticated access, IIS authenticates the user before granting access and then uses the Windows permissions to determine what files and folders the user can access.

Following table provides general guidelines for assigning permissions based on content type.

<table>
<thead>
<tr>
<th>File Type</th>
<th>File Extension</th>
<th>Permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGI scripts and executables</td>
<td>.exe, .dll, .cmd</td>
<td>Users (Execute), Administrators (Full Control), System (Full Control)</td>
</tr>
<tr>
<td>Dynamic content</td>
<td>.asp, .aspx, .vbs, .js, .pl</td>
<td>Users (Read Only), Administrators (Full Control), System (Full Control)</td>
</tr>
<tr>
<td>Include files</td>
<td>.inc, .shtm, .shtml, .stm</td>
<td>Users (Read Only, Deny Write), Administrators (Full Control), System (Full Control)</td>
</tr>
<tr>
<td>Static content</td>
<td>.txt, .rtf, .gif, .jpg, .jpeg, .htm, .html, .doc, .ppt, .xls</td>
<td>Users (Read Only, Deny Write), Administrators (Full Control), System (Full Control)</td>
</tr>
</tbody>
</table>

Table 5.3: General Guidelines for Permissions Based on Content Type

**NOTE**: It is advisable that instead of setting permissions on individual files; organize the content by type in subdirectories and consider to specifically deny permission, such as Full Control, Modify, or Write.

5.2.2 Viewing File and Folder Permissions

Security permissions for files and folders can be viewed in Windows Explorer or in IIS Manager by completing the following steps:

1. Open *Windows Explorer* or *IIS Manager* as appropriate. In *Windows Explorer*, right-click the file or folder to work with, and then select *Properties*. In *IIS Manager*, navigate to the site node or folder node to work with, and then in the *Actions pane*, click *Edit Permissions*.
2. In the *Properties* dialog box for the file or folder previously selected. On the *General* tab, note applied NTFS attributes, such as Read only or Hidden, and change accordingly, if needed.
3. Select the Security tab. In the Group Or User Names list box, select the user, computer, or group to view their respective permissions. If check boxes in the Permissions For list are dimmed, it means that the permissions are inherited from a parent object.

5.2.3 Setting File and Folder Permissions

Set permissions for files and folders by completing the following steps:

1. Open Windows Explorer or IIS Manager as appropriate. In Windows Explorer, right-click the file or folder to work with, and then select Properties. In IIS Manager, navigate to the site node or folder node to work with, and then in the Actions pane, click Edit Permissions.
2. In the Properties dialog box, select the Security tab; select a user, computer, or group, and then click Edit. This displays an editable version of the Security tab, as shown in Figure 5.2.

![Figure 5.2: Use the Security tab to configure basic permissions for the file or folder](image)

3. Users or groups that already have access to the file or folder are listed in the Group Or User Names list box. Change permissions for these users and groups by doing the following:
   - Select the user or group.
   - Use the Permissions For list box to grant or deny access permissions.
Note: Inherited permissions are dimmed. If an inherited permission is needed to be override, select the opposite permission. For example, if, because of inheritance, a user is granted permission that user should not have, override the inheritance by explicitly denying the permission in the ‘Permissions For’ list box.

4. Click Add to set access permissions for additional users, contacts, computers, or groups. This displays the Select Users, Computers, Or Groups dialog box. [If the server is a part of an Active Directory Domain then only the members of that domain can select computer accounts and configure their permissions]
5. In the Select Users, Computers, Or Groups dialog box, select the users, computers, or groups to set access permissions, and then click OK.
6. In the Group Or User Names list box, select the user, computer, or group to configure, and then use the fields in the Permissions For list box to allow or deny permissions. Repeat for other users, computers, or groups.
7. Click OK when finished.

5.3 Enforcement of Security Configurations through Policies

Group policies are another aspect of Windows security that can be used to automate key security administration tasks and to manage IIS resources more effectively. It can be used to lock down the security configuration of Windows based systems. It can be used in conjunction with security templates to easily create and deploy custom configurations for locked-down systems.

There are two kinds of policies which can be used to enforce security configurations.
- Local Security Policy
- Group Policy

5.3.1 Local Security Policy

A local security policy is a combination of security settings that affect the security on a local computer. It contains the following types of security information:
- Which domains are trusted for authentication of logon attempts?
- Which user accounts are allowed to access the system and the way in which they can access it (interactively, through a network or as a service)
- The various rights and privileges assigned to user accounts
- The audit policy for the machine
- Password and account lockout restrictions

If local computer has joined a domain, it is subject to obtain security policy from the domain's policy or from the policy of any organizational unit that it is a member of. If the computer is getting policy from more than one source, then any conflicts are resolved in order of precedence.

5.3.2 Group Policy

Group policies are for sites, domains, and organizational units (OUs) which can be configured only for computer, group, and user accounts that are part of a domain. It is integrated with Active Directory (AD) directory service to simplify the configuration and management of systems across large networks and it includes configuration options

---

4 ‘Lock-Down’ is another name for hardening, configuring a host to make it more secure for a specific role.
5 Security Templates are *.inf files used for defining policy settings for securing different aspects of a Windows based systems.
for authentication methods, system auditing, event logging, password settings, registry access, IPSEC encryption and many other aspects of systems and network security.

Group policies provide central control over privileges, permissions, and capabilities of users and computers. A policy can be thought as a set of rules that can be applied to multiple computers and to multiple users. Because computers can be a part of larger organizational groups, multiple policies can be applied.

The order in which policies are applied is extremely important in determining which rules are enforced and which rules are not. When multiple policies are in place, the policies are applied in the following order:

1. Local group policies that affect the local computer only
2. Site group policies that affect all computers that are part of the same site, which can include multiple domains
3. Domain policies that affect all computers in a specific domain
4. Organizational unit policies that affect all computers in an organizational unit
5. Child organizational unit policies that affect all computers in a subcomponent of an organizational unit

As successive policies are applied, the rules in those policies override the rules set in the previous policy. For example, domain policy settings have precedence over the local Group Policy settings. Exceptions allow blocking, overriding, and disabling policy settings.

Two graphical user interface (GUI) tools are provided for managing Active Directory Group Policy:

- **Group Policy Object Editor** (basic editor): It enables to view, configure, and manage policy settings for Group Policy Objects in any forest and domain to which administrators can connect and have appropriate administrator permissions.
- **Group Policy Management Console** (GPMC— an advanced editor): Management features in Group Policy Management Console (GPMC) enables to import, export, back up, and restore GPOs. It can also be used to plan Group Policy changes and to determine how group policies are being applied to particular computers and users.

### 5.3.2.1 Group Policy Object Editor

To use the Group Policy Object Editor and related features to access and use site, domain, and OU policies, complete the following steps:

1. For sites, open the *Active Directory Sites and Services* console to create a GPO that is linked to the site. For domains and OUs, open the *Active Directory Users and Computers* console to create a GPO that is linked to the domain or OU.
2. In the left pane of the appropriate *Active Directory* window, to create or manage Group Policy right-click the appropriate *site, domain, or OU*. Then on the shortcut menu, select *Properties*. The *Properties* dialog box opens.
3. In the *Properties* dialog box, click the *Group Policy* tab:
   - **Create a new policy**: To create a new policy, click *New*. Type a *name* for the policy, and then press *Enter*. Then click *Edit* to configure the new policy.
   - **Edit an existing policy**: To edit an existing policy, select the *policy*, and then click *Edit* to edit the policy settings.
- **Change the priority of a policy:** To change the priority of a policy, click the *Up* or *Down* button to change its position in the *Group Policy Object Links* list.

### 5.3.2.2 Group Policy Management Console (GPMC)

The Group Policy Management Console (GPMC) is included with Windows 2000 Server and later releases of the Windows operating system. To use the GPMC and related features to access and work with site, domain, and OU policies, complete the following steps:

1. When the *Group Policy Management* feature is added using the *Add Feature Wizard*, the *Group Policy Management Console* is available on the *Administrative Tools* menu. Click *Start*, point to *Administrative Tools*, and then select *Group Policy Management*.
2. In the MMC, there are two top-level nodes: *Group Policy Management* (the label for the console root) and *Forest* (a node representing the currently connected forest). Expand the *Forest* node for the following nodes:
   - **Domains:** Provides access to the policy settings for domains in the related forest. By default, it is connected to logon domain and can add connections to other domains. Expand a domain to access Default Domain Policy, the Domain Controllers OU (and the related Default Domain Controllers Policy), and Group Policy Objects defined in the domain.
   - **Sites:** Provides access to the policy settings for sites in the forest. Sites are hidden by default.
   - **Group Policy Modeling:** Provides access to the Group Policy Modeling Wizard, which can be used to plan policy deployment and simulate settings for testing purposes. The wizard also provides access to any saved policy models.
   - **Group Policy Results:** Provides access to the Group Policy Results Wizard. For each connected domain, all the related Group Policy Objects and OUs are available to work with in one location.
3. Now:
   - **Create a new policy:** Right-click the *site, domain, or OU* to work with, and then select *Create And Link A GPO Here*. In the *New GPO* dialog box, type a *descriptive name* for the new GPO, and then click *OK*. The GPO is now created and linked to the *site, domain, or OU*. Right-click the *GPO*, and then choose *Edit*. This opens the *Group Policy Object Editor* to edit the policy settings.
   - **Edit an existing policy:** Expand the *site, domain, or OU* node in which the related policy is stored. Right-click the *policy*, and then choose *Edit*. This opens the *Group Policy Object Editor*.

### 5.3.3 Local Group Policies

Manage local group policies for an individual computer by completing the following steps:

1. Click *Start*, point to *All Programs*, and then point to *Accessories*.
2. Right-click *Command Prompt*, and then select *Run As Administrator*.
3. At the *command prompt*, type *mmc*. This opens an empty *Microsoft Management Console (MMC)*.
4. On the *File* menu, select *Add/Remove Snap-In*.
5. In the *Add Or Remove Snap-In* dialog box, under *Available Snap-Ins*, select *Local Group Policy Object Editor*, and then click *Add*. 
6. By default, the editor works with the local computer’s Group Policy Object (GPO), click Finish to accept this as the default.

7. Click OK and manage the local policy on the selected computer.

There is another way to start the Group Policy Object Editor for the local computer: On the Start menu, click Run, in the Run box type gpedit.msc, and then press Enter.

Group policies for passwords, account lockout, and auditing are essential to Web server’s security. Following sections provide the recommended settings for them and configuring steps. It is encouraged that these settings should be in-line with the organizational security policy and individual requirements.

<table>
<thead>
<tr>
<th>Guidelines for password policies</th>
<th>Recommended Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set a minimum password age for all accounts</td>
<td>2–3 days</td>
</tr>
<tr>
<td>Set a maximum password age for all accounts</td>
<td>30 days</td>
</tr>
<tr>
<td>Set a minimum password length</td>
<td>8 characters</td>
</tr>
<tr>
<td>Enable secure passwords by enforcing password complexity requirements</td>
<td></td>
</tr>
<tr>
<td>Enforce password history</td>
<td>5 or more</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Guidelines for account lockout policies</th>
<th>Recommended Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set an account lockout threshold</td>
<td>5 bad attempts</td>
</tr>
<tr>
<td>Set account lockout duration</td>
<td>Indefinitely [This requires Administrator’s intervention]</td>
</tr>
<tr>
<td>Reset the lockout threshold</td>
<td>after 30–60 minutes</td>
</tr>
</tbody>
</table>

5.3.4 Setting Account Policies for IIS Servers

Set account policies by completing the following steps:

1. Access the group policy container to work with. Expand the Computer Configuration node, then Windows Settings, then Security Settings, and finally, Account Policies to manage.

2. For domains, sites, and OUs, there are Password Policy, Account Lockout Policy, and Kerberos Policy nodes. For local computers, there are Password Policy and Account Lockout Policy nodes only.

3. To configure a policy, double-click its entry or right-click it and select Properties. This opens a Properties dialog box for the policy. Then do one of the following:
   - For a local policy, the Properties dialog box will be different from that for a site, domain, or OU. Use the appropriate fields to configure the local policy. Skip the remaining steps; they apply to global group policies.
   - For a site, domain, or OU, all policies are either defined or not defined—that is, they are either configured for use or not configured for use. A policy that isn't defined in the current container could be inherited from another container.

4. Select or clear the Define This Policy Setting check box to determine whether a policy is defined.

5. Policies can have additional fields; often, these fields have the following option buttons:
   - **Enabled** Turns on the policy restriction
   - **Disabled** Turns off the policy restriction
5.3.5 Setting Auditing Policies

Auditing is the best way to track what's happening on IIS server. Auditing can be used to collect information related to resource usage, such as file access, system logon, and system configuration changes. Whenever an action occurs that is configured for auditing, the action is written to the system's security log, where it's stored for review and can be accessed the security log from Windows Event Viewer.

Set auditing policies by completing the following steps:

2. Following auditing options are accessible now:
   - **Audit Account Logon Events**: Tracks events related to user logon and logoff.
   - **Audit Account Management**: Tracks account management. Events are generated anytime user, computer, or group accounts are created, modified, or deleted.
   - **Audit Directory Service Access**: Tracks access to the Active Directory service. Events are generated whenever users or computers access the directory.
   - **Audit Logon Events**: Tracks events related to user logon, logoff, and remote connections to network systems.
   - **Audit Object Access**: Tracks system resource usage for files, directories, shares, printers, and Active Directory objects.
   - **Audit Policy Change**: Tracks changes to user rights, auditing, and trust relationships.
   - **Audit Privilege Use**: Tracks the use of user rights and privileges, such as the right to back up files and directories, but doesn't track system logon or logoff.
   - **Audit Process Tracking**: Tracks system processes and the resources they use.
   - **Audit System Events**: Tracks system startup, shutdown, and restart, in addition to actions that affect system security or the security log.
3. To configure an auditing policy, double-click its entry or right-click it and select Properties. This opens a Properties dialog box for the policy.
4. Select Define These Policy Settings, and then select either the Success check box or the Failure check box, or both. Success logs successful events, such as successful logon attempts. Failure logs failed events, such as failed logon attempts.
5. Click OK to finish.
6. Securing the Web Server

After installing the underlying Operating System and following best practices to secure it; next step is to install the Web Server software and Web Application following the best practices to secure it.

The overarching principle for securely installing the Web Server is to install only the services required for the Web server and to eliminate any known vulnerabilities through patches or upgrades. Any unnecessary applications, services, or scripts that are installed should be removed immediately once the installation process is complete. During the installation of the Web server, the following steps should be performed:

- Install the Web server software either on a dedicated host or on a dedicated guest OS if virtualization is being employed.
- Apply any patches or upgrades to correct for known vulnerabilities.
- Create a dedicated physical disk or logical partition (separate from OS and Web server application) for Web content.
- Remove or disable all services installed by the Web server application but not required (e.g., gopher, FTP, remote administration).
- Remove or disable all unneeded default login accounts created by the Web server installation.
- Remove all manufacturers’ documentation from the server.
- Remove all example or test files from the server, including scripts and executable code.
- Apply appropriate security template or hardening script to server.
- Reconfigure HTTP service banner (and others as required) not to report Web server and Operating System type and version (if possible).
- Consider installing the Web server with non-standard directory names, directory locations, and filenames. [This increases the likelihood of attack detection because of the failed attempts to access the default filenames and directories and the additional time needed to perform an attack.]

6.1 IIS 7.0 Design Principle 2: Reduce Attack Surface Area

As discussed earlier in section 4.1, IIS 7.0 gives an ability to reduce the attack surface area of the Web server through its modular architecture by enabling to remove all functionality other than what is absolutely necessary to host the web application.

Reducing the attack surface area of the Web server is a key strategy in reducing the risk of a security vulnerability being successfully exploited by an attacker. It provides the following benefits:

- It directly reduces the number of features and services exposed by the Web server to outside clients, minimizing the amount of code available for an attacker to exploit.
- It reduces complexity, which makes it easier to configure the Web server in a secure manner.
- If a vulnerability is exposed, the uptime of the Web server is not affected as much, because if the component affected by the vulnerability is not installed, it is not necessary to take the Web server offline or patch it immediately.
The process for reducing the attack surface area of the IIS 7.0 Web server includes the following steps:

- Install the minimal required set of Web server features.
- Enable only the required Internet Server Application Programming Interface (ISAPI) filters.
- Enable only the required ISAPI extensions.
- Enable only the required Common Gateway Interface (CGI) applications.
- Enable only the required FastCGI applications.

### 6.1.1 Install the minimal required set of Web server features

The IIS 7.0 modular feature set comprises more than 40 individual Web server modules that provide various request processing and application services. The Web server core engine retains only the minimal set of functionality needed to receive the request and dispatch it’s processing to modules. This architecture can be leveraged to deploy minimal surface area Web servers by installing only the modules that are required for the Web server’s operation.

The modular feature set provided in IIS 7.0 is fully integrated with Windows Setup. This means that most of the IIS 7.0 modules can be installed or uninstalled by installing IIS 7.0 features directly from the Web Server (IIS) role in Server Manager on Windows Server 2008 as shown in Figure 6.1. Each feature typically corresponds to one module (or in some cases several modules) and installs any corresponding configuration information as well as feature dependencies.

The default installation of IIS 7.0 includes only the features necessary for IIS 7.0 to function as a static file Web server. In many cases, this may not be sufficient to properly host the Web application, so install additional features as needed, including support for hosting dynamic application technologies. While doing this the setup will prompt for installing any dependencies of the feature and configure the proper default configuration for that feature.
**Figure 6.1: Installing IIS 7.0 using the Add Roles Wizard**

**Note:**
*Do not install all the IIS 7.0 features if unsure of requirements. Doing so can unnecessarily increase the surface area of the Web server.*

Take the following steps to reduce the surface area of the Web server:

- Determine the set of features that Web applications need.
- Install only the required features.
- Install only the required third-party modules. [Exercise caution when installing any module on the Web server and make sure that it’s from trusted source. Installing untrusted or buggy modules can compromise the security of the Web server or negatively affect its reliability and performance.]
- Test the application properly for correct operation.
6.1.2 Enable only the required Internet Server Application Programming Interface (ISAPI) filters.

ISAPI filters allow third parties to extend IIS request processing. IIS 7.0 replaces ISAPI filters with IIS 7.0 modules as the preferred mechanism for extending the Web server. However, IIS 7.0 continues to support ISAPI filters for backward compatibility reasons.

Note:
- To enable ISAPI filters to work on IIS 7.0, the ISAPI Filters role service must be installed. This role service installs the IsapiFilterModule module, which provides support for hosting ISAPI filters. If this module is removed, ISAPI filters will not be loaded. This role service is not enabled by default; it is however enabled when the ASP.NET role service is installed.
- To enable ISAPI filters server administrator privileges are needed.

If the Web server uses ISAPI filters, to minimize the Web server attack surface area; ensure that only the required ISAPI filters should be enabled. Following steps should be taken to properly configure ISAPI filters:

1. **If Web server uses ISAPI filters, install the ISAPI Filters role service.** Without this role service, the ISAPI filters will not be loaded and therefore may create a security risk if they are responsible for security-sensitive functionality.
2. **If Web server does not use ISAPI filters, do not install the ISAPI Filters role service.** This eliminates the possibility of unwanted ISAPI filters being configured on the server.
3. **Determine the ISAPI filters that the application requires.** In the majority of cases, Web server should not require any ISAPI filters (with the exception of the ASP.NET ISAPI filter). Therefore, typically ISAPI filters are needed to be configured if migrating an existing application from previous versions of IIS that require specific ISAPI filters, or if installing a new third-party ISAPI filter.
4. **Enable the required ISAPI filters.** By using IIS Manager for a specific Web site, which ISAPI filters are enabled on the server can be controlled.

*Note: IIS 7.0 does not install any ISAPI filters by default. However, ASP.NET 1.1 and ASP.NET 2.0 will install an ISAPI filter named ASP.NET_2.0.50727.0. This filter is required for cookie-less ASP.NET features to work properly. Do not remove this filter.*

To use IIS Manager to configure the ISAPI filters, click the Web server node or Web site node in the tree view and then double-click ISAPI Filters, as shown in Figure 6.2. Exercise extreme caution when installing third-party ISAPI filters and be sure their source is trusted.
Administrators can also control which filters are enabled for the Web server or for a specific Web site by editing the system.webServer/isapiFilters configuration section directly, with the Appcmd command line tool, or with another configuration API.

6.1.3 Enable only the required ISAPI extensions.

ISAPI extensions allow third parties to extend IIS request processing by returning responses for specific content types. IIS 7.0 replaces ISAPI extensions with IIS 7.0 modules as a preferred mechanism for extending IIS. However, IIS 7.0 continues to support ISAPI extensions for backward compatibility reasons.

Note:

a) To enable ISAPI extensions to work on IIS 7.0, the ISAPI Extensions role service must be installed. This role service installs the IsapiModule module, which provides support for hosting ISAPI extensions. If this module is removed, ISAPI extensions will not be loaded. This role service is not enabled by default, but it is enabled when ASP.NET is installed.

b) To enable ISAPI extensions server administrator privileges are needed.
Dynamic application framework technologies frequently use ISAPI extensions to interface with IIS. For example, both ASP.NET (for Classic mode applications) and ASP are implemented as ISAPI extensions. If the Web server uses ISAPI extensions, to minimize the Web server attack surface area only the required ISAPI extensions should be enabled. Following steps should be taken to properly configure ISAPI extensions:

1. **If the Web server uses ISAPI extensions, install the ISAPI Extensions role service.** Without this role service, the ISAPI extensions will not be loaded, and requests to resources mapped to ISAPI extensions will return errors.

2. **If the Web server does not use ISAPI extensions, do not install the ISAPI Extensions role service.** This eliminates the possibility of unwanted ISAPI extensions being configured on server.

3. **Configure the allowed ISAPI extensions.** Each ISAPI extension must be allowed to execute on the server before it can be used. Use IIS Manager to configure all ISAPI extensions that are allowed to execute on the server. Exercise extreme caution when allowing third-party ISAPI extensions and be sure to trust their source. Installing untrusted or buggy ISAPI extensions can compromise the security of the Web server or negatively affect its reliability.

4. **Configure the desired handler mappings.** To use ISAPI extensions, handler mappings are required to be created that map allowed ISAPI extensions to specific content types in the application.

Table 6.1: specifies the common ISAPI extensions and when they are installed.

<table>
<thead>
<tr>
<th>ISAPI Extension</th>
<th>Default State</th>
<th>When Installed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Server Pages</td>
<td>Allowed</td>
<td>ASP role service is installed</td>
</tr>
<tr>
<td>ASP.NET v1.1.4322</td>
<td>Not Allowed</td>
<td>.NET Framework v1.1 SP1 is installed</td>
</tr>
<tr>
<td>ASP.NET v2.0.50727</td>
<td>Allowed</td>
<td>ASP.NET role service is installed</td>
</tr>
</tbody>
</table>

Table 6.1: Common ISAPI Extensions

On IIS 7.0, the ISAPI extensions are automatically allowed when the corresponding role services is being installed. Additionally, only ASP.NET applications running in Classic mode use the ASP.NET 2.0 ISAPI extension. It is a more reliable practice to use the roles or features wizards to control the availability of these features, instead of allowing or not allowing them in the ISAPI and CGI Restrictions. However, the ISAPI extension for ASP.NET v1.1, still be enabled manually.

On IIS 7.0, use IIS Manager to allow or deny an ISAPI extension in the Web Service Extension Restriction List by clicking the Web server node in the tree view and then double-clicking ISAPI And CGI Restrictions to open the feature shown in Figure 6.3. To add a new ISAPI extension, click Add in the Actions pane and then enter the exact path of the ISAPI extension. To allow the ISAPI extension to execute, check the Allow Extension Path To Execute check box.
In addition to using IIS Manager, edit the system.webServer/security/isapiCgi-Restriction configuration section directly by using the Appcmd command line tool or with another configuration API.

### 6.1.4 Enable only the required Common Gateway Interface (CGI) applications.

IIS 7.0 continues to support CGI programs as one of the ways to extend the functionality of the Web server. By default, IIS 7.0 does not provide any CGI programs, so they should be used only if the application uses third-party CGI programs.

**Note:**

a) To enable CGI programs to work on IIS 7.0, the CGI role service must be installed. This role service installs the CgiModule module, which provides support for launching CGI programs. If this module is removed, CGI programs will not be usable. This role service is not enabled by default.

b) To allow CGI programs server administrator privileges are needed.

Following steps should be taken to properly configure CGI programs:

1. **If Web server uses CGI programs, install the CGI role service.** Without this role service, the CGI programs will not be created, and requests to resources mapped to CGI programs will return errors.
2. If Web server does not use CGI programs, do not install the CGI role service. This eliminates the possibility of unwanted CGI programs being configured on server.

3. Configure only the allowed CGI programs. Each CGI program must be allowed to execute on the server before it can be used. Use IIS Manager to configure all CGI programs that are allowed to execute on the server. Exercise extreme caution when allowing third-party CGI programs and be sure their source is trusted. Installing untrusted or buggy CGI programs can compromise the security of the Web server or negatively affect its reliability.

4. Configure the desired handler mappings. To use CGI programs, handler mappings are required to be created that map allowed CGI programs to specific content types in the application.

Similar to ISAPI extensions, administrator must explicitly allow any CGI program that has to execute on server. When a specific CGI program path is allowed, this CGI program can now be launched by any application on the server that configures a handler mapping to this CGI program. To be allowed, each allowed CGI program entry must specify the full path and arguments exactly the same way they are specified in each handler mapping. CGI programs are allowed in the ISAPI and CGI Restrictions feature, similar to the process described earlier in the section titled “Enabling Only the Required ISAPI Extensions”.

6.1.5 Enable only the required FastCGI applications.

IIS 7.0 supports hosting FastCGI programs by using the FastCGI feature, which provides a more reliable way to host many application frameworks than CGI does. By default, IIS 7.0 does not provide any FastCGI programs, so they should be used only if application uses third-party FastCGI programs.

Note:
   a) To enable FastCGI programs to work on IIS 7.0, the CGI role service must be installed. This role service installs the FastCgiModule module, which provides support for launching FastCGI programs. If this module is removed, FastCGI programs will not be usable. This role service is not enabled by default.
   b) To allow CGI programs server administrator privileges are needed.

Following steps should be taken to properly configure FastCGI programs:

1. If Web server uses FastCGI programs, install the CGI role service. Without this role service, the FastCGI programs will not be usable, and requests to resources mapped to FastCGI programs will return errors.

2. If Web server does not use FastCGI programs, do not install the CGI role service. This eliminates the possibility of unwanted FastCGI programs being configured on server.

3. Configure the allowed FastCGI programs. Each FastCGI program must be allowed to execute on the server before it can be used. Though there is no IIS Manager support for configuring FastCGI programs that are allowed to execute on the server, administrator can do this by editing the system.webServer/fastCgi configuration section. Exercise extreme caution when allowing third-party FastCGI programs and be sure their source is trusted. Installing untrusted or buggy FastCGI programs can compromise the security of the Web server or negatively affect its reliability.

4. Configure the desired handler mappings. To use FastCGI programs, handler mappings are required to be created that map allowed FastCGI programs to specific content types in application.

Unlike ISAPI extensions and CGI programs, FastCGI programs are not allowed through the ISAPI and CGI Restriction feature. Instead an entry for each allowed FastCGI program is required to be created in the system.webServer/fastCgi configuration section.
7. Securing the Web Application

7.1 Reduce the attack surface area of the application

Installing only the required Web server features and locking down the enabled ISAPI extensions, ISAPI filters, and CGI and FastCGI programs is a way to reduce the surface area of the Web server as a whole. The process can be taken a step further by reducing the set of functionality available at the application level, by limiting the modules enabled in each application, and by constraining the set of resources that the application is configured to serve.

The process for reducing the attack surface area of the application includes the following steps:

- Enable only the required modules.
- Configure the minimal set of application handler mappings.
- Set Web site permissions.
- Configure a minimal set of MIME types.

7.1.1 Enable only the required modules

When the Web server is configured to run a single dedicated application, install only the modules necessary to host this application. However, if the Web server hosts multiple applications, install a superset of all IIS features and third-party modules that each application requires. In this case, the attack surface area of each application can further be reduced by disabling at the application level modules that the application does not need.

To do this, configure the set of enabled IIS modules (managed or native) for each application. It can be done by using IIS Manager: select the application node in the tree view, double-click Modules, and remove any modules that are not needed in the application.

If operating a Web server on which third parties are able to publish application content, be aware that they can by default (since it is not locked by default) enable new managed modules or disable any installed module at the server level that are included with their application to process requests to their application. This action can be locked either by locking the system.webServer/modules configuration section or using fine-grain configuration locking to lock specific modules against being removed.

Note:

a) Exercise caution when removing modules because removing security-sensitive modules that perform tasks—for example, those that perform authorization—can result in weakening application security. [For more details see Appendix B for “Function of Built-In Modules with Security Implications”]

b) By default, it is not possible to add new native modules or remove native modules associated with IIS features at the application level because IIS setup locks them at the server level.

7.1.2 Configure the minimal set of application handler mappings

Handler mappings directly determine what resource types the Web server is configured to serve. They do this by mapping extensions or URL paths to modules or ASP.NET handlers that provide processing for the corresponding resource type. Similar to modules, handler mappings are typically installed at the server level when IIS features or
third-party application frameworks are installed. This is done to enable all applications on the server to serve the associated content.

If the application does not serve specific content types or does not use specific application framework technologies installed on the Web server, remove the associated handler mapping entries in the `system.webServer/handlers` configuration section at the application level to prevent the Web server from attempting to use them to satisfy requests to application. This reduces the risk of unintended script functionality executing in the application, or an application framework specific vulnerability being exploited. Note that the latter may occur even if the application does not contain any resources or scripts for a particular application framework, if the application framework contains a vulnerability that manifests before it attempts to locate the requested script.

Use the following techniques to configure the minimal set of handler mappings for the application:

- **Review the handler mappings to understand what resource types can be processed in the application.** The Web server will attempt to satisfy each incoming request with the first handler mapping that matches the URL path and verb of the incoming request. Typically, the `StaticFileModule` will process all requests that have not matched other handler mappings, thus serving the requested resource as a static file if its extension is listed in the application’s MIME type configuration.

- **Remove any unused handler mappings in the application.** If possible, remove all handler mappings by clearing the `system.webServer/handlers` configuration section and re-adding only the handler mappings that the application uses.

- **Be aware of preconditions.** Because preconditions can be set on handler mappings to disable the use of these mappings in some application pools, some handler mappings may be ignored, resulting in the request being served using another matching handler mapping. To avoid security problems, do not create multiple handler mappings that rely on order to match similar requests.

- **Add applicable restrictions to handler mappings.** When adding new handler mappings, make use of the resource type restrictions to restrict the handler mappings only to requests that map to existing physical files or directories in the application. This can help stop malicious requests to resources that do not exist in the application. Additionally, make use of the access restrictions.

If operating a Web server on which third parties can publish application content, be aware that they can modify the handler mappings in any way to control how requests to their applications are processed. They can add new handler mappings to any enabled module, remove any of the existing handler mappings, or map requests to other handlers. This action can be locked by locking the `system.webServer/handlers` configuration section.

**Note:** Handler mappings that map requests to ISAPI extensions (IsapiModule), CGI programs (CgiModule), and FastCGI programs (FastCgiModule) are further limited by the ISAPI and CGI Restrictions and FastCGI program configuration at the Web server level, which can be set only by the administrator.

### 7.1.3 Set Web site permissions

Web site permissions are an additional restriction that can be placed on a Web site, application, or URL in configuration to control what requests IIS is allowed to serve. These permissions are implemented at two levels.

1. Each handler mapping specifies the required permission level by using the `requireAccess` attribute. If the request that matches this handler mapping is made and the required permission is not granted for the requested URL, IIS will reject the request with a 403.X response status code.

2. Some IIS components have hard-coded requirements for certain permissions, and they will reject the request if they are processing a request to a URL that does not have this permission.
Note: Web site permissions control what functionality is enabled to be used at a particular URL. They do not consider the identity of the requesting user and therefore cannot be used to replace IIS authorization schemes when implementing access control.

Table 7.1 indicates the permission types that can be granted for a particular URL.

<table>
<thead>
<tr>
<th>Permission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>No permissions are granted.</td>
</tr>
<tr>
<td>Read</td>
<td>Read access to files and directories is enabled. In particular, this enables the following: static file handler serving static files, directory listings, and default documents.</td>
</tr>
<tr>
<td>Script</td>
<td>Script processing is enabled. In particular, this enables the following: ISAPI extensions, CGI programs, FastCGI programs, and ASP.NET handlers. ISAPI extensions and CGI programs must specify a fixed script processor.</td>
</tr>
<tr>
<td>Execute</td>
<td>Running executables is enabled. In particular, this enables the following: ISAPI extensions and CGI programs with no script processor set (that execute the file provided in the request path). If granted, this permission will by default lead to IIS trying to execute EXE files as CGI applications and load DLL files as ISAPI extensions instead of downloading them.</td>
</tr>
<tr>
<td>Source</td>
<td>In previous versions of IIS, this permission enables WebDav requests to access the source of script files. No special handling of this permission is present in IIS 7.0.</td>
</tr>
<tr>
<td>Write</td>
<td>In previous versions of IIS, this permission enables WebDav requests to write files. No special handling of this permission is present in IIS 7.0.</td>
</tr>
<tr>
<td>NoRemoteRead</td>
<td>Prevents remote requests from using the Read permission.</td>
</tr>
<tr>
<td>NoRemoteScript</td>
<td>Prevents remote requests from using the Script permission.</td>
</tr>
<tr>
<td>NoRemoteExecute</td>
<td>Prevents remote requests from using the Execute permission.</td>
</tr>
<tr>
<td>NoRemoteWrite</td>
<td>Prevents remote requests from using the Write permission.</td>
</tr>
</tbody>
</table>

Table 7.1: Permission Types Granted for URLs

In IIS Manager, set the permission granted for a particular Web site, application, or URL by selecting the appropriate node in the tree view and then clicking Handler Mappings. Then, set the Read, Script, and Execute permissions by clicking Edit Permissions in the Actions pane. Doing this also automatically shows the handlers that require a permission that is not granted as disabled, to let know that requests to these handlers will be rejected.

Additionally, set the permissions directly in configuration by editing the accessPolicy attribute of the system.web/handlers configuration section or by using Appcmd or other configuration APIs to do it. For example, to grant only the Read permission to the /files subfolder of the Default Web Site, use the following Appcmd syntax.

```bash
%systemroot%\system32\inetsrv\Appcmd set config "Default Web Site/files" /section:handlers /accessPolicy:Read
```

Note: When specifying a configuration path to apply configuration to a specific Web site or URL, it may give an error indicating that the configuration is locked. This is because most security configuration sections, including all authentication sections, are locked at the Web server level to prevent delegated configuration. These sections can be unlocked to allow delegated configuration, or persist the configuration to applicationHost.config by using the /commit:apphost parameter with Appcmd commands.
Use the following guidelines when setting Web site permissions:

- **Remove unnecessary permissions for URLs that do not require them.** By default, Read and Script permissions are granted. For URLs that do not require the ability to execute dynamic application technologies, remove the Script permission. Do not grant additional permissions such as Execute unless necessary.

- **Applications can configure handler mappings that do not require permissions.** By default, applications can change existing handler mappings or create new handler mappings that do not require permissions. Because of this, do not rely on Web site permissions for controlling which handler mappings can or cannot be created by applications that use delegated configuration. The permissions are only guaranteed for built-in IIS features including the static file handler, IsapiModule, CgiModule, and ASP.NET handlers, which hardcode the permission requirements. In other cases, the permissions are guaranteed only if the `system.webServer/handlers` configuration section is locked and prevents changes to the handler mappings set by the Web server administrator.

Note: Unlike in IIS 6.0, wildcard handler mappings no longer ignore Web site permissions. In IIS 7.0, they require the same level of permissions as they would when mapped with nonwildcard handler mappings. Because of this, configurations in which a wildcard-mapped ISAPI extension is used for URLs that do not allow the Script permission will now be broken and require the Script permission to be granted.

### 7.1.4 Configure a minimal set of MIME types

In IIS 7.0, the MIME types configured by default have been upgraded to contain many of the new common file extensions. By default, to serve the corresponding physical file to the client, IIS handler mappings are preconfigured to direct all requests not mapped to other modules to the `StaticFileModule` (if the file does not exist, a 404 error response code is returned).

For security reasons, the `StaticFileModule` will serve only files whose extensions are listed in the MIME type configuration. This behavior is extremely important, because otherwise applications that contain scripts and other content that is processed by application framework technologies may end up serving these resources directly if the appropriate application framework handler mappings are not installed or become removed. In this situation, the MIME type configuration protects these resources from being served to the client and results in a 404.3 error returned to the client.

The default list of MIME types in IIS 7.0 should be safe for most applications. It can further be configured with the MIME types at the server level—or for a Web site, application, or URL—to mandate which file extensions are servable by the `StaticFileModule`. By reducing this list to only the extensions of the files known to be safe to serve, accidentally serving files that are part of an application and are not meant to be downloaded can be avoided.

Note: MIME type configuration prevents only unlisted files from being downloaded directly through the `StaticFileModule`. It does not protect the resources from being accessed through the application, nor does it protect them from being downloaded if they are mapped to custom handlers. To protect application resources that are not meant to be accessed, forbid their extensions or use Request Filtering to place the content in a directory that is configured as a hidden segment.
Use the following guidelines to securely configure the MIME types list:

- **Do not add file extensions to the MIME type’s configuration that are not meant to be downloaded directly.** This refers to any of the file types that are used by the application, such as ASP, ASPX, or MDB.

- **Configure a minimal set of MIME types for each application.** If possible, configure the MIME types for each application to contain only the minimal set of extensions. This can help prevent accidental serving of new files when they are added to the application. For example, if an application uses XML files to store internal data, make sure that the application does not include xml in its MIME type configuration even though the .xml extension is listed there by default when IIS is installed.

### 7.2 Configuring Applications for Least Privilege

The next most effective strategy to reduce the risk of a successful attack on Web server is to configure applications to run with the least privilege possible (Principle of "Least Privilege"). Doing this minimizes the amount of damage that results if an attacker successfully exploits any known or future vulnerability. Similar to reducing the surface area, this technique is not limited to blocking specific threats—it works well for any threat that may be present in application today or that may be found in the future.

The key to reducing the privilege of the application code in the IIS environment is to understand the identity under which the code executes, select the identity with the minimal number of privileges required, and limit the rights of the identity to access server resources. IIS 7.0 provides following techniques to apply principle of least privilege:

- Use a low privilege application pool identity
- Set NTFS file system (NTFS) permissions to grant minimal access
- Reduce trust of ASP.NET applications
- Isolate applications

#### 7.2.1 Use a low privilege application pool identity

The majority of code executed as part of a Web application is executed in the context of the IIS worker process and typically runs under the identity configured for the application pool. Therefore, using a least privilege application pool identity is the primary way to constrain the privileges and rights granted to the application code.

By default, IIS application pools are configured to run using the built-in Network Service account, which has limited rights on the Web server. When each IIS worker process is started, it also automatically receives membership in the IIS_IUSRS group. This group replaces the IIS_WPG group used in IIS 6.0 as the required group identifying all IIS worker processes on the server. IIS setup may still create the IIS_WPG group for backward-compatibility reasons, in which case IIS_IUSRS will be made a member of this group.

In addition, certain code in the application may execute with the identity of the authenticated user associated with each request. Table 7.2 summarizes the identities that may be used in application.
Identity Type | Used When… | Identities
--- | --- | ---
Application pool identity | - Accessing all files necessary for the execution of the IIS worker process  
- Accessing web.config files  
- Running FastCGI applications (by default)  
- Running ASP.NET applications (by default) | - Network Service by default; otherwise configured application pool identity  
- IIS_IUSRS group  
- Application Pool SID (IIS APPPOOL\<ApppoolName>)
Authenticated user | - Accessing static files  
- Running ISAPI extensions  
- Running CGI programs (by default)  
- Running FastCGI applications (if impersonation is enabled)  
- Running ASP.NET applications (if impersonation is enabled) | - IUSR by default when using anonymous authentication; otherwise configured anonymous user or application pool identity  
- Authenticated user if Windows token authentication methods are used
Virtual directory fixed credentials (when configured) | - Accessing all application content | - The configured virtual directory credentials

Table 7.2: Application Identities

Some important points to remember about authentication schemes are:

- When using authentication schemes that produce Windows tokens, such as Windows Authentication or Basic Authentication, be aware that when highly privileged users access the application, it will execute with higher privileges than intended. Therefore, it is recommended that do not allow users that have administrative privileges on the server to access the application.
- When using authentication schemes that produce Windows identities, applications may execute with the identity of the authenticated user.
- When using anonymous authentication, opt for configuring the anonymous user to be the application pool identity, to ensure that all code always executes under the application pool identity. This makes it significantly easier to manage the access rights of the worker process.

**Note:** To simplify access management, configure the anonymous authentication user to be the application pool identity.

When selecting the application pool identity for applications, use the following guidelines to maintain or improve the security of Web server:

- **Ensure that the application pool identity is not granted sensitive privileges or unnecessary rights to access resources.** Often, in the face of “access denied” errors, administrators tend to grant the application pool identity full or otherwise unnecessary access to resources. This increases the privilege of the worker process and increases the risk of a serious compromise if the code in the worker process is compromised. Grant the worker process the minimal access only needed for the application to work. If this minimal access involves privileges or rights typically associated with administrative users, then there is a need to re-evaluate application’s design.

- **Do not use highly privileged or administrative identities for IIS application pools.** Never use LocalSystem, Administrator, or any other highly privileged account as an application pool identity.

- **Consider using a lower privilege identity.** If application allows it, consider using a custom low privilege account for the IIS worker process. Unlike IIS 6.0, IIS 7.0 automatically injects the new IIS_IUSRS group into the worker process, eliminating the need to make the new identity a member of any group.
• **Separate code with different privilege requirements into different application pools.** If server has multiple applications that require different levels of privilege (for example, one requires the privilege to write to the Web application, and the other one doesn't), separate them into two different application pools.

• **When using anonymous authentication, configure the anonymous user to be the application pool identity.** This significantly simplifies configuring access rights for application by making the application code always execute with the application pool identity.

• **Grant minimal access.** When granting access to the application pool identity, grant the minimal access necessary. Use this in conjunction with separating applications into different application pools to maintain least privilege for applications. To grant access to a resource for all IIS application pools, grant it to the IIS_IUSRS group. To grant access to a resource for a specific application pool, use the unique application pool identity. Alternatively, use the automatic Application Pool SID that is named IIS APPPOOL\<ApppoolName> (the latter does not work for UNC content, only local content). Do not grant access rights to Network Service because it grants access to all services running on the server under the Network Service identity.

### 7.2.2 Set NTFS permissions to grant minimal access

By default, all files required for IIS worker processes to function grant access to the IIS_IUSRS group, which ensures that IIS worker processes can function regardless of the selected application pool identity. However, it is up to administrator to grant access to the application content so that the Web server and the application can successfully access its resources. Additionally, to grant access to the additional resources the IIS worker process uses, such as ISAPI extensions, CGI programs, or custom directories configured for logging or failed request tracing.

Table 7.3 indicates the level of access the Web server typically requires for different kinds of resources.

<table>
<thead>
<tr>
<th>Resource Type</th>
<th>Identity</th>
<th>Required Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content (virtual directory physical path and below)</td>
<td>• Fixed credentials set on the virtual directory (if set) OR</td>
<td>• Read</td>
</tr>
<tr>
<td></td>
<td>• Authenticated users</td>
<td>• Write, if application requires being able to write files in the virtual directory (granting Write is not recommended)</td>
</tr>
<tr>
<td></td>
<td>• IIS worker process identity (application pool identity)</td>
<td></td>
</tr>
<tr>
<td>Additional resources IIS features use: CGI programs, ISAPI extensions, native module dynamic-link libraries (DLLs), compression directory, failed request tracing directory, logging directory, and more</td>
<td>• IIS worker process identity (application pool identity)</td>
<td>• Read</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Execute, for CGI programs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Write, for compression or logging directories, or whenever the Web server needs to write data</td>
</tr>
</tbody>
</table>

Table 7.3: Access Levels for Web Server Resources
When granting access to content directories, use one of the following techniques:

- **Grant access to IIS_IUSRS.** This enables all IIS worker processes to access the content when using the application pool identity, or when using anonymous authentication. However, this does not enable to isolate multiple application pools. If using a Windows-based authentication scheme, grant access to all of the authenticated users also that use the application.

- **Grant access to the identity of the application’s application pool.** This will enable only the IIS worker processes running in the application pool with the configured identity to access the content. If using anonymous authentication, additionally set the anonymous user to be the application pool identity. If using a Windows-based authentication scheme, grant access to all of the authenticated users also that use the application. This approach is the basis for application pool isolation.

- **Configure fixed credentials for the application’s virtual directory and grant access to these credentials.** This will prompt the IIS worker process to access the content by using the fixed credentials, regardless of the authenticated user identity. This option is often used when granting access to remote UNC shares to avoid the difficulties of ensuring that authenticated user identities can be delegated to access the remote network share. It can also be an efficient way to manage access to the content for a single identity regardless of the authenticated user (which can be set to the application pool identity when using custom application pool identities). Finally, it can be used to control access to the application when hosting multiple applications inside the same application pool.

Note: If using IIS Manager to administer the application remotely, grant Read access to the NT Service\WMSvc account.

If using an authentication scheme (other than anonymous authentication) that produces Windows identities for authentication users, such as most of the IIS authentication schemes, make sure that all authenticated users that require the use of the application have access to its content. This is because the Web server will use the authenticated user identity to access application content. Also, many application frameworks will by default impersonate the authenticated user when executing application code and accessing application resources.

If there is need to allow multiple Windows users to use the application, add all of these users to a specific group and grant this group access to the application content. Alternatively, when using the fixed credentials model, do not grant access to the authenticated users. Instead, IIS and application code will always impersonate the fixed virtual directory credentials.

If the content is on a UNC share, administrator will likely need to use the fixed credentials model because most IIS authentication schemes do not produce Windows tokens that can be used for remote network shares (with the exception of basic authentication and IIS client certificate mapping authentication). Alternatively, administrator can configure the Web server for Constrained Delegation and Protocol Transition to allow the authenticated user tokens to be used against the remote share. However, using the fixed credentials for virtual directories on UNC shares is significantly easier to configure, so it is recommended over setting up delegation.

Note: IIS 7.0 requires the IIS worker process identity to have access to the content directories before they can read the web.config configuration files. This happens before IIS determines the authenticated user.
7.2.3 Reduce trust of ASP.NET applications

In addition to constraining the execution rights of the application by using a low privilege application pool identity, administrator can further sandbox the .NET parts of the application by using the ASP.NET trust levels. The ASP.NET parts of the application include the ASP.NET applications themselves, as well as managed modules that provide services for any application in ASP.NET Integrated mode applications.

ASP.NET trust levels use the Code Access Security (CAS) infrastructure in the .NET Framework to limit the execution of the application code, by defining a set of code permissions that control what application code can and cannot do. By default, ASP.NET applications and managed modules execute using the Full trust level, which does not limit their execution. In this trust level, the application can perform any action that is allowed by the Windows privileges and resource access rights.

Administrator can reduce the trust level of ASP.NET applications to limit their execution further. This can be an effective way to achieve lower privilege for application. By default, options available are described in Table 7.4, which are defined by the ASP.NET trust policy files.

<table>
<thead>
<tr>
<th>.NET Trust Level</th>
<th>Execution Limits</th>
<th>Rights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full (internal)</td>
<td>None</td>
<td>All</td>
</tr>
<tr>
<td>High (web_hightrust.config)</td>
<td>None/.NET</td>
<td>Can do anything except call native code</td>
</tr>
<tr>
<td>Medium (web_mediumtrust.config)</td>
<td>Application is trusted within its own scope, but should not be able to affect other applications or the rest of the machine</td>
<td>• Access files in the application root</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Connect to SQL and OLEDB databases</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Connect to Web services on the local machine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Manipulate threads and execution for its own requests</td>
</tr>
<tr>
<td>Low (web_lowtrust.config)</td>
<td>Application is not highly trusted; meant for applications that can use built-in ASP.NET features but do not run custom code</td>
<td>Only read access within application root</td>
</tr>
<tr>
<td>Minimal (web_minimaltrust.config)</td>
<td>Application is untrusted; ability to use built-in ASP.NET features is extremely restricted</td>
<td>Minimal permissions for executing code</td>
</tr>
</tbody>
</table>

Table 7.4: Default Trust Level Options

It is recommended that run ASP.NET applications by using the Medium trust level. In this trust level, the application is not able to access resources outside of itself and cannot perform operations that can compromise the security of the Web server overall. However, administrator should test the application to make sure that it does not experience any security exceptions due to the lack of required permissions. And also make sure that using the reduced trust level does not negatively impact the application’s performance.

*Note: The Medium trust level is the recommended trust level to constrain the execution of ASP.NET applications and managed modules, and to host multiple applications on a shared Web server.*

Use IIS Manager to configure the trust level used for ASP.NET applications and managed modules by double-clicking .NET Trust Levels. Administrator can do this for the entire Web server—or for a specific application—by selecting that application node prior to using the .NET Trust Levels feature. OR alternatively set the trust level directly by changing the level attribute in the `system.web/trust` configuration section.
Note: The system.web/trust configuration section is not locked by default, which means that any application can configure its own trust level. It can be locked through the configuration section at the server level.

7.2.4 Isolate applications

Application pools provide a great way to isolate multiple applications running on the same machine, both in terms of availability and security. This provides the following benefits:

- Failures, instability, and performance degradation experienced in one application do not affect the applications in a different application pool.
- Applications running in a different application pool can restrict access to their resources to that application pool only, preventing other applications from being able to access their resources.

The recommended way to configure applications for isolation is to place each application into a separate application pool. IIS 7.0 makes it easy to isolate applications by automatically injecting a unique application pool security identifier, called the application pool SID, into the IIS worker process of each application pool. Each application pool SID is generated based on the application pool name and has the name \texttt{IIS APPPOOL\textbackslash\{Apppool-Name\}}. The application pool SID makes it possible to quickly isolate applications by placing NTFS permissions on their content to grant access only to the application pool SID of the application’s application pool.

Note: Isolate applications by setting permissions on their content to allow only the Application Pool SID of the corresponding application pool.

Take the following steps to make Application Pool SID–based isolation effective:

1. Configure anonymous authentication to use the application pool identity.
2. Grant access to the application content for the \texttt{IIS APPPOOL\textbackslash\{AppName\}} SID.
3. Do not grant access to the application content to IIS_IUSRS, IIS_WPG or any other application pool identity that may be used by another application pool.
4. Configure separate locations for all temporary and utility directories that IIS and the application use for each application or application pool, and set permissions on them to allow access only for the \texttt{IIS APPPOOL\textbackslash\{ApppoolName\}} SID.

Table 7.5 shows some of the common directories that IIS and ASP.NET applications use. The directories must be configured for isolation for each Web site or application and receive the appropriate permissions to enable access only by the associated application pool.
<table>
<thead>
<tr>
<th>Directory</th>
<th>Virtual directory physical path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows TEMP directory (%TEMP% or %TMP%): used by Windows components</td>
<td>Set the <code>loadUserProfile</code> attribute to true in the <code>processModel</code> element of each application pool. This causes the TEMP directory to point to <code>%SystemDrive%\Users\%UserName%\AppData\Local\Temp</code>.</td>
</tr>
<tr>
<td>Web site log file directory</td>
<td><code>directory</code> attribute of the <code>logFile</code> element for each site. The default is <code>%SystemDrive%\Inetpub\Logs\LogFiles</code>.</td>
</tr>
<tr>
<td>Web site Failed Requests Logs directory</td>
<td><code>directory</code> attribute of the <code>traceFailedRequestsLogging</code> element of each site. The default is <code>%SystemDrive%\Inetpub\Logs\FailedReqLogFiles</code>.</td>
</tr>
<tr>
<td>IIS Static Compression directory</td>
<td>Isolated automatically by creating a subdirectory for each application pool and applying ACLs to each directory for the Application Pool SID.</td>
</tr>
<tr>
<td>ASP.NET Temp directory: used by ASP.NET compilation system</td>
<td><code>tempDirectory</code> attribute in <code>system.web/compilation</code> section for each application. The default is <code>%SystemRoot%\Microsoft.NET\Framework\&lt;version&gt;\Temporary ASP.NET Files</code>.</td>
</tr>
<tr>
<td>ASP Template Disk Cache directory</td>
<td>Isolated automatically by applying ACLs to each file for the Application Pool SID.</td>
</tr>
</tbody>
</table>

Table 7.5: Common Directories Used by IIS and ASP.NET Applications

Note: The Application Pool SIDs can be used only for isolating local content. If using content located on a UNC share, administrator needs to either use a custom application pool identity or configure fixed credentials for each virtual directory. Then he should use that identity to grant access to the network share.

IIS 7.0 provides automatic isolation of the server-level configuration by using configuration isolation. No action is necessary to enable this, because it is done by default. The server-level configuration in applicationHost.config is isolated automatically using configuration isolation.

However, .NET Framework configuration in the machine.config and root web.config files—as well as the configuration in the distributed web.config files that are part of the Web site’s directory structure—are not isolated. To properly isolate the distributed web.config files, set the appropriate permissions on the content directories.

### 7.3 Implementing Access Control

Web applications require the ability to restrict access to their content, to protect sensitive resources, or to authorize access to resources to specific users. IIS 7.0 provides an extensive set of features that can be used to control the access to application content. These features are logically divided into two categories, based on the role they play in the process of determining access to the request resource:

- **Authentication**: Authentication features serve to determine the identity of the client making the request, which can be used in determining whether this client should be granted access.
- **Authorization**: Authorization features use the authenticated identity on the request or other applicable information to determine whether or not the client should be granted access to the requested resource. Authorization features typically depend on the presence of authentication features to determine the
authenticated identity. However, some authorization features determine access based on other aspects of the request or the resource being requested, such as Request Filtering.

IIS 7.0 supports most of the authentication and authorization features available in IIS 6.0, and it introduces several additional features. These features (role services) are listed here in the order in which they apply during the processing of the request:

1. **IP and Domain Restrictions**: Used to restrict access to requests clients make from specific IP address ranges or domain names. The default install does not use this feature.

2. **Request Filtering**: Similar to UrlScan in previous versions of IIS, request filtering is used to restrict access to requests that meet established limits and do not contain known malicious patterns. In addition, Request Filtering is used to restrict access to known application content that is not meant to be served to remote clients. Request filtering is part of the default IIS 7.0 install and is configured to filter requests by default.

3. **Authentication features**: IIS 7.0 offers multiple authentication features that can be used to determine the identity of the client making the request. These include Basic Authentication, Digest Authentication, Windows Authentication (NTLM and Kerberos), and many others. The Anonymous Authentication feature is part of the default IIS install and is enabled by default.

4. **Authorization features**: IIS 7.0 provides a new URL Authorization feature that can be used to create declarative access control rules in configuration to grant access to specific users or roles. In addition, it continues to support NTFS ACL-based authorization for authentication schemes that yield Windows user identities. IIS uses NTFS ACL-based authorization by default.

Note: In IIS 7.0, all of these role services are available as Web server modules that can be individually installed/uninstalled and optionally disabled/enabled for each application. Be careful when removing authentication, authorization, and other access control modules, because it may unintentionally open access to unauthorized users or make the application vulnerable to malicious requests.

Administrators should leverage access control features to configure the right authentication and authorization features for the application. And to ensure that only users with the right to access those resources can access them.

In addition, they should take advantage of Request Filtering to limit usage of the application as much as is possible, by creating restrictions on content types, URLs, and other request parameters. Doing so enables them to preemptively protect the application from unexpected usage and unknown exploits in the future.

### 7.3.1 IP and Domain Restrictions

The IP and Domain Restrictions role service enables to restrict access to the application to clients making requests from a specific IP address range or to clients associated with a specific domain name.

Note: The IP and Domain Restrictions role service is not part of the default IIS installation. It can be installed manually from the IIS \ Security feature category in Windows Setup on Windows Vista, or from the Security category of the Web Server (IIS) role in Server Manager on Windows Server 2008.

Use this feature to allow or deny access to a specific range of IP addresses, or to a specific domain name. The IP and Domain Restrictions role service will attempt to match the source IP address of each incoming request to the configured rules, in the order in which rules are specified in configuration. If a matching rule is found, and the rule is denied access to the request, the request will be rejected with a 403.6 HTTP error code. If the rule allows access, the request will continue processing (all additional rules will be ignored).
Any number of allow or deny rules can be specified and indicate whether access should be granted or denied if no rules match. The common strategies for using IP Address and Domain Restrictions rules include:

- Denying access by default and creating an Allow rule to grant access only to a specific IPv4 address range, such as the local subnet. It can be done to grant access only to clients on the local network or to a specific remote IP address.
- Allowing access by default and creating a Deny rule to deny access to a specific IP address or IPv4 address range. It can be done to deny access to a specific known malicious IP address.

Note: Allowing access by default and denying access for specific IP address ranges is not a secure technique, because attackers can and often will use different IP addresses to access the application. Also, clients that use IPv6 addresses will not match a Deny rule that uses an IPv4 address range.

For restricting access to the application to clients on the local network, implement an additional defense measure by specifying that the site binding should listen only on the IP address associated with the private network. For servers that have both private and public IP addresses, this can restrict requests for specified site to the private network only. It can be used in conjunction with IP and Domain Restrictions where possible.

It is advisable that administrators should not use a domain name instead of specifying an IP address. This is because domain name-based restrictions require a reverse Domain Name System (DNS) lookup of the client IP address for each request, which can have a significant negative performance impact on server. By default, the feature does not enable the use of domain name–based rules.

Take following steps to configure the IP and Domain Restrictions rules:

- Use IIS Manager to configure the rules by selecting the Server node, a Site node, or any node under the site in the tree view. Then double-click the IP Address And Domain Restrictions feature, which is shown in Figure 7.1.
- Use the Add Allow Entry or the Add Deny Entry commands in the Actions pane to create allow or deny rules.
- Use the Edit Feature Settings command to configure the default access (allow or deny) and whether or not domain name–based rules are allowed.
Figure 7.1: Configuring IP and Domain Restrictions using IIS Manager.

Note:

a) Although the IP and Domain Restrictions feature enables use of IPv6 addresses, but addresses cannot be configured to use IPv6 rules using IIS Manager. Also, requests that are made over IPv6 connections do not match rules using IPv4 addresses and vice versa.

b) By default, the ‘ipSecurity’ configuration section is locked at the server level. It can be unlocked by using the ‘Appcmd Unlock Config’ command to specify IP and Domain Restriction rules in web.config files at the site, application, or URL level.

The IP and Domain Restrictions can also be configured by using Appcmd or configuration APIs to edit the system.webServer/security/ipSecurity configuration section.

7.3.2 Request Filtering

The Request Filtering feature is an improved version of the ‘UrlScan’ tool available for previous versions of IIS. The Request Filtering feature enforces limitations on the format of the request URL and its contents to protect the application from possible exploits that may arise from exceeding these limits. With IIS 6.0 and previous versions of IIS, these limits have thwarted the majority of known Web application exploits, such as application-specific buffer overruns resulting from long malicious URLs and query strings. Though the Web server itself, starting with IIS 6.0,
has been engineered to not be vulnerable to these attacks, these limits remain valuable in preventing both known and unknown future exploits for the Web server and applications running on it.

In addition to enforcing request limits, the Request Filtering feature also serves to deny access to certain application resources that are not meant to be served to Web clients, but are located in the servable namespace for an application. These files include the web.config configuration files for IIS and ASP.NET, as well as contents of the /BIN and /App_Code directories for ASP.NET applications.

Request Filtering can be used to do the following:

- **Set request limits**: Configure limits on the length and encoding of the URL, the length of the query string, the length of the request POST entity body, allowed request verbs, and maximum lengths of individual request headers.
- **Configure allowed extensions**: Configure which file extensions are allowed or rejected, regardless of the selected handler mapping.
- **Configure hidden URL segments**: Configure which URL segments are not served, to hide parts of URL hierarchy.
- **Configure denied URL sequences**: Configure which URL patterns are not allowed, possibly to prevent known exploits that use specific URL patterns.

IIS 7.0 depends on request filtering by default to reject requests that may contain malicious payloads and to protect certain content from being served by the Web server. It can be used to further restrict the input to the application, or to protect additional URLs, directories, and files or file extensions from being served by the Web server.

*Note: Request filtering is a critical security component and should not be removed unless it is absolutely clear that it is not needed. Always prefer to relax request filtering limits by setting the configuration rather than uninstalling or removing the Request Filtering module.*

The request filtering configuration does not have an associated IIS Manager page, so these settings cannot be set through IIS Manager. To set them, configure through the `system.webServer/security/requestFiltering` configuration section directly at the command line by using Appcmd.exe or other configuration APIs. The `requestFiltering` section is unlocked by default, thus set request filtering configuration in web.config files at the site, application, or URL level.

### 7.3.2.1 Setting Request Limits

Use Request Filtering to configure even tighter request limits if allowable in application’s usage to further reduce attackers’ ability to exploit the application with malicious input. At other times, relax the default request limits to allow the application to function correctly, for example, if application uses long query strings that may exceed the default limit of 2048 characters.

If any of the default limits interfere with application usage, modify request limits which are applied by request filtering. Use Appcmd to set request filtering limits as follows.

```
%systemroot%\system32\inetsrv\Appcmd set config [ConfigurationPath]
/section:system.webServer/security/requestFiltering
[/allowDoubleEscaping:bool] [/allowHighBitCharacters:bool]
[/requestLimits.maxAllowedContentLength:uint] [/requestLimits.maxUrl:uint]
[/requestLimits.maxQueryString:uint]
```

---

SECURING IIS 7.0 WEB SERVER  Page 49 of 128
This command uses the parameters in Table 7.6.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConfigurationPath</td>
<td>The configuration path at which to set this configuration.</td>
</tr>
<tr>
<td>allowDoubleEscaping</td>
<td>Whether or not request URLs that contain double-encoded characters are allowed. Attackers sometimes use double-encoded URLs to exploit canonicalization vulnerabilities in authorization code. After two normalization attempts, if the URL is not the same as it was after one, the request is rejected with the 404.11 response status code. The default is false.</td>
</tr>
<tr>
<td>allowHighBitCharacters</td>
<td>Whether or not non-ASCII characters are allowed in the URL. If a high bit character is encountered in the URL, the request is rejected with the 404.12 response status code. The default is true.</td>
</tr>
<tr>
<td>requestLimits.maxAllowedContentLength</td>
<td>The maximum length of the request entity body (in bytes). If this limit is exceeded, the request is rejected with the 404.13 response status code. The default is 30000000 bytes (approximately 28 megabytes).</td>
</tr>
<tr>
<td>requestLimits.maxUrl</td>
<td>The maximum length of the request URL’s absolute path (in characters). If this limit is exceeded, the request is rejected with the 404.14 response status code. The default is 4096 characters.</td>
</tr>
<tr>
<td>requestLimits.maxQueryString</td>
<td>The maximum length of the query string (in characters). If this limit is exceeded, the request is rejected with the 404.15 response status code. The default is 2048 characters.</td>
</tr>
</tbody>
</table>

Table 7.6: Parameters for Requesting Filtering Limits

Note: Administrator can also configure the request header length limits for each header by adding header limits in the ‘headerLimits’ collection in the ‘system.webServer/security/requestFiltering’ section. If the request exceeds a configured header limit, it will be rejected with a 404.10 response status code. Additionally, allowed verbs can be configured by adding those verbs to the ‘verbs’ collection. If the request specifies a verb that is not allowed, it is rejected with a 404.6 response status code.

7.3.2.2 Configuring Allowed Extensions

By default, the Web server will process only extensions for which a handler mapping exists. By default, StaticFileModule processes all unmapped extensions. StaticFileModule serves only extensions listed in the system.webServer/staticContent configuration section (known as Mimemaps in IIS 6.0 and previous versions of IIS). Therefore, the handler mapping configuration and the static content configuration serves as the two-level mechanism for controlling which extensions can be served for a particular URL.

However, as a defense in depth measure, use Request Filtering to deny requests to a particular extension, after making sure that it is not configured in the IIS handler mappings and the static content list. This makes sure that requests to this extension are rejected very early in the request processing pipeline, much before they otherwise would be rejected by the configuration mentioned earlier.

To add a prohibited or explicitly allowed extension by using Appcmd, use the following syntax.

```
%systemroot%\system32\inetsrv\Appcmd set config [ConfigurationPath] /section:system.webServer/security/requestFiltering /+fileExtensions.[fileExtension='string',allowed='bool']
```
The fileExtension string is in the format of .extension.

To delete a prohibited extension by using Appcmd, use the following syntax.

```
%systemroot%\system32\inetsrv\Appcmd set config [ConfigurationPath]
/section:system.webServer/security/requestFiltering
/-fileExtensions.[fileExtension='string']
```

These commands use the parameters in Table 7.7.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConfigurationPath</td>
<td>The configuration path at which to set this configuration.</td>
</tr>
<tr>
<td>fileExtension</td>
<td>The extension, in the format of .extension, that should be allowed or denied.</td>
</tr>
<tr>
<td>allowed</td>
<td>Whether or not the extension is allowed or denied.</td>
</tr>
</tbody>
</table>

Table 7.7: Parameters for Deleting Prohibited Extension

Administrator can set this configuration for a particular site, application, or URL by specifying the configuration path with commands shown earlier in this section.

Note: Alternatively, administrator can configure the ‘fileExtensions’ collection to deny all unlisted extensions, explicitly enabling the extensions that are allowed. This may be a more effective practice for reducing surface area than prohibiting specific extensions. However, it requires administrator to know and maintain the exhaustive list of all allowed extensions necessary for application. To do this, set the ‘allowUnlisted’ attribute on the ‘fileExtensions’ collection at the desired configuration path to ‘false’.

7.3.2.3 Configuring Hidden URL Segments

ASP.NET uses a technique to prohibit requests to the /BIN, /App_Code, and other special /App_xxx directories that contain ASP.NET application resources that the application is not to serve. Administrator can create their own special directories that contain nonservable content and protect them with URL segments. For example, to prevent content from being served from all directories named data, administrator can create a hidden segment named data.

To add a hidden URL segment by using Appcmd, use the following syntax.

```
%systemroot%\system32\inetsrv\Appcmd set config [ConfigurationPath]
/section:system.webServer/security/requestFiltering
/+hiddenSegments.[segment='string']
```

The segment string is the segment to protect.

To delete a hidden URL segment by using Appcmd, use the following syntax.

```
%systemroot%\system32\inetsrv\Appcmd set config [ConfigurationPath]
/section:system.webServer/security/requestFiltering
/-hiddenSegments.[segment='string']
```

Note: Unlike file extensions, there is no way to prohibit all segments other than the ones configured. Administrator can only deny specific segments by adding them using the preceding command.
Administrator can target configuration for a particular site, application, or URL by specifying the configuration path with the preceding commands.

### 7.3.2.4 Configuring Denied URL Sequences

In some cases, administrator may want to reject requests that contain specific sequences in the URL, whether or not they are a complete segment. If it is not possible to fix the application itself, this may be an effective way to protect an application from certain URL patterns that are known to cause issues.

To add a denied URL sequence by using Appcmd, use the following syntax.

```bash
%systemroot%\system32\inetsrv\Appcmd set config [ConfigurationPath]
/section:system.webServer/security/requestFiltering
/+denyURLSequences.[sequence='string']
```

The sequence string is the sequence to reject.

To delete a denied URL sequence by using Appcmd, use the following syntax.

```bash
%systemroot%\system32\inetsrv\Appcmd set config [ConfigurationPath]
/section:system.webServer/security/requestFiltering
/-denyURLSequences.[sequence='string']
```

Note: Unlike file extensions, there is no way to prohibit all URL sequences other than the ones configured. Administrator can only deny specific sequences by adding them using the preceding command.

Administrator can set this configuration for a particular site, application, or URL by specifying the configuration path with the preceding commands.

### 7.3.3 Authorization

Authorization refers to determining if the user identity determined during the authentication phase is allowed to access the requested resource. This is the second phase in the process of determining whether or not a client has the right to issue a particular request.

IIS 7.0 provides several authorization mechanisms that can be leveraged to control access to resources:

- **NTFS ACL-based authorization**: By default, IIS 7.0 verifies that the authenticated user identity has the right to access the physical file or folder corresponding to the requested URL. This check is performed only for requests that map to physical files or directories and use authentication methods that produce Windows tokens. This authorization mechanism has multiple usage limitations.

- **URL Authorization**: The new IIS 7.0 URL Authorization role service enables applications to create declarative configuration-based rules to determine which authenticated users and/or roles have the right to access the Web site or specific URLs therein.

- **ASP.NET URL Authorization**: ASP.NET applications that use this feature can configure it to control access to all Web site content when they run using ASP.NET integrated mode.

In addition, developers can provide custom authorization features by developing modules by using the IIS 7.0 native module APIs or the ASP.NET APIs for applications using ASP.NET Integrated mode. In fact, they can use
most existing ASP.NET authorization modules immediately in applications that are using ASP.NET Integrated mode. This makes it significantly easier to develop custom authorization features that implement business authorization rules and can leverage the powerful ASP.NET membership and role infrastructures.

Note: Exercise extreme caution when configuring or removing authorization modules. If an authorization module that is used to restrict access to the application is removed, parts of the application may become exposed to unauthorized users.

7.3.3.1 NTFS ACL-based Authorization

The IIS 7.0 server automatically performs NTFS ACL-based authorization. During this authorization, the Web server checks that the authenticated user identity has the rights to access the physical file or folder being requested.

NTFS ACL-based Authorization is part of the IIS Web server core and therefore is always installed when the Web server is installed. Though it cannot be uninstalled or disabled, still the application can be configured to use it as per requirements in the following list.

This authorization occurs automatically when all of the following conditions are met:

- The authenticated user identity must have a Windows token. If the request is authenticated using an authentication method that does not provide Windows tokens, for example Forms Authentication, this authorization is not performed.
- The selected handler mapping for the request specifies a resource type of File, Directory, or Either. Some mappings use the resource type of Unspecified to enable requests to virtual URLs that do not have a corresponding physical resource on disk. For these handler mappings, this authorization is not performed. **Note:** Most ASP.NET handler mappings are marked as Unspecified by default. However, ASP.NET includes additional functionality that ensures that if the URL maps to a physical file or folder, the access check is performed (with the exception of content located in a virtual directory that stores its files on a UNC path).
- The request URL maps to a physical file or folder that exists on disk. If the file or directory does not exist, IIS does not perform the check.
- The virtual directory corresponding to the request being made does not specify fixed access credentials. If the virtual directory specifies fixed credentials, they will be used to access all content for the virtual directory, and therefore IIS does not use the authenticated user to check access.

In addition, to successfully use NTFS ACL-based authorization, the following conditions must also be true:

- The physical resources have ACLs configured to properly grant or deny access to each authenticated user. This is typically done by placing all of the allowed users in a group, and granting this group access to the content.

- If the virtual directory corresponding to the request refers to a remote UNC share and does not specify fixed UNC credentials, the authenticated user identity must be able to delegate to the remote server. This requires basic authentication, or requires Constrained Delegation or Protocol Transition to be configured.

Because of the aforementioned limitations and the overhead of managing NTFS ACL permissions for multiple users, NTFS ACL-based authorization is not recommended as a generic mechanism for restricting access to IIS resources. Use it only if the application meets the preceding requirements and would like to use ACLs as an authorization mechanism (for example, if sharing static resources for users with domain or local machine accounts, and the resources already have the right permissions configured).

Because this authorization happens automatically for physical resources, configure all required resources to grant access to the authenticated identities that need to use application.
7.3.3.2 URL Authorization

URL Authorization is a new feature in IIS 7.0. It provides a way to configure declarative access control rules that grant or deny access to resources based on the authenticated user and its role membership. It is not part of the default IIS install. It can be installed manually from the Windows Features IIS feature category through Windows Features on Windows Vista or from the Security role service category of the Web Server (IIS) role in Server Manager on Windows Server 2008.

Unlike NTFS ACL-based authorization, URL Authorization has the following advantages:

- It is not tied to authentication schemes that produce Windows identities. It can be used with any authentication schemes, including Forms Authentication, which produces custom authenticated user identities.
- It enables rules to be configured for specific URLs, not underlying files or directories. Therefore, it is not tied to specific resource types and does not require files or directories to exist.
- It stores authorization rules in the configuration, instead of NTFS ACLs. These rules are easier to create and manage, and they can be specified in distributed web.config files that travel with the application when it is deployed or copied between servers.
- It integrates with the ASP.NET Membership and Roles services, which enables custom authentication and role management modules to provide the authenticated users and their roles. ASP.NET Forms Authentication can be used with Membership and Roles to quickly deploy a data-driven user and roles credential store for the application.

The ASP.NET URL Authorization feature inspired the new URL Authorization feature, which implements similar functionality. However, some key differences exist in how rules are configured and processed.

7.3.3.3 Using URL Authorization to Restrict Access

One or more URL Authorization rules need to be configured to use URL Authorization to restrict access to the application. These rules can be configured at the Web server level. Alternatively, it can be configured for a specific Web site, application, or URL. This makes it very easy to use URL Authorization to quickly restrict access to any part of the Web site to a specific set of users or roles.

These rules can be one of the following types:

- **Allow**: An allow rule grants access to the resource being requested and allows request processing to proceed.
- **Deny**: A deny rule denies access to the resource being requested, rejecting the request.

Both types of rules can specify a set of users, roles, and/or verbs that URL Authorization uses to match each rule to the request. As soon as a rule matches, the corresponding action (Allow or Deny) is taken. If the request is denied, URL Authorization will abort request processing with a 401 unauthorized response status code. If no rules matched, the request will also be denied.

Unlike ASP.NET URL Authorization, the deny rules are always processed before allow rules. This means that the relative order between deny and allow rules does not matter. In addition, the order between rules defined by parent configuration levels and the current configuration level does not matter, because all deny rules from all levels are always processed first, followed by all allow rules. Finally, the default behavior when no rules match is to deny the request.
By default, URL Authorization has a single rule configured at the Web server level that provides access to all users. Creating authorization rules by using the following techniques can restrict access to the Web site or a part of it:

- Remove the default allow rule for all users and create explicit allow rules only for users and roles that should have access to the current URL level. This way, by default, all requests will be denied unless the authenticated user or role matches the configured allow rule. This is the recommended practice, because it ensures that only the configured users and roles have access to the resource, and it denies access to everyone else.

- Create explicit deny rules for users and roles that should not have access to the current URL level. This may be appropriate to prevent access for only the specific users and roles. However, it is not generally a secure practice, because the set of users and roles is typically unbounded. The exception to this rule is the technique of creating a deny all anonymous users rule to restrict access only to authenticated users.

Note: When designing the access control rules for application, prefer to grant access to roles instead of specific users. This makes it easier to manage authorization rules as more users are added.

7.3.3.4 Creating URL Authorization Rules

Use IIS Manager to configure URL Authorization rules by selecting the desired node in the tree view and double-clicking Authorization Rules. In the resulting window shown in Figure 7.2, the list of rules currently in effect at the level selected can be seen, which will include both the rules configured at higher configuration levels and the rules configured at the current configuration level.
Figure 7.2: Configuring URL authorization rules.

Existing authorization rules can be removed (including parent authorization rules that are not locked) by selecting them in the list and clicking Remove in the Actions pane. Add allow or deny rule by clicking Add Allow Rule or Add Deny Rule in the Actions pane. Figure 7.3 shows an allow rule that can be used to allow access to all users or specific users or roles.
Alternatively, URL Authorization rules can also be edited by configuring them directly in the `system.webServer/security/authorization` configuration section, by using Appcmd from the command line, or by using configuration APIs. This configuration section is unlocked by default to facilitate storing URL authorization rules in application’s web.config files, which enables them to travel with the corresponding application content when deploying the application to another Web server.

Use the following syntax to add a URL authorization rule with Appcmd:

```
%systemroot%\system32\inetsrv\Appcmd.exe set config [ConfigurationPath]
/section:system.webServer/security/authorization
"/+[users='string',roles='string',verbs='string',accessType='enum']"
```

Use the following syntax to delete a URL authorization rule with Appcmd:

```
%systemroot%\system32\inetsrv\Appcmd.exe set config [ConfigurationPath]
/section:system.webServer/security/authorization
"-/users='string',roles='string',verbs='string']"
```
The parameters to these commands are shown in Table 7.8

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConfigurationPath</td>
<td>The configuration path at which this configuration is set.</td>
</tr>
<tr>
<td>users</td>
<td>The comma-separated list of user names that this rule allows or denies. Each user name is matched with the user name of the authenticated user set for the request, and the rule matches as soon as a single user matches (or a role matches). For Windows identities that represent domain accounts, use the domain qualified user name format of “domain\user” rather than the fully qualified domain name format of “<a href="mailto:user@domain.com">user@domain.com</a>”. Use “*” to refer to all users and “?” to refer to anonymous users. The default is “”.</td>
</tr>
<tr>
<td>Roles</td>
<td>The comma-separated list of roles that this rule allows or denies. Role membership for each role is tested for the authenticated user set for the request, and the rule matches as soon as a single role matches (or user matches). The default is “”.</td>
</tr>
<tr>
<td>Verbs</td>
<td>The comma-separated list of verbs (case-sensitive) that matches the request verb. If specified, one of the verbs must match the request verb for the rule to apply. The default is “”.</td>
</tr>
<tr>
<td>accessType</td>
<td>Whether the rule should allow or deny access. Accepted values are Allow and Deny. This parameter must be specified to add a rule.</td>
</tr>
</tbody>
</table>

Table 7.8: Parameters for Adding URL Authorization Rule

### 7.3.3.5 Using ASP.NET Roles with URL Authorization

In applications using the ASP.NET Integrated mode, it is possible to configure the ASP.NET Roles feature to provide application-specific roles for each authenticated user. IIS 7.0 URL Authorization rules can specify access rules that use roles provided by the .NET Roles feature or another ASP.NET module to implement application-specific authorization schemes.

When the .NET Roles feature is enabled, and a role provider is configured for application, begin configuring URL Authorization rules that rely on these roles in application. To make sure that the roles are available for requests to non-ASP.NET content types, be sure to remove the managedHandler precondition from the RoleManager module.

Roles can also be created directly using IIS Manager by selecting the application node in the tree view and then clicking .NET Roles. In the resulting page, manage existing roles, create new roles using the configured Role provider for the application, and associate application users with roles.

### 7.3.4 Authentication

Authentication is the process of determining the identity of the user making the request to the Web server. Authorization features can then use this identity to allow or reject the request to specific resources or parts of the application. In some cases, the Web server or the application can impersonate it to access resources. Finally, the application can use the identity to personalize the application experience for the requesting user.

IIS 7.0 includes the following authentication features:

---

6 Learn about setting up the ASP.NET Roles feature at [http://msdn2.microsoft.com/en-us/library/9ab2fxh0.aspx](http://msdn2.microsoft.com/en-us/library/9ab2fxh0.aspx)
• **Anonymous Authentication**: This authentication method provides a configured Windows identity for all anonymous users of the application without the need to provide any client credentials. It is used to allow anonymous (unauthenticated) access.

• **Basic Authentication**: This authentication method enables the client to provide the user name and password to the Web server in clear text. Basic Authentication is defined in RFC 2617, and virtually all browsers support it.

• **Digest Authentication**: This authentication method is a more secure version of Basic Authentication, and it enables the client to provide user credentials via a hash of the user name and password. Digest Authentication is defined in RFC 2617, and most browsers support it.

• **Windows Authentication**: This authentication method supports the NT LAN Manager (NTLM) or Kerberos Windows authentication protocols.

• **Client Certificate Mapping Authentication**: This authentication method enables client SSL certificates to be mapped to Windows accounts by using Active Directory (AD) directory services.

• **IIS Client Certificate Mapping Authentication**: This authentication method enables client SSL certificates to be mapped to Windows accounts via one-to-one or many-to-one mappings stored in IIS configuration.

• **UNC Authentication**: Though this is not a true authentication method in the sense that it does not help to establish the identity of the requesting client, IIS 7.0 uses UNC Authentication to establish an identity to access remote content located on a UNC share.

In addition, IIS 7.0 applications using ASP.NET Integrated mode use a unified authentication model between IIS and ASP.NET. This enables existing ASP.NET authentication modules or new managed authentication modules developed with ASP.NET APIs to be used for all content in the application. When ASP.NET is installed, the following authentication methods are also available:

• **Forms Authentication**: This ASP.NET authentication method supports forms-based authentication against pluggable credentials stores via the ASP.NET Membership service.

Developers can also provide custom authentication features developed with the new IIS 7.0 native module API or with ASP.NET APIs for applications using the Integrated mode. In fact, applications running in Integrated mode can use most existing custom ASP.NET authentication modules immediately to provide site-wide authentication.

One or more authentication methods can be configured for Web site, application, or part thereof to protect it with user-based authorization, enable impersonation for resource access, or allow for application personalization.

*Note: IIS 7.0 requires that each request is authenticated. Because of this, at least one authentication method must be enabled and be able to provide an authenticated user for each request.*

### 7.3.4.1 Anonymous Authentication

Anonymous authentication enables clients to access public areas of Web site without requiring the client to provide any credentials. Anonymous authentication is the default authentication method enabled in IIS 7.0. It is part of the default IIS install and is enabled by default. By installing or uninstalling the AnonymousAuthenticationModule module, it can be installed or uninstalled manually.

---

Anonymous authentication applies for all requests that do not have an authenticated user identity determined by other authentication methods. It works by setting the authenticated user identity for such requests to be a Windows identity corresponding to the configured anonymous user account.

By default, anonymous authentication is configured to use the new built-in IUSR account. It no longer uses the custom IUSR_ComputerName account that is used by default with anonymous authentication in IIS 6.0. Because IUSR is a built-in account, it does not have a password that must be periodically changed or synchronized between multiple servers. In addition, because it is built in, the IUSR account has the same SID on all machines. Therefore, ACLs that reference it remain valid when copied from one IIS 7.0 server to another.

When using anonymous authentication, following options are available:

- **Use the built-in IUSR account**: This is the default.
- **Use a custom account**: A custom account can be configured that should be used for anonymous requests instead of the IUSR account.
- **Use the application pool identity**: Anonymous authentication can be configured to use the identity of the IIS worker process (application pool identity) instead of a separate anonymous account.

The application pool identity option can be used to simplify resource access management. This ensures that that resource access is always made under the application pool identity, both when the Web server accesses application resources using the application pool identity and when the Web server or application access resources while impersonating the authenticated user. This way, access rights for a single identity needs to be managed.

Use ‘IIS Manager’ to enable or disable anonymous authentication and set the anonymous user options. Select the desired node in the tree view and double-click Authentication. Then, select Anonymous Authentication in the list and use the Enable, Disable, and Edit commands in the Actions pane to configure it.

Alternatively, set anonymous authentication configuration directly; use Appcmd.exe from the command line, or use configuration APIs to configure the `system.webServer/security/anonymousAuthentication` section, by using the following syntax:

```
```

The parameters of this command are shown in Table 7.9.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConfigurationPath</td>
<td>The configuration path at which to set the specified configuration. If this parameter is specified, the <code>/commit:apphost</code> parameter also need to be specified to avoid locking errors when applying configuration to Web site or URL levels.</td>
</tr>
<tr>
<td>enabled</td>
<td>Whether to enable or disable anonymous authentication.</td>
</tr>
<tr>
<td>username</td>
<td>The user name to use for anonymous authentication. Set to “” to use the application pool identity. Default is IUSR.</td>
</tr>
<tr>
<td>password</td>
<td>The password to use when specifying a custom account for anonymous authentication.</td>
</tr>
<tr>
<td>logonMethod⁸</td>
<td>The logon method to use for the anonymous user. Allowed values are Interactive, Batch, Network, ClearText. Default is ClearText.</td>
</tr>
</tbody>
</table>

Table 7.9: Parameters to Set Anonymous Authentication and Anonymous User Options

7.3.4.2 Basic Authentication

Basic authentication implements the Basic Authentication protocol, a standard HTTP authentication scheme defined in RFC 2617 and supported by most HTTP client software. It enables the client to pass both the user name and the password in clear text, and it uses these credentials to log on locally at the Web server or the Web server’s domain. The credentials, therefore, must correspond to a valid local or domain account, and they result in the request being authenticated with a Windows token corresponding to this account.

Basic authentication is not part of the default IIS install. It can be installed manually from the Security feature category through Windows Features On and Off on Windows Vista or from the Security role service category of the Web Server (IIS) role in Server Manager on Windows Server 2008.

Basic authentication is a challenge-based authentication scheme. When a client makes the initial request to a resource that requires authentication, and basic authentication is enabled, the request will be rejected with a 401 unauthorized status that will include a “WWW-Authenticate:basic” response header. If the client supports basic authentication, it will usually prompt the user for credentials and then reissue the request with the credentials included. The basic authentication module will see that credentials are present on the subsequent request and attempt to authenticate the request by logging on with those credentials. The client will typically send these credentials again on every request to the same URL or any URL that is below the URL included in the initial authenticated request.

Note: Just enabling basic authentication does not mean that authentication is required for application. Either disable anonymous authentication and/or configure URL authorization rules or NTFS permissions that deny access to the anonymous user.

Basic authentication is not secure because it passes the credentials in clear text, and therefore may enable an attacker to steal them by eavesdropping on the request packets at the network level. This can be mitigated by using SSL to secure the communication channel between the client and the server. If SSL is used to protect all requests that include the credentials, basic authentication may be a secure option.

Because basic authentication performs the logon locally at the Web server, the resulting Windows token can be used to access resources on a remote server without configuring delegation or Protocol Transition.

By default, basic authentication caches the logon token for the corresponding user name and password in the token cache. During this time, the token may be available inside that process. If the worker process is compromised, malicious code can use this token to elevate privileges if the token represents a user with high privileges. If the code in the process is not trusted, either disable token caching by uninstalling the token cache module or reduce the amount of time the tokens are cached by setting the `HKLM\SYSTEM\CurrentControlSet\Services\inetInfo\Parameters\UserTokenTTL` value to the number of seconds to cache tokens for.

Use ‘IIS Manager’ to enable or disable basic authentication and set the logon method options. Select the desired node in the tree view and double-click Authentication. Then, select Basic Authentication from the list and use the Enable, Disable, and Edit commands in the Actions pane to configure it.

Alternatively, set basic authentication configuration directly; use Appcmd.exe from the command line, or use configuration APIs to configure the `system.webServer/security/basicAuthentication` section, by using the following syntax:
The configuration path at which to set the specified configuration. If this parameter is specified, the "/commit:apphost" parameter also need to be specified to avoid locking errors when applying configuration to Web site or URL levels.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConfigurationPath</td>
<td>The configuration path at which to set the specified configuration. If this parameter is specified, the &quot;/commit:apphost&quot; parameter also need to be specified to avoid locking errors when applying configuration to Web site or URL levels.</td>
</tr>
<tr>
<td>enabled</td>
<td>Whether to enable or disable basic authentication.</td>
</tr>
<tr>
<td>realm</td>
<td>The basic authentication realm that will be indicated to the client for informational purposes. The Web server does not use the realm during the logon process.</td>
</tr>
<tr>
<td>defaultLogonDomain</td>
<td>The domain that will be used by the server to log on using the credentials provided by the client. If the client user name specifies the domain, it will be used instead. If empty, the computer domain is used. The default value is &quot;&quot;.</td>
</tr>
<tr>
<td>logonMethod</td>
<td>The logon method to use for the logon. Allowed values are Interactive, Batch, Network, and ClearText. Default is ClearText.</td>
</tr>
</tbody>
</table>

Table 7.10: Parameters for Setting Basic Authentication Configuration Directly

### 7.3.4.3 Digest Authentication

The Digest Authentication feature implements the Digest Authentication protocol, a standard HTTP authentication scheme defined in RFC 2617 and supported by some HTTP client software. Unlike basic authentication, the client sends an MD5 hash of the user name and the password to the server so that the real credentials are not sent over the network. The Digest Authentication scheme in IIS 7.0 was known as the Advanced Digest Authentication in IIS 6.0 (IIS 7.0 no longer supports the IIS 6.0 Digest Authentication). If successful, Digest Authentication authenticates the request with a Windows token corresponding to the user’s Active Directory account.

Digest authentication is not part of the default IIS install. It can be installed manually from the Security feature category through Windows Features On and Off on Windows Vista or through the Security role service category of the Web Server (IIS) role in Server Manager on Windows Server 2008.

Like basic authentication, digest authentication is a challenge-based authentication scheme. When a client makes the initial request to a resource that requires authentication, and digest authentication is enabled, the request will be rejected with a 401 unauthorized status that includes a "WWW-Authenticate: digest" response header containing additional information required by the Digest Authentication scheme. If the client supports digest authentication, it will usually prompt the user for the credentials and then reissue the request with the hash of the credentials and the nonce information in the challenge. The Digest Authentication module will see that the hash is present on the subsequent request and attempt to authenticate the hash by comparing it with the hash stored in Active Directory. The client will typically send the hash information again on every request to the same URL or any URL below the URL used in the initial authenticated request.

The Web server and the clients accessing it must meet the following requirements to use IIS 7.0 Digest Authentication:

- Both the Web server and the clients using the application must be members of the same domain, or the client must be a member of a domain trusted by the Web server.
• The clients must use Microsoft Internet Explorer 5 or later.
• The user must have a valid Windows user account stored in Active Directory on the domain controller.
• The domain controller must be using Windows Server 2003 or Windows Server 2008.

Unlike the IIS 6.0 Digest Authentication, the IIS 7.0 Digest Authentication does not require the application pool identity to be LocalSystem. In fact, never use LocalSystem or any other identity with Administrative privileges on the server as an application pool identity.

Note: Just enabling digest authentication does not mean that authentication is required for the application. Either disable anonymous authentication and/or configure URL authorization rules or NTFS permissions that deny access to the anonymous user.

Unlike basic authentication, the authenticated token is not suitable for accessing remote resources, and it requires Constrained Delegation or Protocol Transition to be configured to do so.

Use IIS Manager to enable or disable digest authentication. Select the desired node in the tree view and double-click Authentication. Then, select Digest Authentication from the list and use the Enable, Disable, and Edit commands in the Actions pane to configure it.

Alternatively, set digest authentication configuration directly; use Appcmd.exe from the command line, or use configuration APIs to configure the system.webServer/security/digestAuthentication section, by using the following syntax:

```
%systemroot%\system32\inetsrv\Appcmd set config [ConfigurationPath] /section:system.webServer/security/digestAuthentication [/enabled:bool] [/realm:string]
```

The parameters of this command are shown in the Table 7.11

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConfigurationPath</td>
<td>The configuration path at which to set the specified configuration. If this parameter is specified, the “/commit:apphost” parameter also need to be specified to avoid locking errors when applying configuration to Web site or URL levels.</td>
</tr>
<tr>
<td>enabled</td>
<td>Whether to enable or disable digest authentication.</td>
</tr>
<tr>
<td>realm</td>
<td>The digest authentication realm that will be used as specified in the RFC 2617.</td>
</tr>
</tbody>
</table>

Table 7.11: Parameters for Setting Digest Authentication

7.3.4.4 Windows Authentication

Windows Authentication is best suited for intranet environments. This authentication scheme enables Windows clients to authenticate with two Windows authentication protocols, NTLM (NT LAN Manager) and Kerberos. Both of these schemes involve a cryptographic exchange between the client and the server to authenticate the client.

Windows Authentication is not part of the default IIS install and is not enabled by default. It can be installed manually from the Security feature category through Turn Windows Features On And Off on Windows Vista or via the Security role service category of the Web Server (IIS) role in Server Manager on Windows Server 2008.
Windows Authentication, similar to other IIS authentication methods, is challenge-based. When a request is rejected with a 401 unauthorized response status code, Windows Authentication will issue a WWW-Authenticate challenge header including one or both of the following authentication scheme names:

- **NTLM**: It indicates to the client that it can use the NTLM authentication protocol to authenticate. This is included for older clients that do not support the negotiate wrapper.
- **Negotiate**: It indicates to the client that it can use Kerberos or NTLM protocols to authenticate. Negotiate is used to allow either Kerberos or NTLM authentication, depending on what is available on the client.

Both Kerberos and NTLM authentication methods involve the client making several (typically two to three) requests to the server as part of the authentication handshake. This means that modules may see multiple requests as part of the authentication process. By default, authentication occurs once per connection, so it does not occur again for subsequent requests using the same connection.

The client then makes the decision to use either Kerberos (if available) or NTLM and initiates a sequence of requests to authenticate using the selected protocol. The choice of protocol is based on whether or not the client is configured to be able to use Kerberos to authenticate with the server, which requires a direct connection to the Key Distribution Center (KDC) on the domain controller as well as direct access to Active Directory. NTLM can be used in a non-domain scenario against local Windows accounts on the server or when the connection to domain services required for Kerberos is unavailable.

Windows Authentication is a reasonable choice for Windows-based intranet environments, but for other environments, consider the following limitations:

- It does not work over HTTP proxies. This is because Kerberos and NTLM are connection-based, and proxies may not keep connections open or may share connections between requests from multiple clients.
- The Kerberos protocol requires both the client and the server to be members of the same domain or two domains with a trust relationship and have a direct connection to Active Directory and the KDC services located on the domain controller.
- The Kerberos protocol requires correct Service Principal Name (SPN) registration in Active Directory for all application pools performing Kerberos authentication.

7.3.4.5 Configuring Windows Authentication

Use IIS Manager to enable or disable Windows Authentication. Select the desired node in the tree view and double-click Authentication. Then, select Windows Authentication in the list and use the Enable, Disable, and Edit commands in the Actions pane to configure it.

Alternatively, it can be configured by using Appcmd.exe from the command line, or use configuration APIs to configure the `system.webServer/security/windows-Authentication` section and using the following syntax:

```plaintext
```
The parameters of this command are shown in Table 7.12

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConfigurationPath</td>
<td>The configuration path at which to set the specified configuration. Additionally, the /commit:apphost parameter also need to be specified to avoid locking errors when applying configuration to Web site or URL levels.</td>
</tr>
<tr>
<td>enabled</td>
<td>Whether to enable or disable Windows Authentication.</td>
</tr>
<tr>
<td>authPersistSingleRequest</td>
<td>Whether or not to require each new request to reauthenticate. If set to false, the client will perform the authentication handshake only once per connection, and the server will cache the authenticated identity for all subsequent requests. Otherwise, each request will require the authentication handshake. Default is false.</td>
</tr>
<tr>
<td>authPersistNonNTLM</td>
<td>Whether to require each new request to reauthenticate when using Kerberos. If set to false, the client will perform the authentication handshake only once per connection, and the server will cache the authenticated identity for all subsequent requests. Otherwise, each request will require the authentication handshake. Default is false.</td>
</tr>
<tr>
<td>useKernelMode</td>
<td>Whether to perform Windows Authentication in the kernel. The default is true.</td>
</tr>
<tr>
<td>useAppPoolCredentials</td>
<td>Whether to use the application pool identity instead of LocalSystem when performing kernel Windows Authentication. This is needed when using a domain account as the application pool identity to enable Kerberos authentication on a Web farm. The default is false.</td>
</tr>
</tbody>
</table>

Table 7.12: Parameters for Configure Authentication

In addition, the server can be controlled to use NTLM or Negotiate protocols. To do this, edit the providers collection in the system.webServer/security/windowsAuthentication configuration section. By default, this collection contains both NTLM and Negotiate protocol providers. The server can be forced to use only NTLM by removing the Negotiate protocol provider. However, the server cannot be forced to use only Kerberos in this configuration, because the negotiate wrapper enables the client to use either NTLM or Kerberos. There is no way to tell the client that only Kerberos is supported.

However, configure the NTLM authentication level by using the Local Security Policy console and modifying the Security Settings\Local Policies\Security Options\Network Security: LAN Manager Authentication Level option, as shown in Figure 7.4. The default setting is Send NTLMv2 Response Only, which enables the server to accept all levels. Refuse LM & NTLM for maximum security while allowing clients that do not have the ability to use Kerberos to use the NTLMv2 scheme.
In IIS 6.0, to use the Kerberos authentication protocol, the Setspn.exe command line tool have to be used to register Service Principal Names (SPNs) in Active Directory for the NetBIOS and the Fully Qualified Domain Name (FDQN) names for each application pool account. Additionally, only one application pool account could have been registered for each SPN, preventing multiple application pools with different identities from using Kerberos authentication.

In IIS 7.0, kernel-based Windows Authentication (enabled by default) offers improved functionality. Because HTTP.sys performs the authentication process in the kernel, it is done under the LocalSystem account regardless of the application pool identity. This results in the following improvements:

- It should no longer be necessary to configure separate SPNs, because Kerberos will use the default NetBIOS SPN entry created automatically when the Web server computer is joined to the domain.
- Application pool identity can be changed without the need to reregister the SPN with the new account. The application pool account no longer needs to be a domain account.
- Multiple application pools can use Kerberos authentication.

These changes make it significantly easier to deploy and use the Kerberos protocol with IIS.
Note: The application pool identity need to be used and register SPNs for Kerberos authentication when using it on a Web farm.

However, if using IIS on a Web farm and require the Kerberos protocol, the use of the LocalSystem identity for Kerberos authentication need to be disabled by setting the useAppPoolCredentials attribute in the system.webServer/security/authentication/windows-Authentication configuration section to true. In addition, use a domain account as an identity for the application pool and it is also required to use Setspn.exe⁹ to register the Web site host name using this domain account under which the application pools are configured to run in Active Directory.

7.3.4.6 Client Certificate Mapping Authentication

Client Certificate Mapping Authentication enables clients to authenticate with the Web server by presenting client certificates over Secure Socket Layer (SSL) connections.

The Client Certificate Mapping Authentication uses the Directory Services Mapper (DS Mapper) service in Active Directory to map client certificates provided by the user to domain accounts. IIS also provides a custom certificate mapping feature, the IIS Client Certificate Mapping Authentication, which allows for more flexible mapping of client certificates to Windows accounts.

Client Certificate Mapping Authentication is not part of the default IIS install and is not enabled by default. It can be installed manually from the Security feature category through Turn Windows Features On And Off on Windows Vista or via the Security role service category of the Web Server (IIS) role in Server Manager on Windows Server 2008.

To use Client Certificate Mapping Authentication, the following requirements must be met:

- The Web server must be a member of a Windows domain.
- Client certificates must be issued to users by using a Certificate Authority (CA) trusted by the Web server.
- Each client certificate must be mapped to a valid domain account in Active Directory.

To enable Client Certificate Mapping Authentication on the Web server, perform the following steps (after installing the Certificate Mapping Authentication module).

1) **Enable Client Certificate Mapping Authentication.** Use IIS Manager by clicking the server node, double-clicking Authentication, selecting Active Directory Client Certificate Authentication, and clicking Enable in the Actions pane. Note that this can only be done at the server level when using IIS Manager, although Client Certificate Mapping Authentication can be enabled for a specific URL through configuration.

2) **Configure SSL on each Web site using this authentication method.** Certificate authentication is possible only if the Web site is being accessed over an SSL connection and therefore requires an SSL binding to be configured for the Web site.

3) **Enable DS Mapper for each Web site SSL binding.** IIS Manager does this automatically for each Web site when the Client Certificate Mapping Authentication is enabled and an SSL binding for the Web site is being added. To do this manually, use the Netsh.exe command with the following syntax:

   ```
   netsh http add sslcert IP Address:Port dsmapperusage=enable
   ```

---

where IP Address and Port are the IP address and port of the corresponding binding.

4) **Configure each Web site using this authentication method to accept client certificates** (and possibly require them). This ensures that the server accepts client certificates when provided by the client and can also configure the server to require the client to present a certificate to proceed with the request.

Alternatively, Client Certificate Mapping Authentication can also be enable by editing the `system.webServer/security/authentication/clientCertificateMappingAuthentication` configuration section directly or by using Appcmd or other configuration APIs and by using the following Appcmd syntax.

```
%systemroot%\system32\inetsrv\Appcmd set config /section:
    system.webServer/security/authentication/clientCertificateMappingAuthentication /enabled:true
```

The `enabled` attribute specifies whether or not the Client Certificate Mapping Authentication is enabled. This method can be enabled for a specific URL. However, the decision to use the Directory Services Mapper to map certificates to Windows domain accounts is dependent on each Web site binding having been configured to use the HTTP.sys DS Mapper setting.

### 7.3.4.7 IIS Client Certificate Mapping Authentication

IIS Client Certificate Mapping Authentication enables clients to authenticate with the Web server by presenting client certificates over Secure Socket Layer (SSL) connections.

The IIS Client Certificate Mapping Authentication provides a more flexible mechanism for authenticating clients based on client certificates than does the Active Directory–based Client Certificate Mapping Authentication. Instead of relying on the Directory Services Mapper (DS Mapper) service to map client certificates to Windows accounts, it uses the configuration to perform the mapping. As such, it also does not require the user accounts to be domain accounts and does not require Active Directory to operate.

IIS Client Certificate Mapping Authentication is not part of the default IIS install and is not enabled by default. It can be installed manually from the Security feature category through Turn Windows Features On And Off on Windows Vista or via the Security role service category of the Web Server (IIS) role in Server Manager on Windows Server 2008. After the module is installed, explicitly enable IIS Client Certificate Mapping Authentication for it to be available.

The IIS Client Certificate Mapping Authentication feature supports the following mapping types:

- **One-to-one mapping**: Map a single client certificate to a specific Windows account. The server will use an exact copy of the client certificate to perform the match and therefore must possess a copy of each client certificate.
- **Many-to-one mapping**: Map client certificates to a Windows account by matching wildcard expressions involving specific certificate fields, such as issuer or subject. This does not require a copy of the client certificate.

To use IIS Client Certificate Mapping Authentication, the following requirements must meet:

- Active Directory–based Client Certificate Mapping Authentication cannot be used on any of the sites for which IIS Client Certificate Mapping Authentication is enabled.
- All Windows accounts used to map certificates must have passwords. Unlike Client Certificate Mapping Authentication, which relies on Active Directory to generate the Windows token for the account, the user name and password for each account being mapped must be specified so that IIS can generate the token.
To use one-to-one mappings, an exact copy of each client certificate being mapped is needed. When using many-to-one mappings, copies of the certificates are not needed.

To enable IIS Client Certificate Mapping Authentication for a specific Web site or URL, following steps need to be performed (after installing the IIS Certificate Mapping Authentication module):

1) **Enable IIS Client Certificate Mapping Authentication**: This option is not available in IIS Manager, edit the `system.webServer/security/authentication/iisClientCertificateMappingAuthentication` section directly. Alternatively, use Appcmd or another configuration API and use the following syntax:

   ```
   %systemroot%\system32\inetsrv\Appcmd set config [ConfigurationPath]
   /section:system.webServer/security/authentication/
   iisClientCertificateMappingAuthentication /enabled:true /commit:apphost
   ```

2) **Configure SSL on each Web site using this authentication method**: Certificate authentication is possible only if the Web site is being accessed over an SSL connection. Therefore, it requires an SSL binding to be configured for the Web site.

3) **Configure each Web site using this authentication method to accept client certificates** (and possibly require them). Doing so ensures that the server accepts client certificates when the clients provide them. And also configure the server to require the client to present a certificate to proceed with the request.

4) **Configure the required one-to-one or many-to-one mappings**: Create the mappings to map certificates to Windows accounts.

**Note**: Although the IIS Client Certificate Mapping Authentication feature can be enabled for a specific URL, the mapping configuration can only be set at the server or Web site level, and it is ignored if it is set at a lower configuration level.

### 7.3.4.8 Creating One-to-One Certificate Mappings

Use one-to-one certificate mappings as part of a strong authentication and authorization scheme to control access to application resources based on the exact identity of the client. It can be used instead of a user name and password authentication scheme that requires the user to supply credentials. To use one-to-one mappings, the exact copy of each certificate is needed.

IIS Manager does not provide support for configuring one-to-one mappings. It can be configured either by using the Appcmd command line tool or by editing the `system.webServer/security/authentication/iisClientCertificateMappingAuthentication` configuration section directly or with other configuration APIs. One-to-one mapping can be added by using the following Appcmd syntax:

```
%systemroot%\system32\inetsrv\Appcmd set config [SiteName]
/section:system.webServer/security/authentication/
 iisClientCertificateMappingAuthentication /+oneToOneMappings
 [certificate='string',enabled='bool',username='string',password='string']
```

One-to-one mapping can be removed by using the following Appcmd syntax.

```
%systemroot%\system32\inetsrv\Appcmd set config [SiteName]
/section:system.webServer/security/authentication/
 iisClientCertificateMappingAuthentication /-oneToOneMappings
 [certificate='string']
```
These commands use the parameters in Table 7.13

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SiteName</td>
<td>The site name of the Web site for which to set these settings. If omitted, this parameter sets them for the entire Web server. If these settings are being specified for a configuration path deeper than the Web site root, these settings will not take effect.</td>
</tr>
<tr>
<td>certificate</td>
<td>The exact text of the certificate (not the certificate hash).</td>
</tr>
<tr>
<td>enabled</td>
<td>Whether or not this mapping is enabled.</td>
</tr>
<tr>
<td>userName</td>
<td>The user name for the account to which the certificate maps.</td>
</tr>
<tr>
<td>password</td>
<td>The password for the account to which the certificate maps. This value is stored in the encrypted form.</td>
</tr>
</tbody>
</table>

Table 7.13: Parameters for Creating Certificate Mappings

The exact text of the certificate can be obtained from an exported certificate file (containing unencrypted certificate information) or by dumping the certificate from the local or domain certificate store. To do the latter, use the following command.

```bash
certutil --encode -f CertName cert.cer
```

CertName is the friendly name of the certificate. View the certificate store and obtain the friendly name of the installed certificates with the following command.

```bash
certutil --viewstore StoreName
```

StoreName is the name of the certificate store. Use MY for the personal certificate store.

*Note: Specify the exact base64 encoded certificate contents for the one-to-one mapping, with the training line feed removed. Do not use the certificate hash. If the certificate is not specified correctly, a 401.1 status error will be raised when making requests to the Web site. This error will show the 0x8009310b HRESULT, indicating that IIS failed to load the certificate from the mapping entry.*

### 7.3.4.9 Creating Many-to-One Certificate Mappings

Many-to-one mappings, unlike one-to-one mappings, are not typically used to authenticate specific users. Instead, it can be used to authenticate a group of users by matching fields in their certificates to a single Windows account. Therefore, authorization based on the authenticated user produced by a many-to-one mapping is similar to role- or group-based authorization, with the authenticated user representing a group to which multiple users belong. For example, all certificates issued by a specific organization can be matched to that organization’s account. As such, many-to-one mappings may be less appropriate for user-based personalization or access control than one-to-one mappings, depending on authorization strategy.

*Note: One-to-one mappings are always processed before many-to-one mappings.*

Many-to-one mappings do not require the server to have the exact certificate for each user. Instead, simply configure wildcard rules based on one or more fields in the certificate that map all certificates with matching fields to a Windows account.

IIS Manager does not provide support for configuring many-to-one mappings; it can be configured by using the **Appcmd** command line tool. Alternatively, by editing the
system.webServer/security/authentication/iisClientCertificateMappingAuthentication configuration section directly or with other configuration APIs and by using the following Appcmd syntax:

```
%systemroot%\system32\inetsrv\Appcmd set config [SiteName] /section:system.webServer/security/authentication/iisClientCertificateMappingAuthentication /+manyToOneMappings [name='string',enabled='bool',permissionMode='enum',
username='string',password='string',description='string']
```

Then, add one more matching rule to the mapping by using the following Appcmd syntax, specifying the name of the mapping created in the command shown previously.

```
%systemroot%\system32\inetsrv\Appcmd set config [SiteName] /section:system.webServer/security/authentication/iisClientCertificateMappingAuthentication /+manyToOneMappings [name='string'].rules.[certificateField='enum',
certificateSubField='string',matchCriteria='string',
compareCaseSensitive='bool']
```

To delete a mapping by using the following Appcmd syntax:

```
%systemroot%\system32\inetsrv\Appcmd set config [SiteName] /section:system.webServer/security/authentication/iisClientCertificateMappingAuthentication /-manyToOneMappings [name='string']
```

These commands use the parameters in Table 7.14

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SiteName</td>
<td>The site name of the Web site for which to set these settings. If omitted, this parameter sets them for the entire Web server. If these settings are being specified for a configuration path deeper than the Web site root, these settings will not take effect.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the mapping; can also be used to add rules to it or delete it.</td>
</tr>
<tr>
<td>enabled</td>
<td>Whether or not this mapping is enabled.</td>
</tr>
<tr>
<td>permissionMode</td>
<td>Whether to allow or deny access to the user who is given this mapping.</td>
</tr>
<tr>
<td>userName</td>
<td>The user name for the account to which the certificate maps.</td>
</tr>
<tr>
<td>password</td>
<td>The password for the account to which the certificate maps. This value is stored in the encrypted form.</td>
</tr>
<tr>
<td>description</td>
<td>The friendly description of the mapping.</td>
</tr>
<tr>
<td>certificateField</td>
<td>The certificate field to match in the current rule. Common fields are Issuer and Subject. For more information, get the details about the contents of the certificate from the CA.</td>
</tr>
<tr>
<td>certificateSubField</td>
<td>The certificate subfield to match in the current rule. For more information on the subfields, get the details about the contents of the certificate from the CA.</td>
</tr>
<tr>
<td>matchCriteria</td>
<td>The match criteria. This can include * and ? wildcard matching characters.</td>
</tr>
<tr>
<td>compareCaseSensitive</td>
<td>Whether or not the comparison should be case-sensitive.</td>
</tr>
</tbody>
</table>

Table 7.14: Parameters for Creating Certificate Mappings
7.3.4.10 UNC Authentication

The Web server core uses UNC authentication to establish an identity for accessing remote application content inside virtual directories that reside on a UNC share. It is not a true authentication method in the sense that it does not itself support an authentication scheme for establishing the identity of the client. Rather, it is a mechanism for using the authenticated user that has been established through other authentication mechanisms—and in some cases a fixed identity set in configuration—to determine which identity should be used for remote content access.

IIS uses UNC authentication whenever a request is made to a resource that resides in a virtual directory whose physical path is located on a UNC share (whether or not the UNC share is on the local computer). During UNC authentication, the Web server determines the identity to be used for accessing remote content as follows:

1) **Uses the virtual directory's fixed credentials:** In IIS 7.0, any virtual directory can specify fixed credentials that IIS uses for all accesses to that location. This replaces the `UNCUserName` and `UNCPassword` properties in IIS 6.0 that were used only when the virtual directory referred to a UNC share.

2) **Otherwise, uses the authenticated user if available:** If the virtual directory does not specify fixed credentials, use the authenticated user if it has already been determined by an authentication method. This is referred to as pass-through authentication. It cannot be used to access web.config files, because this access occurs before IIS determines the authenticated user.

3) **Otherwise, uses process identity:** If IIS has not yet determined the authenticated user, it will use the identity of the IIS worker process. The Web server uses this option to access web.config files (if virtual directory credentials are not configured), because configuration is read prior to the authentication stage.

*Note: IIS 7.0 cannot use pass-through authentication to access web.config files located on the remote UNC share. Because of this, the virtual directory must specify fixed credentials, or the application pool identity must have Read access to the remote UNC share.*

By default, IIS cannot access remote UNC content. This is because the default anonymous user IUSR is a local built-in account that does not have network privileges. Additionally, because IIS is required to access web.config by using the IIS worker process identity, it has a similar problem because the Network Service account also does not typically have the right to access remote resources. Thus, there are typically two options for configuring UNC authentication to allow proper access of remote content:

- **Use pass-through authentication:** Pass-through authentication requires both the application pool identity and all allowed authenticated user identities to have access to the remote UNC share. Additionally, it requires the use of an authentication scheme that is capable of delegating the user identity to a remote computer or configuring Constrained Delegation and/or Protocol Transition to enable this for other authentication schemes.

- **Use virtual directory fixed credentials:** This is the recommended approach, because it requires to grant access to the share for a single identity. Also, it does not have the requirement of ensuring that the authentication scheme can delegate its identities to the remote UNC share, because the fixed identity is always used to access the content. However, the fixed credential model does not enable the use of NTFS authorization and auditing for authenticated users accessing the share, because the access is always made under the specified credentials and not the authenticated user identity.

*Note: It is highly recommended to use the fixed credential model to configure access to remote UNC shares. Use this in all cases when NTFS ACL-based access control or auditing of remote content for authenticated users cannot be relied on.*
Following steps need to be taken to use pass-through authentication:

- Use a custom application pool identity that has access to the UNC share.
- If using anonymous authentication, configure the anonymous user to be the application pool identity. Alternatively, configure a custom anonymous user that has access to the UNC share.
- If using other authentication methods that produce Windows identities, ensure that these methods can delegate identities to the UNC share. Then, ensure that all authenticated users have access to the UNC share.

### 7.3.4.11 Understanding Authentication Delegation

Many IIS authentication methods produce Windows identities that can be impersonated for the purpose of accessing resources. When the resources being accessed reside on a remote machine, the authenticated user identity needs to be transmitted to the remote machine for authentication with the remote service. This process is referred to as delegation. It occurs when IIS attempts to access files located on remote UNC shares, or when the application impersonates the authenticated user identity to connect to a remote server such as SQL Server.

Most IIS authentication methods do not produce authenticated identities that are suitable for delegation. This means that when IIS is configured to use these authentication methods, IIS and the application may fail to access resources located on remote machines when impersonating the authenticated identity.

*Note: In general, the rule for remembering which authentication methods can be delegated is to remember which authentication methods perform their logon locally on the Web server. For example, any authentication scheme in which the user name and password are available on the Web server—such as Basic Authentication, IIS Client Certificate Mapping Authentication, or Anonymous Authentication—use the Web server to log on and therefore can delegate authenticated identities.*

Following three options are available to ensure that application has access to its backed resources located on remote servers:

- Use an authentication method that supports delegation (see Table 7.15).
- Use fixed virtual directory credentials to create an authentication identity that can be delegated to and can be impersonated instead of the authenticated user.
- Configure Constrained Delegation and Protocol Transition to upgrade the authenticated identity to an authenticated identity which can be delegated to use the Kerberos protocol.

Table 7.15 lists the built-in IIS authentication schemes and the required configuration to enable delegation of authenticated identities.

<table>
<thead>
<tr>
<th>Authentication</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anonymous</td>
<td>Delegates when using a custom anonymous user or when using a custom application pool identity as the anonymous user (1 hop)</td>
</tr>
<tr>
<td>Basic</td>
<td>Delegates by default (1 hop)</td>
</tr>
<tr>
<td>Windows (Kerberos)</td>
<td>Requires Constrained Delegation</td>
</tr>
<tr>
<td>Windows (NTLM)</td>
<td>Requires Constrained Delegation and Protocol Transition</td>
</tr>
<tr>
<td>Digest</td>
<td>Requires Constrained Delegation and Protocol Transition</td>
</tr>
<tr>
<td>Client certificate mapping</td>
<td>Requires Constrained Delegation and Protocol Transition</td>
</tr>
<tr>
<td>IIS Client certificate mapping</td>
<td>Delegates by default (1 hop)</td>
</tr>
</tbody>
</table>

*Table 7.15: Built-In IIS Schemes and Required Configuration to Enable Delegation of Authenticated Identities*
Most of the authentication schemes that do not perform the logon locally on the machine require Constrained Delegation and Protocol Transition\(^{10}\) to be able to delegate the authenticated identity to a remote machine. Constrained Delegation refers to the ability of a service to use a user identity obtained using the Kerberos protocol to access remote resources. Protocol Transition, used in conjunction with Constrained Delegation, enables other authentication schemes to obtain a Kerberos identity to be used with Constrained Delegation to access remote resources.

7.4 Securing Web Content

The two main components of Web security are the security of the underlying server application and OS, and the security of the actual content. Of these, the security of the content is often overlooked. As organizations have gotten better at protecting and hardening their network perimeters, OSs, and Web servers, attackers have increasingly turned to exploiting vulnerabilities in Web applications and the way information is processed on Web servers. These application layer attacks exploit the interactive elements of Web sites.

Organizations should not use public Web servers to host sensitive information intended to be accessed only by internal users. The compromise of a public Web server often leads to the compromise of such data.

To ensure a consistent approach, an organization should create a formal policy and process for determining and approving the information to be published on a Web server. Such a process should include the following steps:

- Identify information that should be published on the Web
- Identify the target audience (Why publish if no audience exists?)
- Identify possible negative ramifications of publishing the information
- Identify who should be responsible for creating, publishing, and maintaining this particular information
- Create or format information for Web publishing
- Review the information for sensitivity and distribution/release controls (including the sensitivity of the information in aggregate)
- Determine the appropriate access and security controls
- Publish information
- Verify published information
- Periodically review published information to confirm continued compliance with organizational guidelines.

A public Web site should not contain the following information:

- Classified records
- Internal personnel rules and procedures
- Sensitive or proprietary information
- Personal information about an organization’s personnel or users\(^{30}\)
  - Home addresses and telephone numbers
  - Uniquely identifying information, particularly SSNs
  - Detailed biographical material (that could be employed for social engineering)
  - Staff family members
- Telephone numbers, e-mail addresses or general listings of staff unless necessary to fulfill organizational requirements

In some cases, Personally Identifiable Information (PII) is being collected by the organization which is relevant and necessary for the business purpose. In addition to accepted practices (many of which are reflected in laws applicable to both private and public institutions), followings are to provide subject individuals:

- Notice that information about them is being collected, including descriptions of what data is being collected, with whom it is being shared, and what is being done with that data
- Opportunities to opt out of data collection unless the data collection is mandatory under law, necessary to the performance of a contract with the subject individual, or if the individual has freely offered his/her PII
- Opportunities to access and review the records kept about themselves, and to request corrections or additions, especially if that information may be used to make a determination about the individuals’ rights, opportunities, or benefits.

### 7.4.1 Securing Active Content and Content Generation Technologies

Active content refers to interactive program elements downloaded to the client (i.e., a Web browser) and processed there instead of the server. A variety of active content technologies exists; some of the more popular examples are ActiveX, Java, VBScript, JavaScript, and Asynchronous JavaScript and XML (AJAX). The use of active content often requires users to reduce the security settings on their Web browsers for processing to occur. If not implemented correctly, active content can present a serious threat to the end user. For example, active content can take actions independently without the knowledge or expressed consent of the user. While active content poses risk to the client, it can also pose risk to the Web server. The reason is that information processed on the client is under the control of the user, who can potentially manipulate the results by reverse engineering and tampering with the active content. For example, form validation processing done with active content elements on the client side can be changed to return out-of-range options or other unexpected results to the server. Therefore, the results of processing done on the client by elements of active content should not be trusted by the server; instead, the results should be verified by the server. Organizations considering the deployment of client-side active content should carefully consider the risks to both their users and their Web servers.
All Web sites that implement active content and content generators\(^{11}\) should perform additional steps to protect the active content from compromise. Each active content technology has its own strengths and weaknesses; none is perfectly secure. New technologies are emerging all the time and older technologies are continually enhanced. Any Web administrator or Webmaster who is considering deploying a Web site with features that require active content technology on the client side should carefully weigh the risks and benefits of the technology before implementation. The relative risk of active content technologies changes over time, complicating this task. Nevertheless, common risks to the Web server prevail with all active content technologies. Because these technologies involve placing application code on the client, an attacker may attempt to reverse engineer the code to gain an understanding of how it functions with an organization’s Web site and exploit that relationship. For example, relying only on input validation checks performed at the client by the active content and not validating the input again at the Web server would allow an attacker to use an HTTP proxy tool or editor to modify the active content elements and bypass any or all checks performed.

Special caution is also required for downloading preprogrammed scripts or executables from the Internet. In general, no third-party scripts should be installed on a Web server unless they are first subjected to a thorough code review by a trusted expert. Security code reviews should also be considered for content on Web servers that are critical to the organization or are highly threatened.

*Note: For more information see 'Checklist for Securing the Web Content' in Appendix A.*
8. Using Encryption Technologies

8.1 Securing Communications with Secure Socket Layer (SSL)

By default, all communication between the Web server and the client occurs over a clear-text connection, which has the potential to expose the information included in the requests and responses to an attacker able to listen to the communication at the network layer. This includes packet sniffing at a local network, or compromising a router or a proxy server that is located on the path between the client and the Web server. This can result in the unintended disclosure of the response information, which may contain sensitive information, client credentials that are sent as part of some authentication methods (such as basic authentication or forms-based authentication methods), cookies, and more. The attacker can sometimes successfully use this information to misrepresent the client by providing these credentials to the Web server in a replay attack.

To prevent this from happening, use the Secure Socket Layer (SSL) or the newer Transport Layer Security (TLS) protocols to secure the communication between the client and server. TLS is a widely accepted standard that most browser technologies implement. In the rest of this section, both of these protocols collectively will be referred as SSL for ease of reference.

In addition to securing the communication between the client and the Web server, SSL serves to confirm the identity of the Web server to the client. This process is widely used on the Internet today to ensure that the client is dealing with the entity that the Web site claims to represent. IIS can also use it to establish the identity of the client, if the client has an acceptable certificate.

8.1.1 Configuring SSL

To configure SSL, perform the following steps:

- **Obtain a server certificate from a trusted Certificate Authority**: The Certificate Authority (CA) must be a trusted root CA for all of the clients that connect to the Web site that uses this certificate. For intranet sites, this may be a domain CA provided by the Active Directory Certificate Services. For Internet sites, this is usually a CA that is trusted by most client browsers by default. The certificate can be obtained by making a certificate request using the Server Certificates feature in IIS Manager. Alternatively, a self-issued (or self-signed) certificate can be used if organization controls both the Web server and the clients, and if it is intend to use this certificate for testing purposes.

- **Create a secure binding by using the HTTPS protocol and port 443 (or another port), and specify the server certificate for each Web site**: It can be done by creating a binding in IIS Manager, or by adding a binding programmatically and then using the `netsh http add sslcert ipport=IPAddress:443 certstorename=MY certhash=hash appid=GUID` command to associate the certificate with the binding. The certificate hash can be obtained from the Certificates console by viewing the certificate details and copying the value of the Thumbprint property.

*Note: IIS 7.0 stores the certificate information directly in the HTTP.sys configuration. These associations can be manipulated by using IIS Manager or by using the 'netsh http add sslcert' command.*

To be accepted by the clients, the server certificate must contain Common Name (CN) entries for all of the host headers that the Web site used. This needs to be done when the certificate is requested.
It is possible to have multiple SSL Web sites that use unique server certificates if each Web site uses a separate IP address or port for its HTTPS binding. IIS 7.0 does not support having multiple Web sites with their own server certificates if the HTTPS bindings for those Web sites are on the same IP address/port and differentiate only by using host headers. This is because the host header information is not available when the SSL negotiation occurs. Because of this, the only way to have multiple Web sites on the same IP address that use host headers is to configure all of those Web sites to use the same SSL certificate with a wildcard CN. For more information, see http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/596b9108-b1a7-494d-885d-f8941b07554c.mspx?mfr=true

8.1.2 Requiring SSL

To ensure that the communication between Web server and clients is protected, it is required that clients request Web site content over secure connections. This is an effective way to protect clients’ authentication credentials or sensitive cookies issued by the Web site over unsecure connections.

If a Web site enables mixed SSL usage, such as by allowing the Web site to be accessed over both SSL and unsecure connections, or by allowing portions of Web site to be accessed over unsecure connections, be aware that requests made over these connections may leak sensitive information. For example, if the Web site uses Forms authentication to authenticate users, uses cookie-based session state, or stores sensitive information about the user in cookies, then clients to this web-site may leak these cookies when making requests over unsecure connections. Therefore, always prefer to protect the entire Web site with SSL by requiring SSL for the entire Web site’s URL namespace. Also, configure the cookies to include the secure bit to make sure the browser will not attempt to send them over unencrypted connections.

To mandate SSL, use IIS Manager, select the Web server, Web site, or another node corresponding to the URL for which SSL is required. Then double-click SSL Settings. In this feature, select the Require SSL check box to mandate SSL, as shown in Figure 8.1. The option of selecting the Require 128-Bit SSL check box is also there.
Alternatively, configure SSL by editing the `system.webServer/security/access` section directly by using Appcmd or another configuration API. For example, set this configuration using the following Appcmd syntax.

```
%systemroot%\system32\inetsrv\Appcmd set config [ConfigurationPath] /section:system.webServer/security/access /sslFlags:enum
```

This command has the parameters presented in Table 8.1

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConfigurationPath</td>
<td>The configuration path for which to apply this configuration. The <code>/commit:apphost</code> parameter also needs to be specified to avoid locking errors when this configuration is applied at Web site or URL levels.</td>
</tr>
<tr>
<td>sslFlags</td>
<td>A comma-separated list of one or more of the following values: <code>None</code>, <code>Ssl</code>, <code>Ssl128</code>, <code>SslNegotiateCert</code>, <code>SslRequireCert</code>. Set this to <code>Ssl</code> to require SSL, and <code>Ssl,Ssl128</code> to require 128-bit SSL.</td>
</tr>
</tbody>
</table>
8.1.3 Client Certificates

Though SSL is typically used to confirm the identity of the Web server to the client, it can also be used to confirm the client’s identity to the Web server if the client has certificates issued by a CA that the Web server trusts. Client certificates can be used as part of a strong two-factor authentication scheme that requires both a user name/password as well as a physical authentication method to provide the client certificate, such as a Smart Card. Or, it can be used as a single authentication method with one of the client certificate mapping authentication methods that IIS supports.

To use client certificates, following requirements must meet:

- The Web site must be configured to use SSL and have a valid server certificate.
- The client must have a client certificate issued by a CA that the Web server trusts.

When a client makes a connection that uses SSL, the Web server negotiates the client certificates (if configured to do) by indicating the list of trusted CAs on the server, causing the client to respond with the list of certificates that are available on the client and that are issued by those CAs. The server then validates the certificates, including checking their expiration times and making sure that they are not listed on the Certificate Revocation List (CRL) on the Web server.

IIS supports multiple levels of using client certificates:

1) **Negotiate certificates**: This requests the client to provide a client certificate when the request is made, but it does not require the certificate. If the client provides it, the server validates the certificate, and the certificate is made available to the application. To do this, set the `sslFlags` attribute of the `system.webServer/security/access` configuration section to include `SslNegotiateCert`, as described earlier in this chapter.

2) **Require certificates**: This requires that the client provide a client certificate when the request is made. If the certificate is not provided, the request is rejected with a 403.7 – Client Certificate Required error. If the certificate is not successfully validated by the server, the request will be rejected with a 403.16 – Client Certificate Is Untrusted Or Invalid error. It could also be rejected with a 403.17 – Client Certificate Has Expired Or Is Not Yet Valid error. To require certificates, set the `sslFlags` attribute of the `system.webServer/security/access` configuration section to include `SslNegotiateCert, SslRequireCert`, as described earlier in this chapter. The certificates can be required to implement a strong two-factor authentication scheme or in conjunction with a client certificate mapping authentication scheme as the primary authentication scheme for the Web site.

3) **Authenticate users with client certificates**: IIS can also be configured to authenticate clients based on the client certificates.

8.2 Securing Configuration

Previous versions of IIS have used a centralized configuration store known as the metabase. IIS 7.0 abandons the metabase in favor of a new configuration system based on a hierarchy of XML configuration files, in order to provide for simpler deployment and more flexible management of the Web server.

Because the configuration is stored in plain text XML files, some of which may be taken off the server during deployment or otherwise exposed, it is sometimes necessary to protect the information stored therein from being discovered. The configuration system provides built-in encryption support to protect secrets stored in configuration files against disclosure even if an attacker is able to access the file.
The configuration file hierarchy goes beyond centralized configuration and includes distributed web.config configuration files access that can be delegated to the site or application owner. This enables sites and applications to contain required configuration as part of their content for single-step deployment that does not require administrative rights on the server. It also enables site and application owners to manage their applications remotely without having administrative rights.

### 8.2.1 Restricting Access to Configuration

IIS 7.0 configuration is stored in a hierarchy of configuration files, including both server-level configuration files and distributed web.config files that may be delegated to site and application owners. Because these files store configurations by using plain text XML, anyone with the ability to access them can read and/or tamper with a server configuration without using any additional tools or APIs. Therefore, the NTFS access permissions placed on these files determine who can access server configuration and what they can do with it.

To properly secure configuration files, it is important to understand the files that comprise the hierarchy and how they are accessed. This can help to define and maintain the proper access strategy in environment.

The files that constitute the IIS 7.0 configuration hierarchy, and their default access permissions, are listed in Table 8.2.

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>Default Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Framework machine.config</td>
<td>Machine-level .NET Framework configuration</td>
<td>BUILTIN\IIS_IUSRS:(RX)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BUILTIN\Users:(RX)</td>
</tr>
<tr>
<td>Framework root web.config</td>
<td>Machine-level configuration for ASP.NET applications</td>
<td>BUILTIN\IIS_IUSRS:(RX)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BUILTIN\Users:(RX)</td>
</tr>
<tr>
<td>applicationHost.config</td>
<td>IIS machine-level configuration</td>
<td>NT SERVICE\WMSvc:(R)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NT SERVICE\WMSvc:(R)</td>
</tr>
<tr>
<td>&lt;AppoolName&gt;.config</td>
<td>Auto-generated version of applicationHost.config for each application pool</td>
<td>IIS APPPOOL&lt;AppoolName&gt;: (R)</td>
</tr>
<tr>
<td>Distributed web.config (wwwroot)</td>
<td>Delegated configuration files in the Web site directory structure</td>
<td>BUILTIN\IIS_IUSRS:(RX)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BUILTIN\Users:(RX)</td>
</tr>
<tr>
<td>administration.config</td>
<td>Machine-level configuration file for IIS Manager</td>
<td>NT SERVICE\WMSvc:(R)</td>
</tr>
<tr>
<td>redirection.config</td>
<td>Machine-level configuration file for configuring remote location of</td>
<td>NT SERVICE\WMSvc:(R)</td>
</tr>
<tr>
<td></td>
<td>applicationHost.config</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.2: Default Access Permissions for Configuration Files

Note: The default permissions for all entries in Table 14-18 also contain permissions granting full rights to NT AUTHORITY\SYSTEM and BUILTIN\Administrators. These were removed from this table for clarity.

- The server-level configuration files, including Framework machine.config, Framework root web.config, applicationHost.config, administration.config, and redirection.config are writable only by the System and members of the Administrators group.
- All members of the Users group and the IIS_IUSRS group Framework can read machine.config and root web.config files. Unlike IIS server-level configuration files, any user on the machine—as well as any...
application running inside the IIS worker processes—can read the configuration in these files. This is due to the fact that these files are used to configure the behavior of .NET Framework components in any .NET application that runs on the machine.

- The IIS server-level configuration files, applicationHost.config, redirection.config, and administration.config, are only readable by the system, members of the administrators group, and the Web Management Service (NT Service\WMSvc). Unlike .NET Framework configuration files, they cannot be read by non-administrative users or even IIS worker processes. IIS worker processes receive a subset of configuration in the applicationHost.config file from the automatically generated ApppoolName.config files for each application pool.

- The Windows Process Activation Service (WAS) automatically generates the ApppoolName.config files for each application pool, which are readable only by the IIS worker processes in the corresponding application pool.

- The distributed web.config files located in the site directory structure are by default readable by members of the Users group. These files typically must also grant access to the IIS_IUSRS group to allow access to the IIS worker process (IIS setup automatically grants this for the default Web site root located in %SystemDrive%\Inetpub\Wwwroot).

### 8.2.1.1 Setting Permissions on Configuration Files

The configuration files in the IIS hierarchy have restrictive permissions configured by default and should typically not be changed (with the exception of distributed web.config files that are part of the site directory structure). Changes to the permissions on these files may cause these files to become more vulnerable to unauthorized access. Keep the following in mind to maintain the security of these files:

- Never grant non-administrative identities (with the exception of NT SERVICE\WMSvc) access to applicationHost.config, redirection.config, and administration.config (either Read or Write). This includes Network Service, IIS_IUSRS, IUSR, or any custom identity used by IIS application pools. IIS worker processes are not meant to access any of these files directly.

- Never share out applicationHost.config, redirection.config, and administration.config on the network. When using Shared Configuration, prefer to export applicationHost.config to another location.

- All users can read .NET Framework machine.config and root web.config files by default. Do not store sensitive information in these files if it should be for administrator eyes only. Encrypt sensitive information that should be read by the IIS worker processes only and not by other users on the machine.

The only exception to this rule is the distributed web.config files that are part of Web site's directory structure. It is up to administrator to implement ACL on these files correctly to prevent unauthorized access to their contents.

### 8.2.1.2 Understanding Configuration Isolation

The configuration isolation feature provided by IIS 7.0 is always on by default. Instead of enabling IIS worker processes to read applicationHost.config directly when reading the configuration file hierarchy, IIS generates filtered copies of this file and uses these copies as a replacement of applicationHost.config when configuration is read inside the IIS worker process.

The reason for doing this is to prevent IIS worker processes from application pool A to be able to read configuration information in applicationHost.config that is intended for application pool B. Because applicationHost.config may contain sensitive information, such as the user name and password for custom
application pool identities, as well as user name and password for virtual directories, allowing all application pools to access applicationHost.config would break application pool isolation.

WAS is responsible for generating the temporary application pool configuration files that each IIS worker process uses as a replacement of applicationHost.config. These files are placed by default in the %SystemDrive%\inetpub\temp\Apppools directory and are named AppPoolName.config. As mentioned earlier, these files are configured to allow access only to the IIS worker processes in the corresponding application pool, by using the IIS APPPOOL\AppPoolName Application Pool SID.

This process occurs automatically each time applicationHost.config is changed and therefore does not require any manual action from the administrator outside of normal configuration procedures.

Each application pool configuration file contains the configuration in applicationHost.config, with the following information removed:

- All application pool definitions in the system.applicationHost/applicationPools configuration section. Only WAS is required to read this configuration section.
- Any Web site definitions in the system.applicationHost/sites configuration section for sites that do not have applications in the current application pool.
- Any configuration in location tags for specific Web sites, applications, or URLs that do not reside inside the applications in the current application pool.

Note: All application definitions (and their virtual directory definitions, possibly containing user name and password credentials) for any site that has at least one application in the current application pool will be present in the application pool configuration file. To disable this behavior and include only the application definitions for applications in the application pool, set the 'IsolationWholeSiteInclude' DWORD value to 0 in the HKLM\System\CurrentControlSet\Services\WAS\Parameters key and perform an IISRESET. This may break applications in sites with applications in multiple application pools when they attempt to map physical paths for URLs in other applications.

Keep in mind that global configuration settings set in the applicationHost.config (without using location tags to apply them to specific Web sites, applications, or URLs) are not filtered. Each application pool configuration file will contain all of these settings.

Configuration isolation is a key part of the application pool isolation strategy in IIS 7.0. It is enabled by default to provide configuration isolation for server-level configuration in applicationHost.config.

Note: Configuration stored in .NET Framework machine.config and root web.config files is not isolated. Only configuration stored in applicationHost.config is isolated.

8.2.1.3 Setting Permissions for Shared Configuration

IIS 7.0 supports sharing server configuration in the applicationHost.config configuration file between multiple Web servers on a Web farm. Using a shared configuration requires exporting the configuration from the source Web server, placing it on a network share, and configuring the share so that all member Web servers have access to it.

To prevent unauthorized access to the shared configuration, follow these guidelines:
- Do not grant Write access to the identity that the Web server uses to access the shared applicationHost.config. This identity should have only Read access.
- Use a separate identity to publish applicationHost.config to the share. Do not use this identity for configuring access to the shared configuration on the Web servers.
- Use a strong password when exporting the encryption keys for use with shared configuration.
- Maintain restricted access to the share containing the shared configuration and encryption keys. If this share is compromised, an attacker will be able to read and write any IIS configuration for Web servers, redirect traffic from the Web site to malicious sources, and in some cases gain control of all Web servers by loading arbitrary code into IIS worker processes. Consider protecting this share with firewall rules and IPsec policies to allow only the member Web servers to connect.

8.2.2 Securing Sensitive Configuration

The information in the configuration files in the IIS 7.0 configuration hierarchy is protected by the restricted permissions specified by the NTFS ACLs on each file. These permissions should prevent unauthorized users from being able to access these files.

However, this alone may not provide a sufficient level of protection for especially sensitive information stored in configuration files, such as user names and passwords of custom application pool identities. It is essential to prevent this information from being discovered even if an attacker manages to compromise the local Web server and gain access to the configuration file containing the information. In addition, if someone copies the configuration file off the server for archival or transport reasons, an attacker should not be able to read the secrets stored in the configuration file. To ensure this, IIS 7.0 provides the ability to encrypt specific information stored in configuration.

8.2.2.1 Using Configuration Encryption to Store Configuration Secrets

IIS 7.0 configuration encryption works by encrypting the contents of configuration attributes for which encryption is enabled before storing their values in the configuration file. Therefore, even if someone obtains access to the file, they cannot read the contents of the attribute without decrypting it first.

Whether or not configuration encryption is used for each attribute is determined by the attribute's definition in the schema of the containing configuration section. The schema also serves as a mechanism to select the encryption provider used to encrypt the data for this attribute.

When any configuration tool or API writes the value of each encrypted attribute, the configuration system will automatically encrypt it using the configured encryption provider before persisting it to the configuration file on disk.

If the resulting configuration file is inspected, the encrypted value will be seen, as shown in the following code example for the password attribute in the application pool definition inside the system.applicationHost/applicationPools configuration section.

```xml
<applicationPools>
  <add name="MyAppPool">
    <processModel identityType="SpecificUser" userName="TestUser"
      password="[enc:IISWASOnlyAesProvider:N8mr4dLU6PnMW5xImCWg6914ckePeU0ffBxewZppiwytLmBqh0mZnFywQO78pQY:enc]" />
  </add>
</applicationPools>
```
The configuration system decrypts the attribute automatically when it is accessed, provided that the caller of the configuration system has the rights to use the encryption provider used to perform the encryption. Therefore, the decryption and encryption process is completely transparent to the administrator, while ensuring that the resulting configuration is not stored in plain text.

### 8.2.2.2 Selecting Encryption Providers

IIS provides several encryption providers that can be used to encrypt configuration, in addition to several encryption providers provided by the .NET Framework. One of these providers is used for each configuration attribute that is marked for encryption. The providers are listed in Table 8.3.

<table>
<thead>
<tr>
<th>Provider</th>
<th>Use</th>
<th>Encryption Key Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>RsaProtectedConfigurationProvider</td>
<td>Encrypting .NET Framework configuration sections using exportable RSA encryption</td>
<td>RSA machine key: SYSTEM and administrators only; grant access using Aspnet_regiis.exe-pa.</td>
</tr>
<tr>
<td>DataProtectionConfigurationProvider</td>
<td>Encrypting .NET Framework configuration sections using machine-local Data Protection API encryption</td>
<td>By default, everyone on the Web server; optionally user-based key</td>
</tr>
<tr>
<td>IISWASOnlyRsaProvider</td>
<td>Encrypting IIS configuration sections read by WAS using exportable RSA encryption</td>
<td>RSA machine key: SYSTEM and administrators only</td>
</tr>
<tr>
<td>IISWASOnlyAesProvider</td>
<td>Encrypting IIS configuration sections read by WAS using AES encryption</td>
<td>Session key encrypted with RSA machine key: SYSTEM and administrators only</td>
</tr>
<tr>
<td>AesProvider</td>
<td>Encrypting IIS configuration sections read by the IIS worker process using AES encryption</td>
<td>Session key encrypted with RSA machine key: IIS_IUSRS, NT Service\WMSvc</td>
</tr>
</tbody>
</table>

Table 8.3: Configuration Encryption Providers

.NET Framework creates both the RsaProtectedConfigurationProvider and the DataProtectionConfigurationProvider providers. These providers are primarily used to encrypt .NET configuration for ASP.NET applications using the Aspnet_regiis.exe tool.

*Note: IIS configuration encryption cannot be used to encrypt .NET configuration sections. Likewise, .NET configuration encryption cannot be used to encrypt the contents of IIS configuration sections with Aspnet_regiis.exe. An error will be received if attempted to read .NET configuration sections encrypted with .NET configuration encryption by using IIS configuration APIs, because IIS does not support section-level encryption used by the .NET Framework configuration system.*

Use the IIS encryption providers—IISWASOnlyRsaProvider, IISWASOnlyAesProvider, and AesProvider—to encrypt IIS configuration sections.

IISWASOnlyRsaProvider and IISWASOnlyAesProvider are both used to encrypt configuration sections that WAS reads, such as the `system.applicationHost/applicationPools` section, and do not allow IIS worker processes to decrypt the configuration. The IISWASOnlyAesProvider provides better performance because it uses AES encryption using an RSA encrypted session key, instead of using full RSA encryption, and is used by default. The session key itself is encrypted using the RSA key container used by IISWASOnlyAesProvider, so it has the same access requirements. Configuration attributes encrypted using these providers can only be decrypted by SYSTEM and members of the Administrators group.
The AesProvider provider is an AES provider that uses a session key encrypted using an RSA key container that has permissions for the IIS_IUSRS group, therefore allowing IIS worker processes to encrypt and decrypt configuration encrypted with this provider. This is the provider used by default by all IIS configuration sections that are read by IIS worker processes.

It can also be considered for using to protect custom configuration sections. Configuration attributes encrypted using this provider can be decrypted by any IIS application.

*Note: The IIS configuration system does not support pluggable encryption providers unlike the .NET configuration system. However, new instances of the IIS configuration provider types can be configured to use different key containers for encryption purposes.*

Additional instances of the RSA and AES providers can also be created by creating new entries in the configProtectedData configuration section, and configure them to use new RSA key containers. Create new RSA key containers by using the Aspnet_regiis.exe –pc command.

Then manipulate the permissions on the RSA key to determine who can use it to encrypt and decrypt configuration by using the Aspnet_regiis.exe –pa and Aspnet_regiis –pr commands.

Keep in mind the following guidelines when using encryption:

- Configuration encrypted with IISWASOnlyAesProvider can only be decrypted by members of the Administrators group. This provider is only used to encrypt configuration read exclusively by WAS.
- Configuration encrypted using AesProvider can be decrypted by any IIS application. It can be used to protect configuration from being disclosed outside of the Web server, but it is not a way to protect configuration from applications running on the Web server. It also does not protect configuration used by one application pool from another application pool (although this protection may be afforded by proper NTFS permissions configured for application pool isolation).
- If it is required to encrypt configuration for each application pool as an additional isolation measure, create separate RSA keys for each application pool identity and ACL them for that application pool using the Application Pool SIDs or custom application pool identities. Then create a provider for each application pool, using the corresponding RSA keys, and encrypt the configuration for each application pool using the corresponding provider.
- In order to share a configuration on a Web farm or deploy an application with an encrypted configuration to another server, share encryption keys and provider definitions between the original server on which encryption was performed and the target server. When exporting encryption key containers, be sure to use a strong password and protect these keys from being accessed by unauthorized users.

*Note: Changing the permissions on the RSA key containers may lead to compromise of the encryption keys and therefore may expose encrypted configuration. Do not change the default permissions on the built-in IIS RSA key containers.*

---

12 For more information, see [http://msdn2.microsoft.com/en-us/library/2w117ede.aspx](http://msdn2.microsoft.com/en-us/library/2w117ede.aspx)
8.2.2.3 Limitations of Storing Secrets in Configuration

When secrets are stored in configuration, the secret is protected by both the NTFS permissions on the configuration file and configuration encryption, if configured. However, be aware of the following limitations that may impact the security of the secret:

- NTFS permissions provide the basic level of protection for secrets in configuration files. However, when configuration files are archived, copied off the machine, or sent over a network, this protection is lost. Always use encryption to provide protection for secrets in these cases.
- Any code in the IIS worker process can decrypt any encrypted configuration data that the IIS worker process has access to. By default, any IIS worker processes can decrypt any configuration data encrypted using the default IIS encryption provider (AesProvider).
- Encryption is only as secure as the key that is used to perform the encryption. Therefore, be sure that only the users authorized to perform decryption have access to the key container used to perform the encryption and make sure that this key container is not compromised if it is exported off the machine.

8.2.2.4 Limiting Access to Configuration from Managed Code in Partial Trust Environments

When accessing IIS configuration from native code, the permissions set on the configuration files are the basis for determining whether or not access to the configuration is granted. Native modules and other code running in the IIS worker process can therefore read any of the configuration in the configuration file hierarchy that is not hidden by application pool isolation or encrypted with encryption keys that the IIS worker process does not have access to.

However, when managed code modules access configuration using the Microsoft.Web.Administration API, their ability to read some configuration sections can be further constrained through Code Access Security (CAS) policy configured for the application. This is similar to how CAS is used to prevent managed code applications from performing other actions that the hosting process may otherwise be allowed to perform, such as accessing files or opening network connections.

This mechanism can be leveraged to prevent ASP.NET applications running in partial trust from being able to access information from certain configuration sections. This is done by setting the requirePermission attribute on the section declaration to true. When this is done, only ASP.NET applications and managed modules running with Full Trust can read the contents of these configuration sections.

Note: The 'requirePermission' attribute only prevents the application from using the configuration APIs to read the configuration section when in partial trust. The application can still access the file directly, if the CAS policy and file permissions allow it. Because of this, 'requirePermission' is only effective at preventing medium or below trust applications from reading the contents of configuration sections specified outside of the application’s directory structure, such as in applicationHost.config. The application can still open the distributed web.config files in its directory structure by using file IO APIs directly.

By default, no IIS configuration sections are declared with requirePermission set to true, so the contents of IIS configuration sections can be read by partial trust applications. So, this technique is more applicable to new configuration sections being declared.
8.2.3 Controlling Configuration Delegation

One of the key management scenarios that the IIS 7.0 configuration system has in mind is configuration delegation. Configuration delegation refers to the ability of the Web site or application owner to specify the required IIS configuration for their application without being an administrator on the Web server computer. To allow this, the IIS 7.0 configuration file hierarchy supports specifying configuration in distributed web.config files, which can be located anywhere in the Web site’s directory structure to override the configuration specified at the server level. This also allows Web sites and applications to become portable by including all of the configuration files necessary alongside their content, so they can be deployed by simply being copied to the Web server.

If configuration delegation was an all or nothing approach, it likely wouldn’t work, because most Web server administrators would not want to allow the Web site or application to be able to override all of the configuration set at the server level, especially for configuration sections that affect security, reliability, and performance of the Web server. Therefore, the IIS 7.0 configuration system provides an extensive set of controls that server administrators can use to determine which configuration sections, and further yet, which specific configuration attributes, can be overridden at the Web site or application level. If managing a Web server that allows others to publish application content, it is a likely need to review the configuration allowed for delegation, and in some cases lock or unlock specific configuration for delegation.

Further, IIS 7.0 also provides the infrastructure for Web site and application administrators to manage their configuration remotely through IIS Manager, without having administrative privileges on the Web server computer. Again, server administrators have fine-grained control over who can manage the Web sites and applications on server remotely, and what management features they can use.

8.2.3.1 Controlling Which Configuration Is Delegated

The configuration section is the basic unit of configuration delegation. By default, each configuration section is marked to initially allow or deny delegation when it is first declared, by specifying the overrideModeDefault attribute in the section declaration (this is typically determined by the developer based on whether the section is considered sensitive and should not be modifiable by non-Administrators by default). If the section is marked as not delegated, any attempt to specify the configuration for this section at any lower level in the configuration hierarchy will lead to a configuration error when this section is accessed.

By default, all IIS configuration sections are declared in applicationHost.config. Each section declaration specifies whether or not this section is available for delegation, based on the Microsoft IIS team’s criteria for whether or not the configuration section is sensitive. This criteria includes considerations of whether the configuration section can be used to weaken the security, reduce reliability, or significantly impact the performance of the Web server overall, or allow the Web site or application to access information outside of its boundaries.

Note: The delegation of .NET configuration sections can also be managed by using the IIS administration stack. Both IIS and .NET configuration use the same mechanism for controlling delegation, including section-level locking and fine-grained configuration locking.
The default delegation of IIS configuration sections is shown in Table 8.4

<table>
<thead>
<tr>
<th>Section</th>
<th>Default State</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>system.applicationHost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>applicationPools</td>
<td>n/a</td>
<td>Section can be specified only in applicationHost.config</td>
</tr>
<tr>
<td>configHistory</td>
<td>n/a</td>
<td>Section can be specified only in applicationHost.config</td>
</tr>
<tr>
<td>customMetadata</td>
<td>n/a</td>
<td>Section can be specified only in applicationHost.config</td>
</tr>
<tr>
<td>listenerAdapters</td>
<td>n/a</td>
<td>Section can be specified only in applicationHost.config</td>
</tr>
<tr>
<td>log</td>
<td>n/a</td>
<td>Section can be specified only in applicationHost.config</td>
</tr>
<tr>
<td>sites</td>
<td>n/a</td>
<td>Section can be specified only in applicationHost.config</td>
</tr>
<tr>
<td>webLimits</td>
<td>n/a</td>
<td>Section can be specified only in applicationHost.config</td>
</tr>
<tr>
<td>system.webServer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>asp</td>
<td>Deny</td>
<td>Contains security, performance, and reliability sensitive settings for ASP applications</td>
</tr>
<tr>
<td>caching</td>
<td>Allow</td>
<td></td>
</tr>
<tr>
<td>cgi</td>
<td>Deny</td>
<td>Security sensitive: createProcessAsUser</td>
</tr>
<tr>
<td>defaultDocument</td>
<td>Allow</td>
<td></td>
</tr>
<tr>
<td>directoryBrowse</td>
<td>Allow</td>
<td></td>
</tr>
<tr>
<td>fastCgi</td>
<td>n/a</td>
<td>Section can be specified only in applicationHost.config</td>
</tr>
<tr>
<td>globalModules</td>
<td>n/a</td>
<td>Section can be specified only in applicationHost.config</td>
</tr>
<tr>
<td>handlers</td>
<td>Deny; Allow when .NET Extensibility is installed</td>
<td>For compatibility with IIS 6.0; section is effectively unlocked as soon as .NET Extensibility is installed;</td>
</tr>
<tr>
<td>httpCompression</td>
<td>n/a</td>
<td>Section can only be specified in applicationHost.config</td>
</tr>
<tr>
<td>httpErrors</td>
<td>Deny</td>
<td>Security sensitive: ability to specify error pages outside of the application</td>
</tr>
<tr>
<td>httpLogging</td>
<td>Deny</td>
<td>Security sensitive: turning off logging can create repudiation issues</td>
</tr>
<tr>
<td>httpProtocol</td>
<td>Allow</td>
<td></td>
</tr>
<tr>
<td>httpRedirect</td>
<td>Allow</td>
<td></td>
</tr>
<tr>
<td>httpTracing</td>
<td>Deny</td>
<td>Performance sensitive: list of ETW URLs to trace</td>
</tr>
<tr>
<td>isapiFilters</td>
<td>n/a</td>
<td>Section can be specified only in applicationHost.config</td>
</tr>
<tr>
<td>modules</td>
<td>Deny; Allow when .NET Extensibility is installed</td>
<td>For compatibility with IIS 6.0; section is effectively unlocked as soon as .NET Extensibility is installed;</td>
</tr>
<tr>
<td>odbcLogging</td>
<td>Deny</td>
<td>Security sensitive: configuring logging to external database</td>
</tr>
<tr>
<td>serverRuntime</td>
<td>Deny</td>
<td>Security, performance, and reliability affecting settings for the core Web server engine</td>
</tr>
<tr>
<td>serverSideInclude</td>
<td>Deny</td>
<td>Security sensitive: enabling server-side include can allow the application to access content outside of its boundaries</td>
</tr>
<tr>
<td>staticContent</td>
<td>Allow</td>
<td></td>
</tr>
<tr>
<td>urlCompression</td>
<td>Allow</td>
<td></td>
</tr>
<tr>
<td>validation</td>
<td>Allow</td>
<td></td>
</tr>
<tr>
<td>Section</td>
<td>Default State</td>
<td>Reason</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>---------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>system.webServer/security</td>
<td></td>
<td></td>
</tr>
<tr>
<td>access</td>
<td>Deny</td>
<td>Security sensitive: configure SSL requirements</td>
</tr>
<tr>
<td>applicationDependencies</td>
<td>n/a</td>
<td>Section can be specified only in applicationHost.config</td>
</tr>
<tr>
<td>authorization</td>
<td>Allow</td>
<td></td>
</tr>
<tr>
<td>ipSecurity</td>
<td>Deny</td>
<td>Security sensitive: determine who can access the application</td>
</tr>
<tr>
<td>isapiCgiRestriction</td>
<td>n/a</td>
<td>Section can only be specified in applicationHost.config</td>
</tr>
<tr>
<td>requestFiltering</td>
<td>Allow</td>
<td>Caution: delegated, but application can end up removing basic protection configured at server level and lessen its security</td>
</tr>
<tr>
<td>system.webServer/security/authentication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>anonymousAuthentication</td>
<td>Deny</td>
<td>Security sensitive: enable or disable authentication method</td>
</tr>
<tr>
<td>basicAuthentication</td>
<td>Deny</td>
<td>Security sensitive: enable or disable authentication method</td>
</tr>
<tr>
<td>clientCertificateMapping</td>
<td>Deny</td>
<td>Security sensitive: enable or disable authentication method</td>
</tr>
<tr>
<td>digestAuthentication</td>
<td>Deny</td>
<td>Security sensitive: enable or disable authentication method</td>
</tr>
<tr>
<td>iisClientCertificateMapping</td>
<td>Deny</td>
<td>Security sensitive: enable or disable authentication method</td>
</tr>
<tr>
<td>windowsAuthentication</td>
<td>Deny</td>
<td>Security sensitive: enable or disable authentication method</td>
</tr>
<tr>
<td>system.webServer/tracing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>traceFailedRequests</td>
<td>Allow</td>
<td></td>
</tr>
<tr>
<td>traceProviderDefinitions</td>
<td>n/a</td>
<td>Section can only be specified in applicationHost.config</td>
</tr>
</tbody>
</table>

Table 8.4: Default Delegation of IIS Configuration Sections

The default delegation state for IIS configuration sections is just that—a default—and may not work for everyone. If third parties are allowed to publish Web site or application configuration on the server, the impacts of allowing need to be reviewed for each section to be delegated and strike a balance between application requirements for delegation and the need to protect the Web server from unintended or malicious configuration changes. Then, lock or unlock configuration sections to allow them for delegation or even use fine-grained configuration locking to allow section delegation but lock specific configuration attributes, elements, or collection entries.

When determining which configuration should be delegated, keep the following guidelines in mind:

- Err on the side of leaving configuration sections locked at the server level and unlock specific sections as needed by the application. Unlock specific sections for specific Web sites or applications only and leave them locked for others. This is an effective method to avoid unexpected configuration changes at the application level even if configuration to other parties is not delegated.
- When unlocking a specific section, parts of it still can be locked that contain sensitive configuration or configuration that should not be changed. Use fine-grained configuration locking to lock the attributes, elements, or collection elements that should not be changed while allowing other parts of the configuration section to be delegated.
9. Implementing a Secure Network Infrastructure

The network infrastructure that supports the Web server plays a critical role in the security of the Web server. In most configurations, the network infrastructure is the first line of defense between the Internet and a public Web server. Network design alone, however, cannot protect a Web server; a through layered and diverse protection mechanisms (defense-in-depth) should be implemented for protection. Although security issues are paramount, network infrastructure considerations are influenced by many factors other than security, including cost, performance, and reliability.

Following important factors need to be considered while implementing a secure network infrastructure:

- **DMZ**: It is advisable that the Web Server should be placed in DMZ. A demilitarized zone (DMZ) describes a host or network segment inserted as a “neutral zone” between an organization’s private network and the Internet. It prevents outside users of the Web server from gaining direct access to an organization's internal network (intranet). A DMZ mitigates the risks of locating a Web server on an internal network or exposing it directly to the Internet. It allows access to the resources located within it to both internal and external users. There are a wide variety of DMZ configurations, like single-firewall DMZ, two-firewall DMZ and “service leg” DMZ, each with its own strengths and weaknesses; organizations have to choose either of them as per their requirements.

- **Outsourcing**: Some organizations choose to outsource the hosting of their Web servers to a third party (e.g., an ISP, Web hosting service, or other government agency). In this case, the Web server would not be located on the organization’s network. The hosting service network would have a dedicated network that hosts many Web servers (for many organizations) operating on a single network. From a security standpoint, the advantages of outsourcing are:
  - DoS attacks aimed at the Web server have no effect on the organization’s production network.
  - Compromise of the Web server does not directly threaten the internal production network.
  - The outsourcer may have greater knowledge of securing and protecting Web servers.
  - The network can be optimized solely for the support and protection of Web servers.

The disadvantages of outsourcing from a security standpoint are:
  - It requires trusting a third party with Web server content.
  - It is more difficult to remotely administer the Web server or remotely update Web server content.
  - Less control can be provided over the security of the Web server.
  - The Web server may be affected by attacks aimed at other Web servers hosted by the outsourcer on the same network.

Outsourcing often makes sense for smaller organizations that cannot afford to support the necessary Web server staff. It may also be appropriate for larger organizations that do not wish to host their own Web servers. Outsourcing usually does not make sense for organizations that wish to maintain tight control over their Web servers.

- **Management Network**: Web servers and other important servers should be connected to each other and managed through a separate network known as a management network. An additional network interface known as a management interface is provided in each server/host that connects to the management network. The host being managed is unable to pass any traffic between its management interface and any of its other network interfaces. Consoles and other hosts that are used to manage the Web components should be attached to the management network only. This architecture effectively isolates the management network from the production networks. The benefits of doing this are to protect the components from some attacks and to ensure that the components can be managed under adverse
conditions (e.g., DDoS attack). Disadvantages of using a management network include the additional costs of networking equipment and other hardware.

- **Firewalls**: Organizations have to choose a firewall [according to their requirements and network architecture] from a range that is present in the market. It is advisable that an Application layer firewall should be deployed to monitor traffic passing to the Web Server. Application layer firewalls (sometimes called application-proxy gateway firewalls) are advanced firewalls that combine network and transport layer access control with application layer functionality. Application layer firewalls permit no traffic directly between the Internet and the internal network. They can also log traffic, perform authentication, convert protocols and perform other useful actions. They are more complex than other types of firewalls, since they require configuration for each application-layer protocol for which traffic will be allowed to pass. They are very resource intensive and usually require special software or configuration by the user. Although not strictly a limitation, some application layer firewalls are implemented on hosts running general-purpose OSs (e.g., Windows, Linux, and UNIX). This arrangement introduces an added layer of complexity and some additional risk because the general-purpose OS must also be secured in addition to the firewall software itself. Application layer firewalls are increasingly being deployed as appliance-based devices, which may use specialized OSs. To successfully protect a Web server using a firewall, ensure that the firewall is patched to the latest or most secure level (both the application and the underlying OS) and is configured to perform the following:
  - Control all traffic between the Internet and the Web server
  - Block all inbound traffic to the Web server except traffic which is required, such as TCP ports 80 (HTTP) and/or 443 (HTTPS)
  - Block all inbound traffic with an internal IP address (to prevent IP spoofing attacks)
  - Block client connections from the Web server to the Internet and the organization’s internal network (this will reduce the impact of some successful compromises)
  - Block (in conjunction with the intrusion detection or prevention system IP addresses or subnets that the IDS or IPS reports are attacking the organizational network
  - Notify the network or Web server administrator or appropriate security personnel of suspicious activity through an appropriate means (e.g., page, e-mail, network trap)
  - Provide content filtering
  - Protect against DoS attacks
  - Detect malformed or known attack URL requests
  - Log critical events, including the following details:
    - Time/date
    - Interface IP address
    - Manufacturer-specific event name
    - Standard attack event (if one exists)
    - Source and destination IP addresses
    - Source and destination port numbers
    - Network protocol.

- **Intrusion Detection and Prevention Systems (IDPS)**: An IDS is an application that monitors the events occurring in a system or network and analyzes them for signs of potential incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices. An IPS has all the capabilities of an IDS and can also attempt to stop potential incidents. Because IDS and IPS systems offer many of the same capabilities, they are often collectively called intrusion detection and prevention systems (IDPS). When an IDPS detects a potential incident, it notifies administrators through IDPS console messages, emails, pages, or other mechanisms. The two types of IDPSs most relevant for Web security are host-based and network-based. Each of them has its own strengths and weakness; organizations have to choose either of them as per their requirements or
To successfully protect a Web server using an IDPS, following configurations need to consider:

- Monitor network traffic to and from the Web server
- Monitor changes to critical files on the Web server (file integrity checking capability)
- Monitor the system resources available on the Web server host (host-based)
- Block (in conjunction with the firewall) IP addresses or subnets that are attacking the organizational network
- Notify the appropriate parties (e.g., IDPS administrator, Web server administrator, incident response team) of suspected attacks through appropriate means according to the organizational incident response policy and procedures
- Detect as wide a variety of scanning and attacks as possible with an acceptable level of false positives
- Log events, including the following details:
  - Time/date
  - Sensor IP address
  - Manufacturer-specific attack name
  - Standard attack name (if one exists)
  - Source and destination IP addresses
  - Source and destination port numbers
  - Network protocol
- For network events, capture packet header information to assist with the analysis and forensics process
- Update with new attack signatures frequently (e.g., on a daily to weekly basis, typically after testing the updates).
- Update the underlying OS if IDPS is running on general purpose OS
- If remote management is required, it should be conducted through out-of-band means (i.e. separated isolated network).

- **Load Balancers:** Load balancers distribute HTTP requests over multiple Web servers, allowing organizations to increase the capacity of their Web site by transparently adding additional servers. Load balancers act as virtual servers, receiving all HTTP requests to the Web site. These requests are forwarded, based on the load balancer’s policy, to one of the servers that hosts the Web site. The load balancer’s policy attempts to ensure that each server receives a similar number of requests. Many load balancers are capable of monitoring the servers and compensating if one of the servers becomes unavailable.

  Load balancers are not specifically security appliances, but they are essential tools for maintaining the availability of a Web site. By ensuring that several individual Web servers are sharing the load, rather than placing it on a single Web server, the organization is better able to withstand the high volume of requests used in many DoS attacks. Firewalls, switches, and routers should also be configured (when possible) to limit the amount of traffic that is passed to the Web servers, which further reduces the risk of successful DoS attacks.

- **Caching servers:** Many of the HTTP requests an organization’s Web server receives are identical and return identical HTTP responses. However, when dynamic content generation is in use, these identical responses need to be regenerated each time the request is made. To alleviate this requirement and further reduce the load on individual Web servers, organizations can deploy caching servers.

- **Reverse proxies:** These are devices that sit between a Web server and the server’s clients. The term “reverse proxy” is used because the data flow is the reverse of a traditional (forward) proxy. Reverse proxies can serve as a valuable addition to the security of a Web server and should be considered for any high-risk Web server deployment. While they do add risk by requiring the deployment of additional hardware and software, the risk is generally outweighed by the benefits. Reverse proxies
are also valuable because they add an additional layer between a Web server and its less trusted users. Due to their highly specialized nature, proxies are easier to secure than Web servers. Proxies also further obfuscate a Web server’s configuration, type, location, and other details that are pertinent to attackers. For example, Web servers have banners that frequently reveal the Web server type and version, and these banners sometimes cannot be changed. With a reverse proxy, this is not an issue because the proxy can rewrite the banner before it is sent to users.

Note: For comprehensive checklist, see “Appendix-A: Web Server Security Checklist”

9.1 Threat Modeling

Threat modeling is based on the notion that any system or organization has assets of value worth protecting, these assets have certain vulnerabilities, internal or external threats exploit these vulnerabilities in order to cause damage to the assets, and appropriate security countermeasures exist that mitigate the threats.

Threat Modeling is done at two levels namely Application and IT Infrastructure.

9.1.1 Application

Threat modeling allows to apply a structured approach to security and to address the top threats that have the greatest potential impact to application. It enables development teams to better understand the threat that a component will have to face after release. By identifying and rating threats based on understanding of the architecture and implementation of application, development team can address threats with appropriate countermeasures in a logical order. This way the defensive mechanisms are built into the code as it is written rather than patched in later. This is much more cost effective and has the added benefit of increasing security awareness in the development team.

9.1.2 IT Infrastructure

Threat modeling allows to determine what threats exist that could affect the organization's IT infrastructure, helps to identify threat mitigations to protect resources and sensitive information, and helps to prioritize the identified threats so that security efforts can be managed in a proactive manner.

Threat modeling of IT infrastructure should immediately follow a well-conducted risk assessment that is fully supported by management.

It should be performed throughout the technology implementation process, much like any other component that is measured for performance, usability, and availability. The result might reveal weaknesses in architecture or implementation and design planning that could require significant changes to the project. Design changes early in the implementation process are significantly less expensive than a complete reimplementation after a failed attempt that wasn’t well planned, or if an insufficiently secured system achieves production status.

Consider engaging in an IT infrastructure threat model portfolio review process annually, even if no major changes are made to the organization's IT infrastructure. Because threats and risks are dynamic, threat modeling efforts should be as well. Attackers won’t stop their attempts to exploit vulnerabilities, and efforts to mitigate potential threats will help make compromise less likely. The IT infrastructure threat modeling process might also improve an organization's audit profile by showing that the process is consistently practiced. In addition, mitigations achieved
during the threat modeling process might prevent the need to undertake the same effort in the high-pressure spotlight of an audit and its potential mandates.

Note: Refer ‘IT Infrastructure Threat Modeling Guide’, from Microsoft [Ref: 7] for more information. Threat modeling has become an integral part of Microsoft’s SDL (Security Development Lifecycle) process.
10. IIS 7.0 Logging

In IIS 7.0, log files are handled by the HTTP.sys kernel mode device driver. No user code runs in this service because HTTP.sys runs in kernel mode.

Logging greatly benefits from the flexibility provided by the modularity in IIS 7.0 and the Integrated Pipeline because own logging module can be written and inject it into the pipeline. The custom module can capture just the information needed for the application.

IIS 7.0 incorporates several changes pertinent to logging:

- Use the IIS Manager to configure logging-related settings.
- The new configuration system is based on XML.
- There are a number of new logging configuration options and a new set of status codes.
- IIS 7.0 provides logging for a new service that enables remote administration of an IIS 7.0 server.

10.1 IIS Manager

IIS 7.0 introduces a completely new user interface, the IIS Manager. The IIS Manager makes it easier to browse and make changes to IIS settings, including log file settings. For example, the Centralized Logging option is exposed in the IIS Manager, as shown in Figure 10.1
10.2 The XML-Based Logging Schema

IIS 7.0 uses a new configuration system that is XML-based and is very similar to the ASP.NET configuration system. Each configuration section is defined in XML schema files located in %SystemRoot%\system32\inetsrv\config\schema.

Because information is defined in XML files, it is easy to determine what attributes, elements, and enums are used. The schema for IIS 7.0 contains a list of all the configurable options, so looking in the schema file is a quick way to identify all the configurable settings for any feature, including logging.

The following listing is from the system.applicationHost/log section that is located in %SystemRoot%\system32\inetsrv\config\schema\IIS_Schema.xml. (Some long lines have been split to fit on the printed page.)
<sectionSchema name="system.applicationHost/log">
  <attribute name="logInUTF8" type="bool" defaultValue="true"/>
  <attribute name="centralLogFileMode" type="enum" defaultValue="Site">
    <enum name="Site" value="0"/>
    <enum name="CentralBinary" value="1"/>
    <enum name="CentralW3C" value="2"/>
  </attribute>
  <element name="centralBinaryLogFile">
    <attribute name="enabled" type="bool" defaultValue="false"/>
    <attribute name="directory" type="string" expanded="true" defaultValue="%SystemDrive%\inetpub\logs\LogFiles"/>
    <attribute name="period" type="enum" defaultValue="Daily">
      <enum name="Hourly" value="4"/>
      <enum name="Daily" value="1"/>
      <enum name="Weekly" value="2"/>
      <enum name="Monthly" value="3"/>
      <enum name="MaxSize" value="0"/>
    </attribute>
    <attribute name="truncateSize" type="int64" defaultValue="20971520" validationType="integerRange" validationParameter="1048576,4294967295"/>
    <attribute name="localTimeRollover" type="bool" defaultValue="false"/>
  </element>
  <element name="centralW3CLogFile">
    <attribute name="enabled" type="bool" defaultValue="true"/>
    <attribute name="directory" type="string" expanded="true" defaultValue="%SystemDrive%\inetpub\logs\LogFiles" validationType="nonEmptyString"/>
    <attribute name="period" type="enum" defaultValue="Daily">
      <enum name="Hourly" value="4"/>
      <enum name="Daily" value="1"/>
      <enum name="Weekly" value="2"/>
      <enum name="Monthly" value="3"/>
      <enum name="MaxSize" value="0"/>
    </attribute>
    <attribute name="truncateSize" type="int64" defaultValue="20971520" validationType="integerRange" validationParameter="1048576,4294967295"/>
    <attribute name="localTimeRollover" type="bool" defaultValue="false"/>
    <attribute name="logExtFileFlags" type="flags" defaultValue="Date, Time, ClientIP, UserName, SiteName, ServerIP, Method, UriStem, UriQuery, HttpStatus, Win32Status, ServerPort, UserAgent, HttpSubStatus">
      <flag name="Date" value="1"/>
      <flag name="Time" value="2"/>
      <flag name="ClientIP" value="4"/>
      <flag name="UserName" value="8"/>
      <flag name="SiteName" value="16"/>
      <flag name="ComputerName" value="32"/>
      <flag name="ServerIP" value="64"/>
      <flag name="Method" value="128"/>
    </attribute>
  </element>
</sectionSchema>
10.3 Centralized Logging Configuration Options

Following is the logging section defined in the ApplicationHost.config file that controls Centralized Logging options. Web Administrator can change this so that files are stored on another drive or volume, enable options needed and disable whatever options not need.

```xml
<log>
  <centralBinaryLogFile enabled="true"
    directory="%SystemDrive%\inetpub\logs\LogFiles"/>
  <centralW3CLogFile enabled="true"
    directory="%SystemDrive%\inetpub\logs\LogFiles"/>
</log>
```

10.4 SiteDefaults Configuration Options

The SiteDefaults section in the ApplicationHost.config file, shown in the following code, controls the logging settings that are used when creating new sites. It can be configured for two options: the format of the log file and the location in which Failed Request tracing files are stored.

```xml
<siteDefaults>
  <logFile logFormat="W3C"
    directory="%SystemDrive%\inetpub\logs\LogFiles"/>
  <traceFailedRequestsLogging
directory="%SystemDrive%\inetpub\logs\FailedReqLogFiles"/>
</siteDefaults>
```

10.5 Disable HTTP Logging Configuration Options

In some cases, a Web administrator may not require log files. HTTP logging can be disabled at the site server level as well as the site level.
The default value for this setting, as shown here, is false.

<httpLogging dontLog="false" />

*Note: It is advised here that Web Administrator should evaluate the options before disabling httpLogging. Check with business or legal department to be certain about company’s logging requirements and policies.*

### 10.6 Default Log File Location

One of the most significant changes in IIS 7.0 is that the folder where IIS stores WWW logs has been changed to %SystemDrive%\inetpub\logs\LogFiles. For example, the Default Web Site would log to C:\inetpub\logs\LogFiles\w3svc1. This means that by default in IIS 7.0, all log files are stored in a single folder. Note, however, that log files for the legacy built-in File Transfer Protocol (FTP) and Simple Mail Transfer Protocol (SMTP) services are still located in %windir%\System32\Logfiles. These files can be managed by using the IIS Manager 6.0, an MMC console that is installed when the legacy FTP service or the SMTP service is being installed.

*Note: The new FTP Publishing Service for IIS 7.0 is an add-on and does not ship with it. It stores its log files in %SystemDrive%\inetpub\logs\LogFiles by default.*

### 10.7 Default UTF-8 Encoding

By default, IIS 7.0 stores log files by using UTF-8 encoding. This changes the default file naming convention so that the files start with u_ (for example, u_exYYMMDD.log). Here is the portion of the IIS_Schema.xml file that sets the UTF-8 encoding option. The default setting is true.

```xml
<sectionSchema name="system.applicationHost/log">
  <attribute name="logInUTF8" type="bool" defaultValue="true" />
</sectionSchema>
```

UTF-8 encoding allows for single-byte and multi-byte characters in one string. This encoding enables to read text-based logs (for example, logs that use W3C Extended, IIS, and NCSA Common formats) in a language other than English. IIS does not support the UTF-8 format for the built-in FTP Publishing Service log files. UTF-8 encoding is available in IIS 6.0, but it is not enabled by default.

### 10.8 New Status Codes

In IIS 7.0, new status codes have been introduced for HTTP and FTP. These additional error codes provide more details about events and better descriptions of how to fix errors, with suggestions about what to look for or what procedures to run.

### 10.9 Management Service

IIS 7.0 introduces Management Service, which enables computer and domain administrators to remotely manage a machine by using the IIS Manager. The Management Service also enables non-administrators to control sites and various applications by using the IIS Manager from a workstation.
This service has its own logs that are used to track information related to the Management Service. This service is not installed by default. If installed and enabled this service, the logs will be saved in %SystemDrive%\inetpub\logs\WMSvc.

From a logging perspective, web administrator should make sure the Management Service logging is enabled. The logs can help to audit and troubleshoot issues when clients are connecting to the server.

10.10 Log File Formats That Have Not Changed

IIS 7.0 supports all the common logging formats that are available in prior versions of IIS. There have been no changes in IIS 7.0 to the following log file formats:

- Microsoft IIS
- NCSA
- W3Svc extended

10.11 Centralized Logging

Centralized logging in IIS 7.0 operates the same way as it does in IIS 6.0. However, this option can now be configured in the IIS 7.0 Manager. To access this feature, go to Administrative Tools > Internet Information Services (IIS) Manager. Click the computer name and locate the Logging option listed in the IIS section.

Using the Logging option can reduce administrative costs because only one IIS log file is being maintained. If binary logging is used, the log can be stored in a much smaller file than the equivalent text log file.

10.11.1 W3C Centralized Logging Format

W3C centralized logging is a server-level setting. When this feature is enabled on a server, all Web sites on that server are configured to write log data to a central log file. Data is stored in the log file using the W3C Extended log file format. This setting can be enabled through the IIS 7.0 Manager or by using Appcmd. If W3C centralized logging is being used, the log file can be viewed with a text editor such as Notepad.

Note: W3C centralized logging uses the W3C Extended log format, which includes the following four fields: HostHeader, Cookie, UserAgent, and Referrer. These fields are not available in centralized binary logging. The built-in FTP and SMTP services do not support W3C centralized logging.

10.11.2 Centralized Binary Logging Format

Centralized binary logging is essentially the same as W3C centralized logging, except that the log file uses a proprietary, binary format. Because the resulting file is binary, it is smaller than an equivalent text file so that disk space can be conserved. It cannot be read with a text editor and requires parsing to produce useful information. This type of logs can be read by Log Parser tool

Log parser is a powerful, versatile tool that provides universal query access to text-based data such as log files, XML files and CSV files, as well as key data sources on the Windows® operating system such as the Event Log, the Registry, the file system, and Active Directory.


Log parser is a powerful, versatile tool that provides universal query access to text-based data such as log files, XML files and CSV files, as well as key data sources on the Windows® operating system such as the Event Log, the Registry, the file system, and Active Directory.

10.12 Remote Logging

IIS 7.0 supports writing log files to a network share. This option enables to have the log files stored in real time to a remote computer. For example, in a Web farm configured for logging to a central location. The remote file server could be a server running DFS (distributed file system). DFS can provide multiple benefits including a central location to collect log files and automatic replication of logs to multiple locations. Having such a primary collection point can make handling the reporting processes much easier. Use either the IIS 7.0 Manager or Appcmd to set up Universal Naming Convention (UNC) remote logging.

10.12.1 Setting Up Remote Logging by Using the IIS Manager

This example assumes that the NETWORK SERVICE is being used as application pool account and that the remote server and Web server are in the same domain. Following are the steps to enable remote logging by using the IIS Manager:

1. Create a directory called IISLogs on the remote server that will store the log files. This machine is typically in the same domain as the Web servers.
2. Share the IISLogs folder created in the previous step. Change the share permissions to—at minimum—enable both the remote machine accounts Administrators group and the account that is writing the log files full control. Change the NTFS file system (NTFS) permissions so that the remote machine accounts Administrators have full control and the account writing the log files has modify permissions.
3. In the IIS Manager, navigate to Web site and type in the UNC path to the server. To do so, go to Administrative Tools > Internet Information (IIS) Manager. Select the computer name in the leftmost column and then double-click the Logging icon in the IIS Section. Type the path to the share in the Directory text box by using the syntax \ServerName\ShareName, as shown in Figure 10.2
4. Click Apply.
5. Browse a Web page in site.
6. Open a command prompt by using elevated credentials and type netsh http flush logbuffer. If this is the first time entries have been logged, HTTP.sys will create the folder and a log file. Open the log file in Notepad to confirm that example entries have been logged.
10.12.2 Setting Up Remote Logging by Using Appcmd

Appcmd can also be used to update the logfile directory for a specific Web site. The syntax for configuring UNC remote logging using Appcmd is shown here. (The line has been split to fit it on the printed page.)

//Appcmd to set the log directory path for Default Web Site
Appcmd set sites "Default Web Site"
-logFile.directory:\\RemoteServerCMD.Contoso.com\LogFiles

10.12.3 Remote Logging Using the FTP 7.0 Publishing Service

The FTP 7.0 Publishing Service is an out-of-band add-on that is meant to replace the built-in FTP service. The FTP 7.0 Publishing Service supports logs stored on a remote computer, which can enhance the ability to track down security breaches. Imagine a particular machine is compromised, but logs stored on a remote system. When the infiltrator tries to cover his tracks by deleting the local log files, those log files will be unavailable because they are stored on a remote share. If remote share uses DFS, the log files can even be replicated to multiple locations. Remote logging with replication can help in forensic efforts. To configure the FTP logs to be stored on a remote server, configure the remote server that houses logs files the same as configure a Web server.
10.12.4 Custom Logging

The modular architecture of IIS 7.0 enables to implement user’s owned logging modules or extend or replace existing logging options. User owned module can be implemented directly into the request pipeline. It can be either a native module or a module written using managed code.

10.13 Configuring IIS Logging

IIS 7.0 provides multiple ways to configure and administer Web server, and that includes configuring log settings. This section covers how to use the IIS Manager, Appcmd, and Windows PowerShell to configure log settings.

10.13.1 IIS Manager

To access the Logging section of the IIS Manager, follow this procedure:

1. Go to Administrative Tools > Internet Information Services (IIS) and select the server name. Figure 10.3 shows the icon for the global Logging section when it is selected.

![IIS Manager](image)

Figure 10.3: The icon for the global Logging section selected in the IIS Manager

2. Double-click the Logging icon to view the interface through which logging settings for the server can be administered.
The default settings are shown in Figure 10.4. Because the server node selected is in the tree in the left pane, these settings are inherited by all Web sites configured on the server.

3. To make changes, select the appropriate drop-down box and select the option. For example, to change the server from site-level logging (creating one log file per site) to server-level logging (creating one log file per server), select Server in the One Log File Per drop-down list, as shown in Figure 10.5.
Figure 10.5: Go to the IIS Manager to change logging from site-level logging to server-level logging.

Note: When IIS 7.0 is configured to use server-level logging, the Binary format is selected by default. To have server-level log use W3C extended logging, simply select W3C in the Format drop-down list.

IIS 7.0 also enables to make changes on individual Web sites. For example, click the Select Fields button to adjust which options are logged for a specific Web site, as shown in Figure 10.6. In this figure, the Bytes Sent ( sc-bytes ), Bytes Received ( cs-bytes ), Time Taken ( time-taken ), and Referer ( cs(Referer) ) options have been selected. Other options can also be adjusted the log Directory setting, the Log File Rollover setting, and the Use Local Time For File Naming And Rollover setting.
10.13.2 **Appcmd**

Appcmd is a tool that is intended to perform all administrative functions from a command line. Appcmd replaces a variety of scripts and tools used in previous IIS versions.

All the logging settings that might be needed to adjust are located in three sections of application-Host.config: system.applicationHost/log, system.applicationHost/sites, and system.web-Server/httpLogging.

The previous example uses IIS Manager to configure server-level logging. To use Appcmd to perform this same operation, follow this procedure:

1. Open a command prompt and navigate to the %SystemRoot%\System32\inetsrv folder where Appcmd is deployed.
2. Execute the following command from the command prompt to list the current settings:
   
   Appcmd list config -section:log
   
   ```xml
   <system.applicationHost>
   <log>
   <centralBinaryLogFile enabled="true">
   <centralW3CLogFile enabled="true">
   </log>
   </system.applicationHost>
   ```
This will display the ApplicationHost.config section where the centralLogFileMode settings are stored.

3. Next execute the following command to configure server-level logging:
   ```
   Appcmd set config -section:log -centralLogFileMode:CentralW3C
   ```

4. After the command in step 3 have been executed, execute the following command to list the current settings and verify the settings have been changed:
   ```
   Appcmd list config -section:log
   ```

   The result, showing that the centralLogFileMode has changed to CentralW3C, should look like the following. (Some lines have been split to fit on the printed page.)

   ```
   C:\Windows\System32\inetsrv>Appcmd list config -section:log
   <system.applicationHost>
   
   <log centralLogFileMode="CentralW3C">
   
   <centralBinaryLogFile enabled="true">
   directory="%SystemDrive%\inetpub\Logs\LogFiles" />
   </centralBinaryLogFile>

   <centralW3CLogFile enabled="true">
   directory="%SystemDrive%\inetpub\Logs\LogFiles" />

   </log>

   </system.applicationHost>
   ```

   Notice the log centralLogFileMode="CentralW3C" setting. Before executing the Appcmd set config command, there was no value listed, because the Site option is the default setting as defined in the schema.

Appcmd enables to quickly use the command line to make changes to IIS log settings. A set of scripts can be created that use Appcmd to replace the repetitive changes typically required when using the IIS Manager GUI. Such scripts can help streamline and automate server configuration and deployment.

### 10.14 HTTP.sys Logging

HTTP.sys introduced another log called HTTPERR log. The HTTPERR logs for Windows Server 2008 are located at the path %SystemRoot%\System32\LogFiles\HTTPERR. This log records all errors that are not handed off to a valid worker process, typically responses to clients, connection time-outs, and orphaned requests. This additional information helps to troubleshoot HTTP-based errors, which are logged before the request reaches IIS.

Windows Vista and Windows Server 2008 introduce enhancements to the HTTP.sys logging process. Use ETW (Event Tracing for Windows) to obtain the enhanced information. Here are steps to start, capture, and display information from an ETW tracing session:

1. Open a command prompt (click Start, select Run, and then type cmd.exe).
2. Start the ETW trace session for HTTP.sys by using the following command:
   ```
   logman.exe start httptrace -p Microsoft-Windows-HttpService 0xFFFF -o httptrace.etl --ets
   ```
3. Reproduce or perform the steps or tests that need to be traced.
4. To stop the ETW trace session for HTTP.sys, use the following command:
   logman stop httptrace –ets

5. To convert the ETL file to a comma-separated file (CSV) file, use this command:
   tracerpt httptrace.etl -of csv -o httptrace.csv /y

   The CSV files can then be viewed in a text editor or spreadsheet application.

10.15  Application Logging

Besides the standard IIS type logs, other items can be logged. Many of these options can be set with the IIS Manager or by using Appcmd.

10.15.1  Process Recycling Logging

In IIS 7.0, events are logged to the Windows event log when an application pool recycles. Following eight configuration settings can be controlled with each option listed in Table 10.1

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Default Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Pool Configuration Changed</td>
<td>Event is logged when the application pool recycles due to a change in its configuration</td>
<td>No</td>
</tr>
<tr>
<td>ISAPI Report Unhealthy</td>
<td>Event is logged because an ISAPI extension has reported itself as unhealthy</td>
<td>No</td>
</tr>
<tr>
<td>Manual Recycle</td>
<td>Event is logged when the application pool has been manually recycled</td>
<td>No</td>
</tr>
<tr>
<td>Private Memory Limit Exceeded</td>
<td>Event is logged when the application pool recycles after exceeding its private memory limit</td>
<td>Yes</td>
</tr>
<tr>
<td>Regular Time Interval</td>
<td>Event is logged when the application pool recycles on its scheduled interval</td>
<td>Yes</td>
</tr>
<tr>
<td>Request Limit Exceeded</td>
<td>Event is logged when the application pool recycles after exceeding its request limit</td>
<td>No</td>
</tr>
<tr>
<td>Specific Time</td>
<td>Event is logged when the application pool recycles at a scheduled time</td>
<td>No</td>
</tr>
<tr>
<td>Virtual Memory Limit Exceeded</td>
<td>Event is logged when the application pool recycles after exceeding its virtual memory limits</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 10.1: Recycling Options Under Generate Recycle Event Log Entry

10.15.2  ASP

Classic ASP (Active Server Pages) is alive and well in IIS 7.0, and configure options for logging ASP errors under the ASP section in the IIS Manager. Use the following options to discover issues when migrating Classic ASP applications to IIS 7.0.

- Enable Log Error Requests Controls whether the Web server writes ASP errors to the application event log
- Log Errors To The NT Log Specifies that ASP errors are recorded in the Windows event log
10.15.3 ASP.NET

All ASP.NET 2.0 unhandled exceptions are written to the Application Event log. Along with application pool recycles events or other errors in the event logs, this can be very helpful in troubleshooting application errors. ASP.NET logging can be turned off by following the instructions in the Knowledge Base article at http://support.microsoft.com/kb/911816.

10.15.4 IIS Events

Other processes related to IIS also log to the Windows Event log. This includes the HTTP, IISAdmin, FTP Publishing Service, and W3SVC services.

For a complete list of events, go to http://technet.microsoft.com and search for “IIS events reference.”

10.16 Folder Compression Option

Log files are necessary to keep track of Web site statistics and trends, and Web developers and business people use them to ensure their Web sites continue to grow. One of the biggest challenges administrators face is how to retain and manage log files. By default, IIS rolls over log files once a day. Log files can become quite large even if the default log file rollover setting is being used.

Windows Server 2008 allows for folder compression to help save space. This option can be enabled by using Windows Explorer.

Because HTTP.sys buffers information written to the IIS log files, there will not be a performance hit if log files are in a folder for which compression is enabled. Third-party log compression products or free tools such as Gzip can also be used along with scripts to compress log files.
11. Administering the Web Server

After initial deployment of a Web Server, administrators need to maintain its security continuously. It’s an iterative process and involves activities like analyzing log files, performing regular Web server backups, recovering from server compromises, testing Web server security regularly and performing remote administration.

11.1 Analyzing log files

Logging is a cornerstone of a sound security posture. Capturing the correct data in the logs and then monitoring those logs closely is vital. Network and system logs are important, especially system logs in the case of HTTPS-protected communications, where network monitoring is less effective. Web server software can provide additional log data relevant to Web-specific events. Similarly, Web applications may also maintain their own logs of actions.

Reviewing logs is mundane and reactive, and many Web server administrators devote their time to performing duties that they consider more important or urgent. However, log files are often the only record of suspicious behavior. Enabling the mechanisms to log information allows the logs to be used to detect failed and successful intrusion attempts and to initiate alert mechanisms when further investigation is needed. Procedures and tools need to be in place to process and analyze the log files and to review alert notifications.

Web server logs provide:
- Alerts to suspicious activities that require further investigation
- Tracking of an attacker’s activities
- Assistance in the recovery of the system
- Assistance in post-event investigation
- Required information for legal proceedings.

The selection and implementation of specific Web server software determines which set of detailed instructions the Web server administrator should follow to establish logging configurations.

If a public Web server supports the execution of programs, scripts, or plug-ins, it may be necessary for the programs, scripts, or plug-ins to perform additional logging. Often, critical events take place within the Web application code itself and will not be logged by the Web server. If Webmasters develop or acquire application programs, scripts, or plug-ins, it is strongly recommended that they define and implement a comprehensive and easy-to-understand logging approach based on the logging mechanisms provided by the Web server host OS. Log information associated with programs, scripts, and plug-ins can add significantly to the typical information logged by the Web server and may prove invaluable when investigating events.

Ensuring that sufficient log capacity is available is a concern because logs often take considerably more space than administrators initially estimate, especially when logging is set to a highly detailed level. Administrators should closely monitor the size of the log files when they implement different logging settings to ensure that the log files do not fill up the allocated storage. Because of the size of the log files, removing and archiving the logs more frequently or reducing the logging level of detail may be necessary.

Some Web server programs provide a capability to enforce or disable the checking of specified access controls during program startup. This level of control may be helpful, for example, to avoid inadvertent alteration of log files because of errors in file access administration. Web server administrators should determine the
circumstances under which they may wish to enable such checks (assuming the Web server software supports this feature).

Reviewing log files is a tedious and time-consuming task that informs administrators of events that have already occurred. Web server logs should also be reviewed for indications of attacks. The frequency of the reviews depends on the following factors:

- Amount of traffic the server receives
- General threat level (certain sites receive many more attacks than other sites and thus should have their logs reviewed more frequently)
- Specific threats (at certain times, specific threats arise that may require more frequent log file analysis)
- Vulnerability of the Web server
- Value of data and services provided by the Web server

Reviews should take place regularly (e.g., daily) and when a suspicious activity has been noted or a threat warning has been issued. Obviously, the task could quickly become burdensome to a Web server administrator. To reduce this burden, automated log analysis tools have been developed.

In addition, a long-term and more in-depth analysis of the logs is needed. Because a typical Web server attack can involve hundreds of unique requests, an attacker may attempt to disguise a Web server attack by increasing the interval between requests. In this case, reviewing a single day’s or week’s logs may not show recognizable trends. However, when trends are analyzed over a week, month, or quarter, multiple attacks from the same host or subnet can be more easily recognized.

Log files should be protected to ensure that if an attacker does compromise a Web server, the log files cannot be altered to cover the attack. Although encryption can be useful in protecting log files, the best solution is to store log files on a host separate from the Web server. This is often called a centralized logging server. Centralized logging is often performed using syslog, which is a standard logging protocol. Alternately, some organizations use security information and event management (SIEM) software that uses centralized servers to perform log analysis, database servers to store logs, and either agents installed on the individual hosts or processes running on the centralized servers to transfer Web server logs or log data from the hosts to the servers and parse the logs.

Log files should be backed up and archived regularly. Archiving log files for a period of time is important for several reasons, including supporting certain legal actions and troubleshooting problems with the Web server. The retention period for archived log files depends on a number of factors, including the following:

- Legal requirements
- Organizational requirements
- Size of logs (which is directly related to the traffic of the site and the number of details logged)
- Value of Web server data and services
- Threat level
11.2 Web Server Backup

One of the most important functions of a Web server administration is Backup which maintains the integrity of the data on the Web server. There are two principal components to backing up data on a Web server:

- Regular backup of the data and OS
- Maintenance of a separate protected authoritative copy of the organization’s Web content

All organizations need to create a Web Server data backup policy. Accordingly, they have to choose any of the three types of backup: Full, Differential or Incremental or their combinations. The Web Server backup policy will be different for each organization that reflect its particular environment and needs.

11.3 Maintain a Test Web Server

It is advisable to maintain a test or development Web server. Ideally, this server should have hardware and software identical to the production or live Web server and be located on an internal network segment (intranet) where it can be fully protected by the organization’s perimeter network defenses. Although the cost of maintaining an additional Web server is not inconsequential, having a test Web server offers numerous advantages:

- It provides a platform to test new patches and service packs prior to application on the production Web server.
- It provides a development platform for the Webmaster and Web server administrator to develop and test new content and applications.
- It provides a platform to test configuration settings before applying them to production Web servers.
- Software critical for development and testing but that might represent an unacceptable security risk on the production server can be installed on the development server (e.g., software compliers, administrative tool kits, remote access software).

The test Web server should be separate from the server that maintains an authoritative copy of the content on the production Web server

11.4 Maintain an Authoritative Copy of Organizational Web Content

All organizations should maintain an authoritative (i.e., verified and trusted) copy of their public Web sites on a host that is inaccessible to the Internet. This is a supplement to, but not replacement for, an appropriate backup policy. The purpose of the authoritative copy is to provide a means of restoring information on the public Web server if it is compromised as a result of an accident or malicious action. To successfully accomplish the goal of providing and protecting an authoritative copy of the Web server content, the following requirements must be met:

- Protect authoritative copy from unauthorized access.
- Establish appropriate authoritative copy update procedures.
- Establish a process for copying authoritative copy to a production Web server.
- Include the procedures for restoring from the authoritative copy in the organizational incident response procedures.
11.5 Recovering From a Security Compromise

The first step in recovering from a compromise is to create and document the required policies and procedures for responding to successful intrusions before an intrusion. The response procedures should outline the actions that are required to respond to a successful compromise of the Web server and the appropriate sequence of these actions (sequence can be critical).

A Web server administrator should follow the organization’s policies and procedures for incident handling, and the incident response team should be contacted for guidance before the organization takes any action after a suspected or confirmed security compromise. Based on the organization’s policy and procedures, system administrators should decide whether to reinstall the OS of a compromised system or restore it from a backup. As a thumb rule, if the lower the level of access is gained by the intruder and the Web server administrator understands about the attacker’s actions, the less risk there is in restoring from a backup and patching the vulnerability. For incidents in which there is less known about the attacker’s actions and/or in which the attacker gains high-level access, it is recommended that the OS and applications be reinstalled from the manufacturer’s original distribution media and that the Web server data be restored from a known good & trusted backup.

If legal action is pursued, system administrators need to be aware of the guidelines for handling a host after a compromise. Consult legal counsel and relevant law enforcement authorities as appropriate.

11.6 Security Testing Web Servers

Periodic security testing of public Web servers is critical. Without periodic testing, there is no assurance that current protective measures are working or that the security patch applied by the Web server administrator is functioning as advertised. Two techniques are primarily used for security testing of Web Servers, they are:

- **Vulnerability Scanning:** Vulnerability scanning assists a Web server administrator in identifying vulnerabilities & misconfigurations and verifying whether the existing security measures are effective. They can help to identify out-of-date software versions, missing patches, or system upgrades. In addition they can validate compliance with or deviations from the organization’s security policy.
- **Penetration testing:** It is used less frequently and usually only as part of an overall penetration test of the organization’s network. Penetration testing is security testing in which evaluators attempt to circumvent the security features of a system based on their understanding of the system design and implementation. The purpose of penetration testing is to exercise system protections (particularly human response to attack indications) by using common tools and techniques developed by attackers. It can uncover a variety of weaknesses, ineffective policies and configuration problems. This testing is highly recommended for complex or critical systems. It is a very labor-intensive activity and requires great expertise to minimize the risk to targeted systems.

11.7 Remotely Administering a Web Server

It is strongly recommended that remote administration and remote updating of content for a Web server be allowed only after careful consideration of the risks. The most secure configuration is to disallow any remote administration or content updates. However, that might not be viable for all organizations. The risk of enabling remote administration or content updates varies considerably depending on the location of the Web server on the network. For a Web server that is located behind a firewall, remote administration or content updating can be implemented relatively securely from the internal network, but not without added risk. Remote administration or
content updating should generally not be allowed from a host located outside the organization’s network unless it is performed from an organization-controlled computer through the organization’s remote access solution, such as a VPN.

If an organization determines that it is necessary to remotely administer or update content on a Web server, following these steps should ensure that content is implemented in as secure a manner as possible:

- Use a strong authentication mechanism (e.g., public/private key pair, two-factor authentication).
- Restrict which hosts can be used to remotely administer or update content on the Web server.
  - Restrict by authorized users
  - Restrict by IP address (not hostname)
  - Restrict to hosts on the internal network or those using the organization’s enterprise remote access solution.
- Use secure protocols that can provide encryption of both passwords and data (e.g., SSH, HTTPS); do not use less secure protocols (e.g., telnet, FTP, NFS, HTTP) unless absolutely required and tunneled over an encrypted protocol, such as SSH, SSL, or IPsec.
- Enforce the concept of least privilege on remote administration and content updating (e.g., attempt to minimize the access rights for the remote administration/update accounts).
- Do not allow remote administration from the Internet through the firewall unless accomplished via strong mechanisms, such as VPNs.
- Change any default accounts or passwords for the remote administration utility or application.
- Do not mount any file shares on the internal network from the Web server or vice versa.
12. References

1. NIST Special Publication 800-44: Guidelines on Securing Public Web Servers
2. Internet Information Services (IIS) 7.0 Resource Kit, by Mike Volodarsky, Olga Londer, Brett Hill, Bernard Cheah, Steve Schofield, Carlos Aguilar Mares and Kurt Meyer with the Microsoft IIS Team, Microsoft Press
3. Internet Information Services (IIS) 7.0 Administrator’s Pocket Consultant, by William R. Stanek, Microsoft Press
5. Securing Public Web Servers, CMU/SEI-SIM-011
   www.sei.cmu.edu/reports/00sim011.pdf
7. IT Infrastructure Threat Modeling Guide, Microsoft
8. Improving Web Application Security: Threats and Countermeasures
## Appendix A:
### Web Server Security Checklist

#### i) Planning and Managing Web Servers

<table>
<thead>
<tr>
<th>Completed</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Plan the configuration and deployment of the Web server</strong></td>
<td></td>
</tr>
<tr>
<td>☐</td>
<td>Identify functions of the Web server</td>
</tr>
<tr>
<td>☐</td>
<td>Identify categories of information that will be stored, processed, and transmitted through the Web server</td>
</tr>
<tr>
<td>☐</td>
<td>Identify security requirements of information</td>
</tr>
<tr>
<td>☐</td>
<td>Identify how information is published to the Web server</td>
</tr>
<tr>
<td>☐</td>
<td>Identify the security requirements of other hosts involved (e.g., backend database or Web service)</td>
</tr>
<tr>
<td>☐</td>
<td>Identify a dedicated host to run the Web server</td>
</tr>
<tr>
<td>☐</td>
<td>Identify network services that will be provided or supported by the Web server</td>
</tr>
<tr>
<td>☐</td>
<td>Identify the security requirements of any additional services provided or supported by the Web server</td>
</tr>
<tr>
<td>☐</td>
<td>Identify how the Web server will be managed</td>
</tr>
<tr>
<td>☐</td>
<td>Identify users and categories of users of the Web server and determine privilege for each category of user</td>
</tr>
<tr>
<td>☐</td>
<td>Identify user authentication methods for the Web server and how authentication data will be protected</td>
</tr>
<tr>
<td>☐</td>
<td>Identify how access to information resources will be enforced</td>
</tr>
<tr>
<td>☐</td>
<td>Identify appropriate physical security mechanisms</td>
</tr>
<tr>
<td>☐</td>
<td>Identify appropriate availability mechanisms</td>
</tr>
</tbody>
</table>

**Choose appropriate OS for Web server**

| ☐ | Minimal exposure to vulnerabilities |
| ☐ | Ability to restrict administrative or root level activities to authorized users only |
| ☐ | Ability to control access to data on the server |
| ☐ | Ability to disable unnecessary network services that may be built into the OS or server software |
| ☐ | Ability to control access to various forms of executable programs, such as CGI scripts and server plug-ins |
| ☐ | Ability to log appropriate server activities to detect intrusions and attempted intrusions |
| ☐ | Provision of a host-based firewall capability |
| ☐ | Availability of experienced staff to install, configure, secure, and maintain OS |

**Choose appropriate platform for Web server**

| ☐ | General purpose OS |
| ☐ | Trusted OS |
| ☐ | Web server appliance |
| ☐ | Pre-hardened OS and Web server |
| ☐ | Virtualized platform |
## ii) Securing the Web Server Operating System

<table>
<thead>
<tr>
<th>Completed</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Patch and upgrade OS</strong></td>
</tr>
<tr>
<td></td>
<td>Create, document, and implement a patching process</td>
</tr>
<tr>
<td></td>
<td>Keep the servers disconnected from networks or on an isolated network that severely restricts communications until all patches have been installed</td>
</tr>
<tr>
<td></td>
<td>Identify and install all necessary patches and upgrades to the OS</td>
</tr>
<tr>
<td></td>
<td>Identify and install all necessary patches and upgrades to applications and services included with the OS</td>
</tr>
<tr>
<td></td>
<td>Identify and mitigate any unpatched vulnerabilities</td>
</tr>
</tbody>
</table>

### Remove or disable unnecessary services and applications

<table>
<thead>
<tr>
<th>Completed</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Disable or remove unnecessary services and applications</td>
</tr>
</tbody>
</table>

### Configure OS user authentication

<table>
<thead>
<tr>
<th>Completed</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Remove or disable unneeded default accounts and groups</td>
</tr>
<tr>
<td></td>
<td>Disable non-interactive accounts</td>
</tr>
<tr>
<td></td>
<td>Create the user groups for the particular computer</td>
</tr>
<tr>
<td></td>
<td>Create the user accounts for the particular computer</td>
</tr>
<tr>
<td></td>
<td>Check the organization’s password policy and set account passwords appropriately (e.g., length, complexity)</td>
</tr>
<tr>
<td></td>
<td>Prevent password guessing (e.g., increase the period between attempts, deny login after a defined number of failed attempts)</td>
</tr>
<tr>
<td></td>
<td>Install and configure other security mechanisms to strengthen authentication</td>
</tr>
</tbody>
</table>

### Configure resource controls appropriately

<table>
<thead>
<tr>
<th>Completed</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Deny read access to unnecessary files and directories</td>
</tr>
<tr>
<td></td>
<td>Deny write access to unnecessary files and directories</td>
</tr>
<tr>
<td></td>
<td>Limit the execution privilege of system tools to system administrators</td>
</tr>
</tbody>
</table>

### Install and configure additional security controls

<table>
<thead>
<tr>
<th>Completed</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Select, install, and configure additional software to provide needed controls not included in the OS, such as antivirus software, antispyware software, rootkit detectors, host-based intrusion detection and prevention software, host-based firewalls, and patch management software</td>
</tr>
</tbody>
</table>

### Test the security of the OS

<table>
<thead>
<tr>
<th>Completed</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Identify a separate identical system</td>
</tr>
<tr>
<td></td>
<td>Test OS after initial install to determine vulnerabilities</td>
</tr>
<tr>
<td></td>
<td>Test OS periodically (e.g., quarterly) to determine new vulnerabilities</td>
</tr>
</tbody>
</table>
### iii) Securing the Web Server

<table>
<thead>
<tr>
<th>Completed</th>
<th>Action</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Securely install the Web server</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Install the Web server software on a dedicated host or a dedicated virtualized guest OS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Apply any patches or upgrades to correct for known vulnerabilities</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Create a dedicated physical disk or logical partition (separate from OS and Web server application) for Web content</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Remove or disable all services installed by the Web server application but not required (e.g., gopher, FTP, remote administration)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Remove or disable all unneeded default login accounts created by the Web server installation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Remove all manufacturer documentation from server</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Remove any example or test files from server, including scripts and executable code</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Apply appropriate security template or hardening script to the server</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Reconfigure HTTP service banner (and others as required) NOT to report Web server and OS type and version</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Configure OS and Web server access controls</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Configure the Web server process to run as a user with a strictly limited set of privileges</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Configure the Web server so that Web content files can be read but not written by service processes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Configure the Web server so that service processes cannot write to the directories where public Web content is stored</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Configure the Web server so that only processes authorized for Web server administration can write Web content files</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Configure the host OS so that the Web server can write log files but not read them</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Configure the host OS so that temporary files created by the Web server application are restricted to a specified and appropriately protected subdirectory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Configure the host OS so that access to any temporary files created by the Web server application is limited to the service processes that created the files</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Install Web content on a different hard drive or logical partition than the OS and Web server application</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If uploads are allowed to the Web server, configure it so that a limit is placed on the amount of hard drive space that is dedicated for this purpose; uploads should be placed on a separate partition</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ensure that log files are stored in a location that is sized appropriately; log files should be placed on a separate partition</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Configure the maximum number of Web server processes and/or network connections that the Web server should allow</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ensure that any virtualized guest OSs follow this checklist</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ensure users and administrators are able to change passwords</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Disable users after a specified period of inactivity</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ensure each user and administrator has a unique ID</td>
<td></td>
</tr>
<tr>
<td>Completed</td>
<td>Action</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Configure a secure Web content directory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dedicated a single hard drive or logical partition for Web content and establish related subdirectories exclusively for Web server content files, including graphics but excluding scripts and other programs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Define a single directory exclusively for all external scripts or programs executed as part of Web server content (e.g., CGI, ASP)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Disable the execution of scripts that are not exclusively under the control of administrative accounts. This action is accomplished by creating and controlling access to a separate directory intended to contain authorized scripts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Disable the use of hard or symbolic links (e.g., shortcuts for Windows)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Define a complete Web content access matrix. Identify which folders and files within the Web server document should be restricted and which should be accessible (and by whom)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Check the organization’s password policy and set account passwords appropriately (e.g., length, complexity)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use the robots.txt file, if appropriate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Configure anti-spambot protection, if appropriate (e.g., CAPTCHAs, nofollow, or keyword filtering)</td>
<td></td>
</tr>
</tbody>
</table>
iv) Securing Web Content

<table>
<thead>
<tr>
<th>Completed</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>Ensure that none of the following types of information are available on or through a public Web server</td>
</tr>
<tr>
<td>☐</td>
<td>Classified records</td>
</tr>
<tr>
<td>☐</td>
<td>Internal personnel rules and procedures</td>
</tr>
<tr>
<td>☐</td>
<td>Sensitive or proprietary information</td>
</tr>
<tr>
<td>☐</td>
<td>Personal information about an organization’s personnel</td>
</tr>
<tr>
<td>☐</td>
<td>Telephone numbers, e-mail addresses, or general listings of staff unless necessary to fulfill organizational requirements</td>
</tr>
<tr>
<td>☐</td>
<td>Schedules of organizational principals or their exact location (whether on or off the premises)</td>
</tr>
<tr>
<td>☐</td>
<td>Information on the composition, preparation, or optimal use of hazardous materials or toxins</td>
</tr>
<tr>
<td>☐</td>
<td>Sensitive information relating to National security</td>
</tr>
<tr>
<td>☐</td>
<td>Investigative records</td>
</tr>
<tr>
<td>☐</td>
<td>Financial records (beyond those already publicly available)</td>
</tr>
<tr>
<td>☐</td>
<td>Medical records</td>
</tr>
<tr>
<td>☐</td>
<td>Organization’s physical and information security procedures</td>
</tr>
<tr>
<td>☐</td>
<td>Information about organization’s network and information system infrastructure</td>
</tr>
<tr>
<td>☐</td>
<td>Information that specifies or implies physical security vulnerabilities</td>
</tr>
<tr>
<td>☐</td>
<td>Plans, maps, diagrams, aerial photographs, and architectural plans of organizational building, properties, or installations</td>
</tr>
<tr>
<td>☐</td>
<td>Copyrighted material without the written permission of the owner</td>
</tr>
<tr>
<td>☐</td>
<td>Privacy or security policies that indicate the types of security measures in place to the degree that they may be useful to an attacker</td>
</tr>
<tr>
<td>☐</td>
<td>Establish an organizational-wide documented formal policy and process for approving public Web content that—</td>
</tr>
<tr>
<td>☐</td>
<td>Identifies information that should be published on the Web</td>
</tr>
<tr>
<td>☐</td>
<td>Identifies target audience</td>
</tr>
<tr>
<td>☐</td>
<td>Identifies possible negative ramifications of publishing the information</td>
</tr>
<tr>
<td>☐</td>
<td>Identifies who should be responsible for creating, publishing, and maintaining this particular information</td>
</tr>
<tr>
<td>☐</td>
<td>Provides guidelines on styles and formats appropriate for Web publishing</td>
</tr>
<tr>
<td>☐</td>
<td>Provides for appropriate review of the information for sensitivity and distribution/release controls (including the sensitivity of the information in aggregate)</td>
</tr>
<tr>
<td>☐</td>
<td>Determines the appropriate access and security controls</td>
</tr>
<tr>
<td>☐</td>
<td>Provides guidance on the information contained within the source code of the Web content</td>
</tr>
</tbody>
</table>

Maintain Web user privacy

| ☐         | Maintain a published privacy policy |
| ☐         | Prohibit the collection of personally identifying data without the explicit permission of the user and collect only the data that is absolutely needed |
| ☐         | Prohibit the use of “persistent” cookies |
| ☐         | Use the session cookie only if it is clearly identified in published privacy policy |
### Mitigate indirect attacks on content

<table>
<thead>
<tr>
<th>Completed</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ensure users of the site are aware of the dangers of phishing and pharming attacks and how to avoid them</td>
</tr>
<tr>
<td></td>
<td>Validate official communication by personalizing emails and providing unique identifying (but not confidential) information only the organization and user should know</td>
</tr>
<tr>
<td></td>
<td>Use digital signatures on e-mail if appropriate</td>
</tr>
<tr>
<td></td>
<td>Perform content validation within the Web application to prevent more sophisticated phishing attacks (e.g., cross-site scripting based attacks)</td>
</tr>
<tr>
<td></td>
<td>Personalize Web content to aid in users’ identifying fraudulent Web sites</td>
</tr>
<tr>
<td></td>
<td>Use token-based or mutual authentication if applicable</td>
</tr>
<tr>
<td></td>
<td>Suggest the use of Web browsers or browser toolbars with phishing/pharming protection</td>
</tr>
<tr>
<td></td>
<td>Use current versions of DNS software with the latest security patches</td>
</tr>
<tr>
<td></td>
<td>Install server-side DNS protection mechanisms</td>
</tr>
<tr>
<td></td>
<td>Monitor organizational domains and similar domains</td>
</tr>
<tr>
<td></td>
<td>Simplify the structure of organization domain names</td>
</tr>
<tr>
<td></td>
<td>Use secure connections for logins</td>
</tr>
<tr>
<td></td>
<td>If necessary, engage a vendor to provide stronger anti-phishing/anti-pharming measures</td>
</tr>
</tbody>
</table>

### Client-side active content security considerations

<table>
<thead>
<tr>
<th>Completed</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Weigh the risks and benefits of client-side active content</td>
</tr>
<tr>
<td></td>
<td>Take no actions without the express permission of user</td>
</tr>
<tr>
<td></td>
<td>When possible, only use widely-adopted active content such as JavaScript, PDF, and Flash</td>
</tr>
<tr>
<td></td>
<td>When possible, provide alternatives (e.g., HTML provided along with PDF)</td>
</tr>
</tbody>
</table>

### Maintain server-side active content security

<table>
<thead>
<tr>
<th>Completed</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Only simple, easy-to-understand code should be used</td>
</tr>
<tr>
<td></td>
<td>Limited or no reading or writing to the file system should be permitted</td>
</tr>
<tr>
<td></td>
<td>Limited or no interaction with other programs (e.g., sendmail) should be permitted</td>
</tr>
<tr>
<td></td>
<td>There should be no requirement to run with suid privileges on Unix or Linux</td>
</tr>
<tr>
<td></td>
<td>Explicit path names should be used (i.e., does not rely on path variable)</td>
</tr>
<tr>
<td></td>
<td>No directories have both write and execute permissions</td>
</tr>
<tr>
<td></td>
<td>All executable files are placed in a dedicated folders</td>
</tr>
<tr>
<td></td>
<td>SSIs are disabled or the execute function is disabled</td>
</tr>
<tr>
<td></td>
<td>All user input is validated</td>
</tr>
<tr>
<td></td>
<td>Web content generation code should be scanned or audited</td>
</tr>
<tr>
<td></td>
<td>Dynamically created pages do not create dangerous metacharacters</td>
</tr>
<tr>
<td></td>
<td>Character set encoding should be explicitly set in each page</td>
</tr>
<tr>
<td></td>
<td>User data should be scanned to ensure it contains only expected input, (e.g., a-z, A-Z, 0-9); care should be taken with special characters or HTML tags</td>
</tr>
<tr>
<td></td>
<td>Cookies should be examined for any special characters</td>
</tr>
<tr>
<td></td>
<td>Encryption mechanism is used to encrypt passwords entered through scripts forms</td>
</tr>
<tr>
<td></td>
<td>For Web applications that are restricted by username and password, none of the Web pages in the application should be accessible without executing the appropriate login process</td>
</tr>
<tr>
<td></td>
<td>All sample scripts are removed</td>
</tr>
<tr>
<td></td>
<td>No third-party scripts or executable code are used without verifying the source code</td>
</tr>
</tbody>
</table>
Using Authentication and Encryption Technologies for Web Servers

<table>
<thead>
<tr>
<th>Completed</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Configure Web authentication and encryption technologies</strong></td>
</tr>
<tr>
<td></td>
<td>For Web resources that require minimal protection and for which there is a small, clearly defined audience, configure address-based authentication</td>
</tr>
<tr>
<td></td>
<td>For Web resources that require additional protection but for which there is a small, clearly defined audience, configure address-based authentication as a second line of defense</td>
</tr>
<tr>
<td></td>
<td>For Web resources that require minimal protection but for which there is no clearly defined audience, configure basic or digest authentication (better)</td>
</tr>
<tr>
<td></td>
<td>For Web resources that require protection from malicious bots, configure basic or digest authentication (better) or implement mitigation techniques.</td>
</tr>
<tr>
<td></td>
<td>For organizations required to comply with FIPS 140-2, ensure the SSL/TLS implementation is FIPS-validated</td>
</tr>
<tr>
<td></td>
<td>For Web resources that require maximum protection, configure SSL/TLS</td>
</tr>
<tr>
<td></td>
<td><strong>Configure SSL/TLS</strong></td>
</tr>
<tr>
<td></td>
<td>Ensure the SSL/TLS implementation is fully patched</td>
</tr>
<tr>
<td></td>
<td>Use a third-party issued certificate for server authentication (unless all systems using the server are organization-managed, in which case a self-signed certificate could potentially be used instead)</td>
</tr>
<tr>
<td></td>
<td>For configurations that require a medium level of client authentication, configure server to require username and password via SSL/TLS</td>
</tr>
<tr>
<td></td>
<td>For configurations that require a high level of client authentication, configure server to require client certificates via SSL/TLS</td>
</tr>
<tr>
<td></td>
<td>Ensure weak cipher suites are disabled.</td>
</tr>
<tr>
<td></td>
<td>Configure file integrity checker to monitor Web server certificate</td>
</tr>
<tr>
<td></td>
<td>If only SSL/TLS is to be used in the Web server, ensure access via any TCP port other than 443 is disabled</td>
</tr>
<tr>
<td></td>
<td>If most traffic to the Web server will be via encrypted SSL/TLS, ensure that appropriate logging and detection mechanisms are employed in the Web server (because network monitoring is ineffective against encrypted SSL/TLS sessions)</td>
</tr>
<tr>
<td></td>
<td><strong>Protect against brute force attacks</strong></td>
</tr>
<tr>
<td></td>
<td>Use strong authentication if possible</td>
</tr>
<tr>
<td></td>
<td>Use a delay after failed login attempts</td>
</tr>
<tr>
<td></td>
<td>Lock out an account after a set number of failed login attempts</td>
</tr>
<tr>
<td></td>
<td>Enforce a password policy</td>
</tr>
<tr>
<td></td>
<td>Blacklist IP addresses or domains known to attempt brute force attacks</td>
</tr>
<tr>
<td></td>
<td>Use log monitoring software to detect brute force attacks</td>
</tr>
</tbody>
</table>
vi) Implementing a Secure Network Infrastructure

<table>
<thead>
<tr>
<th>Completed</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ]</td>
<td>Identify network location</td>
</tr>
<tr>
<td></td>
<td>Web server is located in a DMZ, or Web server hosting is outsourced</td>
</tr>
<tr>
<td>[ ]</td>
<td>Assess firewall configuration</td>
</tr>
<tr>
<td></td>
<td>Web server is protected by a firewall; if it faces a higher threat or is more vulnerable, it is protected by an application layer firewall</td>
</tr>
<tr>
<td></td>
<td>Firewall controls all traffic between the Internet and the Web server</td>
</tr>
<tr>
<td></td>
<td>Firewall blocks all inbound traffic to the Web server except TCP ports 80 (HTTP) and/or 443 (HTTPS), if required</td>
</tr>
<tr>
<td></td>
<td>Firewall blocks (in conjunction with the IDPS) IP addresses or subnets that the IDPS reports are attacking the organizational network</td>
</tr>
<tr>
<td></td>
<td>Firewall notifies the network or Web server administrator of suspicious activity through an appropriate means</td>
</tr>
<tr>
<td></td>
<td>Firewall provides content filtering (application layer firewall)</td>
</tr>
<tr>
<td></td>
<td>Firewall is configured to protect against DoS attacks</td>
</tr>
<tr>
<td></td>
<td>Firewall detects malformed or known attack URL requests</td>
</tr>
<tr>
<td></td>
<td>Firewall logs critical events</td>
</tr>
<tr>
<td></td>
<td>Firewall and firewall OS are patched to latest or most secure level</td>
</tr>
<tr>
<td>[ ]</td>
<td>Evaluate intrusion detection and prevention systems</td>
</tr>
<tr>
<td></td>
<td>Host-based IDPS is used for Web servers that operate primarily using SSL/TLS</td>
</tr>
<tr>
<td></td>
<td>IDPS is configured to monitor network traffic to and from the Web server after firewall</td>
</tr>
<tr>
<td></td>
<td>IDPS is configured to monitor changes to critical files on Web server (host-based IDPS or file integrity checker)</td>
</tr>
<tr>
<td></td>
<td>IDPS blocks (in conjunction with the firewall) IP addresses or subnets that are attacking the organizational network</td>
</tr>
<tr>
<td></td>
<td>IDPS notifies the IDPS administrators or Web server administrator of attacks through appropriate means</td>
</tr>
<tr>
<td></td>
<td>IDPS is configured to maximize detection with an acceptable level of false positives</td>
</tr>
<tr>
<td></td>
<td>IDPS is configured to log events</td>
</tr>
<tr>
<td></td>
<td>IDPS is updated with new attack signatures frequently (e.g., on a daily basis)</td>
</tr>
<tr>
<td></td>
<td>Host-based IDPS is configured to monitor the system resources available in the Web server host</td>
</tr>
<tr>
<td>[ ]</td>
<td>Assess network switches</td>
</tr>
<tr>
<td></td>
<td>Switches are used to protect against network eavesdropping</td>
</tr>
<tr>
<td></td>
<td>Switches are configured in high-security mode to defeat ARP spoofing and ARP poisoning attacks</td>
</tr>
<tr>
<td></td>
<td>Switches are configured to send all traffic on network segment to network-based IDPS</td>
</tr>
<tr>
<td>[ ]</td>
<td>Evaluate load balancers</td>
</tr>
<tr>
<td></td>
<td>Load balancers are used to increase Web server availability</td>
</tr>
<tr>
<td></td>
<td>Load balancers are augmented by Web caches if applicable</td>
</tr>
<tr>
<td>[ ]</td>
<td>Evaluate reverse proxies</td>
</tr>
<tr>
<td></td>
<td>Reverse proxies are used as a security gateway to increase Web server availability</td>
</tr>
<tr>
<td></td>
<td>Reverse proxies are augmented with encryption acceleration, user authentication, and content filtering capabilities, if applicable</td>
</tr>
</tbody>
</table>
### vii) Administering the Web Server

<table>
<thead>
<tr>
<th>Completed</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Perform logging</strong></td>
</tr>
<tr>
<td></td>
<td>Use the combined log format for storing the Transfer Log or manually configure the information described by the combined log format to be the standard format for the Transfer Log</td>
</tr>
<tr>
<td></td>
<td>Enable the Referrer Log or Agent Log if the combined log format is unavailable</td>
</tr>
<tr>
<td></td>
<td>Establish different log file names for different virtual Web sites that may be implemented as part of a single physical Web server</td>
</tr>
<tr>
<td></td>
<td>Use the remote user identity as specified in <a href="https://www.rfc-editor.org/rfc/rfc1413">RFC 1413</a></td>
</tr>
<tr>
<td></td>
<td>Store logs on a separate (syslog) host</td>
</tr>
<tr>
<td></td>
<td>Ensure there is sufficient capacity for the logs</td>
</tr>
<tr>
<td></td>
<td>Archive logs according to organizational requirements</td>
</tr>
<tr>
<td></td>
<td>Review logs daily</td>
</tr>
<tr>
<td></td>
<td>Review logs weekly (for more long-term trends)</td>
</tr>
<tr>
<td></td>
<td>Use automated log file analysis tool(s)</td>
</tr>
<tr>
<td></td>
<td><strong>Perform Web server backups</strong></td>
</tr>
<tr>
<td></td>
<td>Create a Web server backup policy</td>
</tr>
<tr>
<td></td>
<td>Back up Web server differentially or incrementally on a daily to weekly basis</td>
</tr>
<tr>
<td></td>
<td>Back up Web server fully on a weekly to monthly basis</td>
</tr>
<tr>
<td></td>
<td>Periodically archive backups</td>
</tr>
<tr>
<td></td>
<td>Maintain an authoritative copy of Web site(s)</td>
</tr>
<tr>
<td></td>
<td><strong>Recover from a compromise</strong></td>
</tr>
<tr>
<td></td>
<td>Report the incident to the organization’s computer incident response capability</td>
</tr>
<tr>
<td></td>
<td>Isolate the compromised system(s) or take other steps to contain the attack so additional information can be collected</td>
</tr>
<tr>
<td></td>
<td>Investigate similar hosts to determine if the attacker has also compromised other systems</td>
</tr>
<tr>
<td></td>
<td>Consult, as appropriate, with management, legal counsel, and law enforcement officials expeditiously</td>
</tr>
<tr>
<td></td>
<td>Analyze the intrusion</td>
</tr>
<tr>
<td></td>
<td>Restore the system</td>
</tr>
<tr>
<td></td>
<td>Test system to ensure security</td>
</tr>
<tr>
<td></td>
<td>Reconnect system to network</td>
</tr>
<tr>
<td></td>
<td>Monitor system and network for signs that the attacker is attempting to access the system or network again</td>
</tr>
<tr>
<td></td>
<td>Document lessons learned</td>
</tr>
<tr>
<td></td>
<td><strong>Test security</strong></td>
</tr>
<tr>
<td></td>
<td>Periodically conduct vulnerability scans on Web server, dynamically generated content, and supporting network</td>
</tr>
<tr>
<td></td>
<td>Update vulnerability scanner prior to testing</td>
</tr>
<tr>
<td></td>
<td>Correct any deficiencies identified by the vulnerability scanner</td>
</tr>
<tr>
<td></td>
<td>Conduct penetration testing on the Web server and the supporting network infrastructure</td>
</tr>
<tr>
<td></td>
<td>Correct deficiencies identified by penetration testing</td>
</tr>
<tr>
<td>Completed</td>
<td>Action</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td>Conduct remote administration and content updates</td>
</tr>
<tr>
<td></td>
<td>Use a strong authentication mechanism (e.g., public/private key pair, two-factor authentication)</td>
</tr>
<tr>
<td></td>
<td>Restrict hosts that can be used to remotely administer or update content on the Web server by IP address and to the internal network</td>
</tr>
<tr>
<td></td>
<td>Use secure protocols (e.g., SSH, HTTPS)</td>
</tr>
<tr>
<td></td>
<td>Enforce the concept of least privilege on remote administration and content updating (e.g., attempt to minimize the access rights for the remote administration/update accounts)</td>
</tr>
<tr>
<td></td>
<td>Change any default accounts or passwords from the remote administration utility or application</td>
</tr>
<tr>
<td></td>
<td>Do not allow remote administration from the Internet unless mechanisms such as VPNs are used</td>
</tr>
<tr>
<td></td>
<td>Do not mount any file shares on the internal network from the Web server or vice versa</td>
</tr>
</tbody>
</table>
## Appendix B:
### Function of Built-In Modules with Security Implications

<table>
<thead>
<tr>
<th>Module</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anonymous Authentication Module</td>
<td>Authentication the request with an anonymous user if no other</td>
</tr>
<tr>
<td></td>
<td>mechanism is present. If removed: Access to resources will be denied for anonymous requests.</td>
</tr>
<tr>
<td>Basic Authentication Module</td>
<td>Supports basic authentication. If removed: Clients will not be able to authenticate with basic authentication.</td>
</tr>
<tr>
<td>Certificate Mapping Authentication Module</td>
<td>Supports client certificate authentication. If removed: Clients will not be able to authenticate with client certificates.</td>
</tr>
<tr>
<td>Configuration Validation Module</td>
<td>Validates ASP.NET configuration in integrated mode. If removed: Strong warning—ASP.NET applications that define modules and handlers using legacy configuration will silently run in integrated mode, but the modules will not be loaded. This may result in unexpected application behavior and security vulnerabilities for unmigrated ASP.NET applications.</td>
</tr>
<tr>
<td>CustomError Module</td>
<td>Detailed error messages will not be generated for IIS errors. If removed: Strong warning—Sensitive application error information may be sent to remote clients.</td>
</tr>
<tr>
<td>Default Authentication</td>
<td>Supports the ASP.NET DefaultAuthentication_OnAuthenticate event. When ASP.NET is configured to use the Forms Authentication mode, removing this module may lead to errors in other ASP.NET modules during anonymous requests. If removed: Warning—The DefaultAuthentication_OnAuthenticate event will not be raised, so any custom authentication code depending on this event will not run. This is not common.</td>
</tr>
<tr>
<td>Digest Authentication Module</td>
<td>Supports digest authentication. If removed: Clients will not be able to use digest authentication to authenticate.</td>
</tr>
<tr>
<td>File Authorization</td>
<td>Verifies that the authenticated client has access to physical resources. If removed: Strong warning—Access may be granted to resources that deny access to the authenticated user.</td>
</tr>
<tr>
<td>Forms Authentication</td>
<td>Supports forms-based authentication. If removed: Clients will not be able to use forms authentication to authenticate.</td>
</tr>
<tr>
<td>HttpCache Module</td>
<td>Supports IIS output caching and kernel caching of responses. If removed: Warning—Response output caching will not occur, possibly resulting in increased load on the server and in the worst case Denial of Service (DoS) conditions.</td>
</tr>
<tr>
<td>HttpLogging Module</td>
<td>Supports request logging. If removed: Warning—Requests may not be logged.</td>
</tr>
<tr>
<td>IISCertificate Mapping Authentication Module</td>
<td>Supports IIS configuration–based client certificate authentication. If removed: Clients may not be able to authenticate with client certificates against IIS configuration.</td>
</tr>
<tr>
<td>Module</td>
<td>Purpose and Removal Effect</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| HttpRedirection Module | **Purpose:** Supports configuration-based redirection rules.  
If **removed:** *Warning*—If the application depends on redirection rules for restricting access to content, removing this module may make otherwise protected resources available. |
| IsapiFilter Module      | **Purpose:** Supports ISAPI filters.  
If **removed:** *Strong warning*—ISAPI filters that enforce access or have other security functionality will not run.                                                                                           |
| OutputCache            | **Purpose:** Supports ASP.NET response output caching.  
If **removed:** *Warning*—ASP.NET response output caching will not occur, possibly resulting in increased load on the server and in the worst case Denial of Service (DoS) conditions.                                      |
| RequestFiltering Module| **Purpose:** Enforces various request restrictions and protects hidden content.  
If **removed:** *Strong warning*—Removing this module may result in protected content being served. It may also lead to nonspecific security vulnerabilities resulting from allowing unrestricted request input into the application. |
| RoleManager            | **Purpose:** Supports the ASP.NET roles functionality.  
If **removed:** *Strong warning*—Roles for the authenticated user may not be available, which may cause authorization decisions to be affected. Typically, this will only restrict access, but in some cases where access is denied based on roles, this may grant access to otherwise unauthorized users. |
| Static Compression Module | **Purpose:** Supports compression of static resources.  
If **removed:** *Warning*—Removing this module may result in significantly higher bandwidth for the site, because compression of static resources will be disabled.                                |
| Url Authorization       | **Purpose:** Supports declarative access rules.  
If **removed:** *Strong warning*—URL authorization access rules will be ignored, and access may be granted to unauthorized users.                                                                                   |
| Url Authorization Module | **Purpose:** Supports declarative ASP.NET access rules.  
If **removed:** *Strong warning*—ASP.NET url authorization access rules will be ignored, and access may be granted to unauthorized users.                                                                 |
| Windows Authentication  | **Purpose:** Supports NTLM and Kerberos authentication.  
If **removed:** Clients will be unable to authenticate with NTLM or Kerberos Windows authentication.                                                                                                       |
| Windows Authentication Module | **Purpose:** Supports raising the ASP.NET WindowsAuthentication_OnAuthentication event.  
If **removed:** *Warning*—WindowsAuthentication_OnAuthentication event will not be raised, so any custom ASP.NET authentication code dependent on this event will not run. Note that this module is not required for Windows authentication. |

SECURING IIS 7.0 WEB SERVER  
Page 128 of 128