.NET Framework 4 - ASP.NET

**Web Application Security at Run Time**

Developing your application requires that you work with one set of security issues. The other set of issues — and the ones that are generally more prominent in any discussion of Web security — address the security of your application when it is deployed and running.

Web applications by definition allow users access to a central resource — the Web server — and through it, to others such as database servers. By understanding and implementing proper security measures, you:

* Guard your own resources against unauthorized access.
* Restrict access levels by user or role.
* Establish data integrity and confidentiality, providing a relatively secure environment in which your users are comfortable working with your application.
* Establish control over how your application can get access to restricted resources.
* Ensure that your application's code runs as intended.

This topic provides a general discussion of how you can accomplish these goals, and it includes links to additional topics where you can get more details on the technologies involved.

You can help protect your application from unauthorized access by taking advantage of these types of security features:

* Security features offered by Internet Information Services (IIS) as part of its general Web server function. These include Windows file, computer, and user-level security.
* Security you can build into your ASP.NET application to provide application-specific access.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifSecurity Process in ASP.NET

IIS provides many security options for Web sites. However, IIS security mechanisms are very generic, in that the same mechanisms are used for all applications. Moreover, IIS security options — for example, using Windows Integrated security — might not always be convenient for your application. (For intranet applications, on the other hand, you might want to use Windows Integrated security for simplicity.)

Therefore, in order to provide access to specific portions of your application, you can use ASP.NET security. ASP.NET security works in conjunction with IIS security but extends it so that you can customize features, such as how to get user credentials.

IIS receives requests first from clients and performs any security checks that have been established for your application using IIS management tools. For example, if the application has been configured in IIS to allow anonymous access, IIS performs no credentials check. After performing this initial check of authentication, IIS sends the request to ASP.NET, which can perform a second level of checking. ASP.NET allows you to specify access restrictions in your application using a variety of criteria: you can restrict access to specific pages, or to specific users, and so on.

**Authentication**

The following table describes the authentication methods that are supported by ASP.NET. Several of these overlap with IIS authentication. For details, see [ASP.NET Authentication](http://msdn.microsoft.com/en-us/library/eeyk640h.aspx).

|  |  |
| --- | --- |
| **Authentication Type** | **Description** |
| Anonymous access | For applications where unknown users will be making requests (typically, public Web applications). Overlaps with IIS. |
| Basic and Digest authentication | (IIS Security Option) In this scenario, users without credentials are prompted to supply a user name and password. |
| Windows Integrated Security (also known as NTLM security) | (IIS Security Option) If the user making the request has already been authenticated in a Windows-based network, IIS can pass the user's credentials through when requesting access to a resource. |
| Certificate authentication | (IIS Security Option) In this scenario, the client has a certificate — a digital identification — that has been obtained from a third-party source. The identity mapped to the client certificate is passed to ASP.NET. |
| Kerberos | (IIS Security Option) The Kerberos authentication protocol defines the interactions between a client and a network Authentication Service known as a Key Distribution Center (KDC). Windows 2000 and 2003 implement a KDC as the authentication service on each domain controller. |
| Windows authentication | (ASP.NET Security Option) Integrates with the previously listed IIS security options. ASP.NET takes the security token created by IIS and makes it available as a [WindowsPrincipal](http://msdn.microsoft.com/en-us/library/system.security.principal.windowsprincipal.aspx) object set as the value of the [User](http://msdn.microsoft.com/en-us/library/system.web.httpcontext.user.aspx) property of the current [HttpContext](http://msdn.microsoft.com/en-us/library/system.web.httpcontext.aspx). |
| Forms authentication | (ASP.NET Security Option) If a user needs to be authenticated, ASP.NET redirects the request to a page that you specify. This page usually contains a form in which you get user name information. (For extra security, the form can be exchanged using HTTPS protocol.) When your application gets the form information, it can perform an application-specific check of the user's credentials. An important point is that the process of authentication is under your control (unlike that for IIS), which allows you to specify what the form looks like and how you choose to store user information.  If a user is successfully authenticated, ASP.NET issues an encrypted cookie containing a token identifying the user for subsequent access. |

Forms authentication is the easiest choice for ASP.NET applications on the public internet because it gives you substantial control over how users are authenticated and it allows you to store authentication in a token on the browser.

For details about IIS security, see the access-control topics in the [Windows Server TechCenter for IIS](http://go.microsoft.com/fwlink/?LinkId=37112) on the Microsoft TechNet Web site. For details about ASP.NET authentication, see [ASP.NET Authentication](http://msdn.microsoft.com/en-us/library/eeyk640h.aspx).

For details about using Forms authentication with protocol transition and constrained delegation in a Windows Server 2003 domain environment, see [Kerberos Protocol Transition and Constrained Delegation](http://go.microsoft.com/fwlink/?LinkId=50220).

**Authorization**

When your Web application runs, it requests resources from the Web server and, often, from other processes as well, such as a database. The ASP.NET process runs in a user context that determines how your application will request those resources. The ASP.NET process runs as a special local user named ASPNET (by default) on Windows 2000 and Windows XP Professional Edition, or it runs as the identity of the application pool for the ASP.NET application on Windows 2003 (by default, the local NETWORK SERVICE account). These accounts run with limited permissions. You can specify a different user context for the ASP.NET process, including the local SYSTEM account (which runs your application in administrator context) or a user whose credentials you explicitly provide, though this is not recommended.

In your ASP.NET application, you can specify that different users have authorized access to different resources. If your application is using Windows authentication, you can use Windows permissions to determine authorization to access specific files or folders on the server.

Alternatively, you can use URL-based authorization, in which authorization can be granted or denied according to different criteria:

* Specific users, or identities, which are based on the credentials provided by the user.
* Roles, which are entities defined to allow multiple users to share privileges based on a common role or function.
* Verbs, which are the HTTP processes (such as GET and POST) for accessing portions of your application.

For example, you can specify that all users can get pages (perform the GET verb) from your application, but that only specific users can post pages to it. Similarly, you might specify that all users are allowed to GET pages but specific roles are denied the right to post.

You can grant URL authorization for the application as a whole or on a directory-by-directory basis. A typical use is to allow all users to view pages in a public directory, but to keep restricted pages in a different directory that is authorized only for specific users or roles.

|  |
| --- |
| **Description: NoteNote** |
| By default, static files, such as images and style sheets, are not subject to ASP.NET authorization when they are served through IIS. You can use IIS security features to restrict access to static files if you do not want all users to be able to access the files. If you use the ASP.NET Development Server to test your ASP.NET application, you will see different behavior because static files are subject to ASP.NET authorization and will not be served to an anonymous user when anonymous access to those files is disabled. Alternatively, you can map static file name extensions in IIS to the ASP.NET ISAPI extension, in which case the ASP.NET authorization rules will apply. |

For more information, see [ASP.NET Authorization](http://msdn.microsoft.com/en-us/library/wce3kxhd.aspx) and [Basic Security Practices for Web Applications](http://msdn.microsoft.com/en-us/library/zdh19h94.aspx).

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifASP.NET Configuration Files

You establish ASP.NET security choices using settings in a Web.config file. The file allows you to include predefined elements for various security options, including sections for authentication and authorization. The relevant sections of the Web.config file might look like the following.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl30_ctl00_ctl00_code');" \o "Copy Code)

<configuration>

<system.web>

<authentication mode="Forms">

<forms loginUrl="login.aspx" />

</authentication>

<authorization>

<deny user="?" />

</authorization>

</system.web>

</configuration>

Your application can contain more than one configuration file. By default, there is a configuration file at the application root that specifies security for the application as a whole — that is, security settings are inherited by subfolders. However, you can also create configuration files in individual folders to create security settings for that folder.

For more information, see [ASP.NET Security Architecture](http://msdn.microsoft.com/en-us/library/yedba920.aspx).

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifXML Web Service Security

XML Web services using .asmx files run as Web applications using ASP.NET and therefore participate in the same security model as any ASP.NET application. For example, an XML Web service might be configured to use Basic authentication or Windows integrated security.

At design time, when you attempt to add a reference to an XML Web service — that is, when you request the XML Web service's discovery documents — the XML Web service will perform standard Web application authentication according to how it was configured. For example, if the XML Web service is configured to use Basic authentication, it will expect to get a user name and password from the requesting client. If the XML Web service is using Basic authentication, for example, the Add Web Reference dialog box will prompt you for credentials.

If you are building an application that includes calls to an XML Web service, you need to be sure that you have appropriate credentials before making the call, or the call will fail. At run time, you can pass credentials to the XML Web service by setting the [Credentials](http://msdn.microsoft.com/en-us/library/system.web.services.protocols.webclientprotocol.credentials.aspx) property of the client-side object representing the XML Web service before you call its methods.

Because other ASP.NET security options might not be sufficiently flexible, XML Web services can implement a custom authentication solution in which credential information is passed in SOAP headers. In this solution, credentials are passed in an optional part of the message exchanged between client and server. You can then write a custom HTTP module (implementing the [IHttpModule](http://msdn.microsoft.com/en-us/library/system.web.ihttpmodule.aspx) interface) that can listen for the information in the header and call your authentication code.

As with other ASP.NET applications, the XML Web service might implement specific role-based authorizations to limit access to specific parts of the application.

For details, see [XML Web Services Infrastructure](http://msdn.microsoft.com/en-us/library/sd5s0c6d.aspx) and [Securing XML Web Services Created Using ASP.NET](http://msdn.microsoft.com/en-us/library/w67h0dw7.aspx).

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifEstablishing Data Integrity and Confidentiality

Authentication and authorization establish who your users are and what resources they can access. These security features are designed primarily to help protect your Web application from unauthorized use.

However, there is a separate aspect to security as well, which is to help protect your users' information and provide them with the confidence that they can exchange sensitive information with you. For example, if your application asks users for credit card or other account numbers, personal information, or any data that users might not want others to know, you must provide a way for them to submit this information to you securely.

You can use Secure Sockets Layer (SSL) in IIS to exchange encrypted information using the HTTPS protocol. SSL provides encryption in both directions: your information is transmitted to the user using encryption, and information the user posts to your application is likewise encrypted.

**Establishing SSL and Encryption**

To use SSL and encryption, you must obtain a server certificate for your company or identity. The certificate is a digital signature that identifies your site in a way that cannot be impersonated. For Internet (public) applications, you obtain a server certificate from a recognized third-party certification authority. For private (intranet) applications, you can issue a server certificate yourself. You might do this to help secure an internal application, such as a personnel site.

Your server certificate also enables to setup SSL-encrypted connections with your browser users. SSL uses an encryption method called *public key encryption*. In this form of encryption, there are two keys: a public key used to encrypt data and a private key that you keep secret and use to decrypt information encrypted with the public key. The server certificate you obtain includes a public key. When users want to use SSL, your application sends the certificate and the public key to the browser. The browser and server then use the public key to establish a way to encrypt their information exchange.

|  |
| --- |
| **Description: NoteNote** |
| Using SSL requires that the browser support an encryption key that is at least 40 bits long. This level is available in most browsers. However this key length is not considered secure. You can optionally configure your application in IIS to only allow SSL connections with a 128-bit key. |

Once you have obtained a server certificate, you can instruct users to use SSL by having them use https:// as the prefix for getting and posting Web pages. Your Web site can also be configured in IIS to only accept HTTPS connections. IIS and the browser will automatically use an encrypted channel to exchange information.

For details about how to use SSL, see article Q307267, "How to: Secure XML Web Services with Secure Sockets Layer in Windows 2000" in the [Microsoft Knowledge Base](http://go.microsoft.com/fwlink/?LinkId=37115). For information about encryption, see [Cryptographic Services](http://msdn.microsoft.com/en-us/library/92f9ye3s.aspx). For information about certificates and about configuring SSL, see the [Windows Server TechCenter for IIS](http://go.microsoft.com/fwlink/?LinkId=37112) on the Microsoft TechNet Web site.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifUsing .NET Code Security

As a final aspect of security, you should take measures to help ensure that the code in your application is protected from misuse, whether by being inadvertently run in an improper context or by being used in a malicious way. Because ASP.NET is part of the .NET Framework, you can also take advantage of code access security to establish permissions for what code is allowed to do. For information, see [Code Access Security](http://msdn.microsoft.com/en-us/library/c5tk9z76.aspx).