# ASP.NET Web Application Security

<http://msdn.microsoft.com/en-us/library/vstudio/330a99hc(v=vs.100).aspx>

ASP.NET Web Application Security

ASP.NET, in conjunction with Microsoft Internet Information Services (IIS), can authenticate user credentials such as names and passwords using any of the following authentication methods:

* Windows: Basic, digest, or Integrated Windows Authentication (NTLM or Kerberos).
* Forms authentication, in which you create a login page and manage authentication in your application.
* Client Certificate authentication

ASP.NET controls access to site information by comparing authenticated credentials, or representations of them, to NTFS file system permissions or to an XML file that lists authorized users, authorized roles (groups), or authorized HTTP verbs.

# How ASP.NET Security Works

Securing Web sites is a critical, complex issue for Web developers. Protecting a site requires careful planning, and Web site administrators and programmers must have a clear understanding of the options for securing their site.

ASP.NET works in concert with the Microsoft .NET Framework and Microsoft Internet Information Services (IIS) to help provide Web application security. To help protect your ASP.NET application, you should perform the two fundamental functions described in the following table.

|  |  |
| --- | --- |
| **Security function** | **Description** |
| [Authentication](http://msdn.microsoft.com/en-us/library/vstudio/eeyk640h(v=vs.100).aspx) | Helps to verify that the user is, in fact, who the user claims to be. The application obtains credentials  (various forms of identification, such as name and password) from a user and validates those credentials  against some authority. If the credentials are valid, the entity that submitted the credentials is considered  an authenticated identity. |
| [Authorization](http://msdn.microsoft.com/en-us/library/vstudio/wce3kxhd(v=vs.100).aspx) | Limits access rights by granting or denying specific permissions to an authenticated identity. |

IIS can also grant or deny access based on a user's host name or IP address. Any further access authorization is performed by NTFS file access permission's URL authorization.

## ASP.NET Authentication

Authentication is the process of obtaining identification credentials such as name and password from a user and validating those credentials against some authority. If the credentials are valid, the entity that submitted the credentials is considered an authenticated identity. Once an identity has been authenticated, the authorization process determines whether that identity has access to a given resource.

ASP.NET implements authentication through authentication providers, the code modules that contain the code necessary to authenticate the requestor's credentials. The topics in this section describe the authentication providers built into ASP.NET.

|  |  |
| --- | --- |
| **Term** | **Definition** |
| [Windows Authentication Provider](http://msdn.microsoft.com/en-us/library/vstudio/907hb5w9(v=vs.100).aspx) | Provides information on how to use Windows authentication in conjunction with Microsoft  Internet Information Services (IIS) authentication to secure ASP.NET applications. |
| [Forms Authentication Provider](http://msdn.microsoft.com/en-us/library/vstudio/9wff0kyh(v=vs.100).aspx) | Provides information on how to create an application-specific login form and perform  authentication using your own code. A convenient way to work with forms authentication is to  use ASP.NET membership and ASP.NET login controls, which together provide a way to collect  user credentials, authenticate them, and manage them, using little or no code. For more  information, see [Managing Users by Using Membership](http://msdn.microsoft.com/en-us/library/vstudio/tw292whz(v=vs.100).aspx) and [ASP.NET Login Controls Overview](http://msdn.microsoft.com/en-us/library/vstudio/ms178329(v=vs.100).aspx). |

### Windows Authentication Provider

Windows Authentication treats the user identity supplied by Microsoft Internet Information Services (IIS) as the authenticated user in an ASP.NET application. IIS provides a number of authentication mechanisms to verify user identity, including anonymous authentication, Windows integrated (NTLM) authentication, Windows integrated (Kerberos) authentication, Basic (base64 encoded) authentication, Digest authentication, and authentication based on client certificates.

Windows Authentication is implemented in ASP.NET using the [WindowsAuthenticationModule](http://msdn.microsoft.com/en-us/library/vstudio/system.web.security.windowsauthenticationmodule(v=vs.100).aspx) module. The module constructs a [WindowsIdentity](http://msdn.microsoft.com/en-us/library/vstudio/system.security.principal.windowsidentity(v=vs.100).aspx) based on the credentials supplied by IIS and sets the identity as the current [User](http://msdn.microsoft.com/en-us/library/vstudio/system.web.httpcontext.user(v=vs.100).aspx) property value for the application.

Windows Authentication is the default authentication mechanism for ASP.NET applications and is identified as the authentication mode for an application using the [authentication](http://msdn.microsoft.com/en-us/library/vstudio/532aee0e(v=vs.100).aspx) configuration element, as shown in the following code example.

<system.web>

<authentication mode="Windows"/>

</system.web>

#### [Impersonating the Windows Identity](javascript:void(0))

Although the Windows Authentication mode sets the value of the current [User](http://msdn.microsoft.com/en-us/library/vstudio/system.web.httpcontext.user(v=vs.100).aspx) property to a [WindowsIdentity](http://msdn.microsoft.com/en-us/library/vstudio/system.security.principal.windowsidentity(v=vs.100).aspx) based on the credentials supplied by IIS, it does not modify the Windows identity that is supplied to the operating system. The Windows identity supplied to the operating system is used for permission checking, such as NTFS file permissions, or for connecting to a database using integrated security. By default, this Windows identity is the identity of the ASP.NET process. On Microsoft Windows 2000 and Windows XP Professional, this is the identity of the ASP.NET worker process, which is the local ASPNET account. On Windows Server 2003, this is the identity of the IIS Application Pool that the ASP.NET application is part of. By default, this is the NETWORK SERVICE account.

You can configure the Windows identity of your ASP.NET application as the Windows identity supplied by IIS by enabling impersonation. That is, you instruct your ASP.NET application to impersonate the identity supplied by IIS for all tasks that the Windows operating system authenticates, including file and network access.

To enable impersonation for your Web application, in the application's Web.config file set the **impersonate** attribute of the [identity](http://msdn.microsoft.com/en-us/library/vstudio/72wdk8cc(v=vs.100).aspx) element to **true**, as shown in the following code example.

<system.web>

<authentication mode="Windows"/>

<identity impersonate="true"/>

</system.web>

For more information on the ASP.NET process identity, see [Configuring ASP.NET Process Identity](http://msdn.microsoft.com/en-us/library/vstudio/dwc1xthy(v=vs.100).aspx). For more information on impersonation, see the [Impersonate](http://msdn.microsoft.com/en-us/library/vstudio/system.security.principal.windowsidentity.impersonate(v=vs.100).aspx) method.

#### [Enabling Authorization using NTFS ACLs](javascript:void(0))

You can improve the security of your ASP.NET application by securing the application's files using the NTFS file system and Access Control Lists (ACLs). ACLs enable you to specify which users and groups of users have access to your application's files. For a list of the minimum required NTFS file permissions that a Windows identity needs to run as the identity of an ASP.NET page, see [ASP.NET Required Access Control Lists (ACLs)](http://msdn.microsoft.com/en-us/library/vstudio/kwzs111e(v=vs.100).aspx).

|  |
| --- |
| **NoteNote** |
| You can also use ASP.NET roles to manage user authorization for pages and sections of your Web application. For more  information, see [Managing Authorization Using Roles](http://msdn.microsoft.com/en-us/library/vstudio/9ab2fxh0(v=vs.100).aspx). |

### Forms Authentication Provider

Forms authentication enables you to authenticate the user name and password of your users using a login form that you create. Unauthenticated requests are redirected to a login page, where the user provides credentials and submits the form. If the application authenticates the request, the system issues a ticket that contains a key for reestablishing the identity for subsequent requests.

The topics in this section describe how to use forms authentication to create a custom login system.

|  |
| --- |
| **NoteNote** |
| A convenient way to use forms authentication is to use ASP.NET membership (which stores user credentials) and the ASP.NET  login controls (which you can use to create a login page). |

#### ASP.NET Forms Authentication Overview

Forms authentication lets you authenticate users by using your own code and then maintain an authentication token in a cookie or in the page URL. Forms authentication participates in the ASP.NET page life cycle through the [FormsAuthenticationModule](http://msdn.microsoft.com/en-us/library/vstudio/system.web.security.formsauthenticationmodule(v=vs.100).aspx) class. You can access forms authentication information and capabilities through the [FormsAuthentication](http://msdn.microsoft.com/en-us/library/vstudio/system.web.security.formsauthentication(v=vs.100).aspx) class.

To use forms authentication, you create a login page that collects credentials from the user and that includes code to authenticate the credentials. Typically you configure the application to redirect requests to the login page when users try to access a protected resource, such as a page that requires authentication. If the user's credentials are valid, you can call methods of the [FormsAuthentication](http://msdn.microsoft.com/en-us/library/vstudio/system.web.security.formsauthentication(v=vs.100).aspx) class to redirect the request back to the originally requested resource with an appropriate authentication ticket (cookie). If you do not want the redirection, you can just get the forms authentication cookie or set it. On subsequent requests, the user's browser passes the authentication cookie with the request, which then bypasses the login page.

You configure forms authentication by using the [authentication](http://msdn.microsoft.com/en-us/library/vstudio/532aee0e(v=vs.100).aspx) configuration element. In the simplest case, you have a login page. In the configuration file, you specify a URL to redirect unauthenticated requests to the login page. You then define valid credentials, either in the Web.config file or in a separate file. The following example shows a section from a configuration file that specifies a login page and authentication credentials for the [Authenticate](http://msdn.microsoft.com/en-us/library/vstudio/system.web.security.formsauthentication.authenticate(v=vs.100).aspx) method. The passwords have been encrypted by using the [HashPasswordForStoringInConfigFile](http://msdn.microsoft.com/en-us/library/vstudio/system.web.security.formsauthentication.hashpasswordforstoringinconfigfile(v=vs.100).aspx) method.

<authentication mode="Forms">

<forms name="SavingsPlan" loginUrl="/Login.aspx">

<credentials passwordFormat="SHA1">

<user name="Kim"

password="07B7F3EE06F278DB966BE960E7CBBD103DF30CA6"/>

<user name="John"

password="BA56E5E0366D003E98EA1C7F04ABF8FCB3753889"/>

</credentials>

</forms>

</authentication>

After successful authentication, the [FormsAuthenticationModule](http://msdn.microsoft.com/en-us/library/vstudio/system.web.security.formsauthenticationmodule(v=vs.100).aspx) module sets the value of the [User](http://msdn.microsoft.com/en-us/library/vstudio/system.web.httpcontext.user(v=vs.100).aspx) property to a reference to the authenticated user. The following code example shows how to programmatically read the identity of the forms-authenticated user.

C#

String authUser2 = User.Identity.Name;

##### [Forms Authentication, ASP.NET Membership, and Login Controls](javascript:void(0))

A convenient way to work with forms authentication is to use ASP.NET membership and ASP.NET login controls. ASP.NET membership lets you store and manage user information and includes methods to authenticate users. ASP.NET login controls work with ASP.NET membership. They encapsulate the logic to prompt users for credentials, validate users, recover or replace passwords, and so on. In effect, ASP.NET membership and ASP.NET login controls provide a layer of abstraction over forms authentication. These features replace most or all the work that you would ordinarily have to do to use forms authentication. For more information, see [Managing Users by Using Membership](http://msdn.microsoft.com/en-us/library/vstudio/tw292whz(v=vs.100).aspx) and the [ASP.NET Login Controls Overview](http://msdn.microsoft.com/en-us/library/vstudio/ms178329(v=vs.100).aspx).

##### [Forms Authentication and the Authentication Service](javascript:void(0))

You can also access forms authentication as a Windows Communication Framework (WCF) service by using the ASP.NET authentication service. The authentication service enables you to use forms authentication from any application that can send and consume messages in SOAP format. The authentication service accepts user credentials and returns a forms authentication cookie.

For example, you can log in users from an application that was not developed with the .NET Framework. For more information, see [Windows Communication Foundation Authentication Service Overview](http://msdn.microsoft.com/en-us/library/vstudio/bb386582(v=vs.100).aspx).

#### How to: Implement Simple Forms Authentication

#### Forms Authentication Control Flow

#### Forms Authentication Credentials

#### Forms Authentication Utilities

#### Handling Forms Authentication Events

#### Forms Authentication Across Applications

## ASP.NET Authorization

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
| F5 |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  | |
|  |  | |
|  |  | |
|  | |  |
|  | |  |
|  | |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  | Solo demostrativo de donde están las definiciones |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Investigar como aparece las opciones de namespace  [ctrl-espacio] |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  | En gestionUsuarios.aspx |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

<http://www.youtube.com/watch?v=4s2vCFaxJeo>

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |