Securing Membership

.NET Framework 4 Other Versions

ASP.NET membership provides functionality for managing and authenticating users and is most commonly used with Forms authentication and authentication controls such as the Login, LoginView, LoginStatus, LoginName, PasswordRecovery, and CreateUserWizard controls. This topic describes how to optimize the security of the membership feature through best practices in configuring a Web site and coding with the membership classes.

While following these best practices can improve the security of your application, it is also important that you continually keep your application server up to date with the latest security patches for Microsoft Windows and Internet Information Services (IIS), as well as any patches for Microsoft SQL Server or other membership data sources.

For more detailed information about best practices for writing secure code and securing applications, see the book "*Writing Secure Code*" by Michael Howard and David LeBlanc and the guidance provided by Microsoft Patterns and Practices (http://www.microsoft.com/resources/practices/default.mspx).

Secure Membership Configuration

The membership feature is enabled by default for ASP.NET applications and cannot be disabled. The default configuration settings are set to the most secure values. For information about membership configuration settings and their default values, see membership Element (ASP.NET Settings Schema). You should set the **requiresQuestionAndAnswer** attribute to **true**, especially where **enablePasswordReset** or **enablePasswordRetrieval** is likewise **true**.

Securing Configuration Values

When storing sensitive information in a configuration file for an application, you should encrypt the sensitive values using Protected Configuration. Information that is especially sensitive includes the encryption keys stored in the machineKey configuration element and connection strings to a data source stored in the connectionStrings configuration element. For more information, see Encrypting Configuration Information Using Protected Configuration.

Secure Encryption Keys and Hashing

It is highly recommended that you encrypt user passwords in the membership data source using a **passwordFormat** attribute set to **Hashed** or **Encrypted**, where **Hashed** is the most secure format. The encryption key values for the specified encryption algorithm are stored in the machineKey configuration element. For strong encryption, specify an encryption key that is a randomly generated value of the appropriate length for the selected encryption algorithm.

On a server that hosts multiple applications, it is recommended that you define unique encryption keys for each application. A less secure alternative is to define a single encryption key and specify the **IsolateApps** option with the key.

You can set the machine configuration on a host server to deny applications from overriding configuration settings. This includes denying the ability for encryption keys to be redefined in the Web.config file for an application.

Securing Connections to a Membership Data Source

Connection Strings

To keep the connection to your database server secure, you should encrypt connection-string information in the configuration using Protected Configuration. For more information, see Encrypting Configuration Information Using Protected Configuration.

Connecting to SQL Server using Integrated Security

You should connect to computers running SQL Server using Integrated Security to avoid the possibility of your connection string being compromised and your user ID

and password being exposed. When you specify a connection that uses Integrated Security to connect to a computer running SQL Server, the membership feature reverts to the identity of the process. You should ensure that the identity of the process running ASP.NET (for example, the application pool) is the default process account or a restricted user account. For more information, see ASP.NET Impersonation.

SQL Server Database Permissions

The SQL Server database that is used to store the user information for membership includes database roles and views that enable you to restrict user access to only the required capabilities and visibility. You should assign the minimum necessary privileges to a user ID that connects to the SQL Server membership database. For more information, see Roles and Views in the Application Services Database for SQL Server.

SQL Server Express Worker Process Identity

SQL Server Express 2005 includes a new mode of operation where it can start a worker process running as the identity of the connecting user. This capability is referred to as "run as user" mode. Although this mode of operation is suitable for desktop development when using IIS, starting worker processes is not appropriate on Web servers hosting multiple, untrusted customer codebases. Shared hosting servers that contain applications that do no trust each other should explicitly disable the "run as user" functionality. This functionality can be turned off by connecting to the SQL Express instance (for example, `osql -E -S .\sqlexpress`) and issuing the following Transact-SQL command.

```
EXEC sp_configure 'show advanced option', '1'

GO

RECONFIGURE WITH OVERRIDE

GO

EXEC sp_configure 'user instances enabled', 0

GO

RECONFIGURE WITH OVERRIDE

GO
```

Secure Web Pages that Use Membership

Application pages that work with sensitive data, such as logon pages, should be secured using standard Web-security mechanisms. These include measures such as using Secure Sockets Layer (SSL) and requiring that users be logged on to carry out sensitive operations like updating user information or deleting users.

Additionally, pages should not expose sensitive feature data such as passwords, and in some cases user names, in clear text. Ensure that pages that display such information make use of SSL and are available only to authenticated users. Also, avoid storing sensitive feature data in cookies or sending it across insecure connections.

Securing Against Denial of Service Attacks

Methods that perform updates or large search operations can reduce the responsiveness of your membership data source if called concurrently by a number of clients. To reduce exposure to a denial of service attack, restrict access to ASP.NET pages that use methods that perform database updates or searches to administrative users, and expose only ASP.NET pages that provide validation and password management for general use.

Error Messages and Events

Exceptions

To prevent sensitive information from being exposed to unwanted sources, configure your application either to not display detailed error messages or to display detailed error messages only when the client is the Web server itself. For more information, see customErrors Element (ASP.NET Settings Schema).

Event Log

If your server is running Windows Server 2003, you can improve the security of your application by securing the event log, and by setting parameters regarding the size, retention, and so on of the event log to prevent an indirect denial of service attack against it.

Health Monitoring

Successful and failed logon attempts are logged using the ASP.NET health-monitoring feature. In the default configuration, this means failed login attempts will record user-name and other diagnostic information in the **Application** event log. Ensure that access to the event log is restricted to keep this information private.

Custom Membership Providers

When creating a custom membership provider, ensure that you follow security best practices to avoid attacks such as SQL-injection attacks when working with a database. When making use of a custom membership provider, ensure that the provider has been reviewed for security best practices.

Working with Culture-Sensitive Characters

When using the SQL Server membership provider or a custom membership provider, your data source might be configured to store membership data in a culture sensitive format. However, ASP.NET evaluates user names from the authorization configuration element and user names from the membership data store as culture invariant. As a result, an unauthorized user could be granted authorization because when the user name is treated as culture invariant, it is the same as the name of an authorized user.

To avoid users gaining unauthorized access, ensure that user names are unique when evaluated as culture invariant. Alternatively, you can specify only role names for authorization using the authorization configuration element and then ensure that role names are unique when evaluated as culture invariant. Specifying authorization using role names is often preferred, because creating and managing roles can be restricted as an administrative function.

See Also

Other Resources
Managing Users by Using Membership

Did you find this helpful? Yes No

Community Additions ADD

Dev centers
- Windows
- Windows Phone
- Office
- Windows Azure
- Visual Studio

More...

Learning resources
- Microsoft Virtual Academy
- Channel 9
- Interoperability Bridges
- MSDN Magazine

Community
- Forums
- Blogs
- Codeplex

Support
- Self support
- Other support options

Programs
- BizSpark (for startups)
- DreamSpark
- Faculty Connection
- Microsoft Student

Did you find this helpful? Yes No