Authentication in SQL Server

03/30/20173 minutes to readContributors all

In this article

SQL Server supports two authentication modes, Windows authentication mode and mixed mode.

- Windows authentication is the default, and is often referred to as integrated security because this SQL Server security model is tightly integrated with Windows. Specific Windows user and group accounts are trusted to log in to SQL Server. Windows users who have already been authenticated do not have to present additional credentials.
- Mixed mode supports authentication both by Windows and by SQL Server. User name and password pairs are maintained within SQL Server.

Important

We recommend using Windows authentication wherever possible. Windows authentication uses a series of encrypted messages to authenticate users in SQL Server. When SQL Server logins are used, SQL Server login names and passwords are passed across the network, which makes them less secure.

With Windows authentication, users are already logged onto Windows and do not have to log on separately to SQL Server. The following `SqlConnection.ConnectionString` specifies Windows authentication without requiring the a user name or password.

Copy

```
"Server=MSSQL1;Database=AdventureWorks;Integrated Security=true;
```

Note

Logins are distinct from database users. You must map logins or Windows groups to database users or roles in a separate operation. You then grant permissions to users or roles to access database objects.

Authentication Scenarios

Windows authentication is usually the best choice in the following situations:

- There is a domain controller.
- The application and the database are on the same computer.
- You are using an instance of SQL Server Express or LocalDB.

SQL Server logins are often used in the following situations:

- If you have a workgroup.
- Users connect from different, non-trusted domains.
- Internet applications, such as ASP.NET.

Note

Specifying Windows authentication does not disable SQL Server logins. Use the ALTER LOGIN DISABLE Transact-SQL statement to disable highly-privileged SQL Server logins.

Login Types

SQL Server supports three types of logins:

- A local Windows user account or trusted domain account. SQL Server relies on Windows to authenticate the Windows user accounts.
- Windows group. Granting access to a Windows group grants access to all Windows user logins that are members of the group.
- SQL Server login. SQL Server stores both the username and a hash of the password in the master database, by using internal authentication methods to verify login attempts.

Note

SQL Server provides logins created from certificates or asymmetric keys that are used only for code signing. They cannot be used to connect to SQL Server.

Mixed Mode Authentication

If you must use mixed mode authentication, you must create SQL Server logins, which are stored in SQL Server. You then have to supply the SQL Server user name and password at run time.

Important

SQL Server installs with a SQL Server login named `sa` (an abbreviation of "system administrator"). Assign a strong password to the `sa` login and do not use the `sa` login in your application. The `sa` login maps to the `sysadmin` fixed server role, which has irrevocable administrative credentials on the whole server. There are no limits to the potential damage if an attacker gains access as a system administrator. All members of the Windows `BUILTIN\Administrators` group (the local administrator's group) are members of the `sysadmin` role by default, but can be removed from that role.

SQL Server provides Windows password policy mechanisms for SQL Server logins when it is running on Windows Server 2003 or later versions. Password complexity policies are designed to deter brute force attacks by increasing the number of possible passwords. SQL Server can apply the same complexity and expiration policies used in Windows Server 2003 to passwords used inside SQL Server.

Important

Concatenating connection strings from user input can leave you vulnerable to a connection string injection attack. Use the **SqlConnectionStringBuilder** to create syntactically valid connection strings at run time. For more information, see **Connection String Builders**.

External Resources

For more information, see the following resources.

Resource	Description
Principals in SQL Server Books Online	Describes logins and other security principals in SQL Server.

See Also

Securing ADO.NET Applications
Application Security Scenarios in SQL Server
Connecting to a Data Source
Connection Strings
ADO.NET Managed Providers and DataSet Developer Center