Application Security Scenarios in SQL Server

03/30/20173 minutes to readContributors all

In this article

There is no single correct way to create a secure SQL Server client application. Every application is unique in its requirements, deployment environment, and user population. An application that is reasonably secure when it is initially deployed can become less secure over time. It is impossible to predict with any accuracy what threats may emerge in the future.

SQL Server, as a product, has evolved over many versions to incorporate the latest security features that enable developers to create secure database applications. However, security doesn't come in the box; it requires continual monitoring and updating.

Common Threats

Developers need to understand security threats, the tools provided to counter them, and how to avoid self-inflicted security holes. Security can best be thought of as a chain, where a break in any one link compromises the strength of the whole. The following list includes some common security threats that are discussed in more detail in the topics in this section.

SQL Injection

SQL Injection is the process by which a malicious user enters Transact-SQL statements instead of valid input. If the input is passed directly to the server without being validated and if the application inadvertently executes the injected code, then the attack has the potential to damage or destroy data. You can thwart SQL Server injection attacks by using stored procedures and parameterized commands, avoiding dynamic SQL, and restricting permissions on all users.

Elevation of Privilege

Elevation of privilege attacks occur when a user is able to assume the privileges of a trusted account, such as an owner or administrator. Always run under least-privileged user accounts and assign only needed permissions. Avoid using administrative or owner accounts for executing code. This limits the amount of damage that can occur if an attack succeeds. When performing tasks that require additional permissions, use procedure signing or impersonation only for the duration of

the task. You can sign stored procedures with certificates or use impersonation to temporarily assign permissions.

Probing and Intelligent Observation

A probing attack can use error messages generated by an application to search for security vulnerabilities. Implement error handling in all procedural code to prevent SQL Server error information from being returned to the end user.

Authentication

A connection string injection attack can occur when using SQL Server logins if a connection string based on user input is constructed at run time. If the connection string is not checked for valid keyword pairs, an attacker can insert extra characters, potentially accessing sensitive data or other resources on the server. Use Windows authentication wherever possible. If you must use SQL Server logins, use the SqlConnectionStringBuilder to create and validate connection strings at run time.

Passwords

Many attacks succeed because an intruder was able to obtain or guess a password for a privileged user. Passwords are your first line of defense against intruders, so setting strong passwords is essential to the security of your system. Create and enforce password policies for mixed mode authentication.

Always assign a strong password to the `sa` account, even when using Windows Authentication.

In This Section

Managing Permissions with Stored Procedures in SQL Server
Describes how to use stored procedures to manage permissions and control data access. Using stored procedures is an effective way to respond to many security threats.

Writing Secure Dynamic SQL in SQL Server
Describes techniques for writing secure dynamic SQL using stored procedures.

Signing Stored Procedures in SQL Server
Describes how to sign a stored procedure with a certificate to enable users to work with data they do not have direct access to. This enables stored procedures to perform operations that the caller does not have permissions to perform directly.

Customizing Permissions with Impersonation in SQL Server
Describes how to use the EXECUTE AS clause to impersonate another user. Impersonation switches the execution context from the caller to the specified user.

Granting Row-Level Permissions in SQL Server
Describes how to implement row-level permissions to restrict data access.

Creating Application Roles in SQL Server
Describes features and functionality of application roles.

Enabling Cross-Database Access in SQL Server
Describes how to enable cross-database access without jeopardizing security.

See Also

SQL Server Security
Overview of SQL Server Security
Securing ADO.NET Applications
ADO.NET Managed Providers and DataSet Developer Center