Summary

This How To shows a number of ways to help protect your ASP.NET application from SQL injection attacks. SQL injection can occur when an application uses input to construct dynamic SQL statements or when it uses stored procedures to connect to the database. Conventional security measures, such as the use of SSL and IPSec, do not protect your application from SQL injection attacks. Successful SQL injection attacks enable malicious users to execute commands in an application's database.

Countermeasures include using a list of acceptable characters to constrain input, using parameterized SQL for data access, and using a least privileged account that has restricted permissions in the database. Using stored procedures with parameterized SQL is the recommended approach because SQL parameters are type safe. Type-safe SQL parameters can also be used with dynamic SQL. In situations where parameterized SQL cannot be used, consider using character escaping techniques.

Contents

[Objectives](https://msdn.microsoft.com/en-us/library/ff648339.aspx#paght000002_objectives)  
[Overview](https://msdn.microsoft.com/en-us/library/ff648339.aspx#paght000002_overview)  
[Summary of Steps](https://msdn.microsoft.com/en-us/library/ff648339.aspx#paght000002_summaryofsteps)  
[Step 1. Constrain Input](https://msdn.microsoft.com/en-us/library/ff648339.aspx#paght000002_step1)  
[Step 2. Use Parameters with Stored Procedures](https://msdn.microsoft.com/en-us/library/ff648339.aspx#paght000002_step2)  
[Step 3. Use Parameters with Dynamic SQL](https://msdn.microsoft.com/en-us/library/ff648339.aspx#paght000002_step3)

MVC

LINQ and Entity Framework already check for SQL Injection for you.

But you should read the documentation anyhow:

LINQ [MSDN Link](http://msdn.microsoft.com/en-us/library/bb386929.aspx) (section SQL-Injection Attacks)

Entity Framework [MSDN Link](http://msdn.microsoft.com/en-us/library/cc716760.aspx) (section Security Considerations for Queries)

## SQL-Injection Attacks

Q. How is LINQ to SQL protected from SQL-injection attacks?

A. SQL injection has been a significant risk for traditional SQL queries formed by concatenating user input. LINQ to SQL avoids such injection by using [SqlParameter](https://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlparameter.aspx) in queries. User input is turned into parameter values. This approach prevents malicious commands from being used from customer input.

Entity Framework

Use connection string builders when dynamically creating connections.

If you must construct connection strings at runtime, use the [EntityConnectionStringBuilder](https://msdn.microsoft.com/en-us/library/system.data.entityclient.entityconnectionstringbuilder.aspx) class. This string builder class helps prevent connection string injection attacks by validating and escaping invalid input information. For more information, see [How to: Build an EntityConnection Connection String](https://msdn.microsoft.com/en-us/library/bb738533.aspx). Also use the appropriate string builder class to construct the data source connection string that is part of the Entity Framework connection string. For information about connection string builders for ADO.NET providers, see [Connection String Builders](https://msdn.microsoft.com/en-us/library/ms254947.aspx).

### Prevent SQL injection attacks.

Applications frequently take external input (from a user or another external agent) and perform actions based on that input. Any input that is directly or indirectly derived from the user or an external agent might have content that uses the syntax of the target language in order to perform unauthorized actions. When the target language is a Structured Query Language (SQL), such as Transact-SQL, this manipulation is known as a SQL injection attack. A malicious user can inject commands directly into the query and drop a database table, cause a denial of service, or otherwise change the nature of the operation being performed.

* Entity SQL injection attacks:

SQL injection attacks can be performed in Entity SQL by supplying malicious input to values that are used in a query predicate and in parameter names. To avoid the risk of SQL injection, you should never combine user input with Entity SQL command text.

Entity SQL queries accept parameters everywhere that literals are accepted. You should use parameterized queries instead of injecting literals from an external agent directly into the query. You should also consider using query builder methods to safely construct [Entity SQL](https://msdn.microsoft.com/en-us/library/bb896238.aspx).

* LINQ to Entities injection attacks:

Although query composition is possible in LINQ to Entities, it is performed through the object model API. Unlike Entity SQL queries, LINQ to Entities queries are not composed by using string manipulation or concatenation, and they are not susceptible to traditional SQL injection attacks.