



De la Seguridad del acceso a datos en aplicaciones ASP .NET

(Nota de Willy, Revisar toda la guie en Castellano en <http://www.WillyDev.net/Seguridad>)

J.D. Meier, Alex Mackman, Michael Dunner y Srinath Vasireddy
Microsoft Corporation

Resumen

Este capítulo incluye recomendaciones y orientación que le ayudarán a desarrollar una estrategia de acceso seguro a datos. Los temas que se tratan son la autenticación de Windows desde ASP.NET a la base de datos, la creación de cadenas de conexión seguras, el almacenamiento de credenciales en una base de datos de forma segura, la protección frente a ataques de inyección SQL y el uso de funciones de base de datos.

Contenido

Introducción a la seguridad del acceso a datos
Autenticación
Autorización
Comunicación segura
Conectar con privilegios mínimos
Crear una cuenta de base de datos con privilegios mínimos
Almacenar cadenas de conexión a bases de datos de forma segura
Autenticar usuarios en una base de datos
Ataques de inyección SQL
Auditoría
Identidad del proceso para SQL Server
Resumen

Al generar aplicaciones basadas en Web es fundamental realizar el acceso y almacenamiento de datos de forma segura. En este capítulo se tratan algunos de los principales problemas del acceso a datos. Le ayudará a:

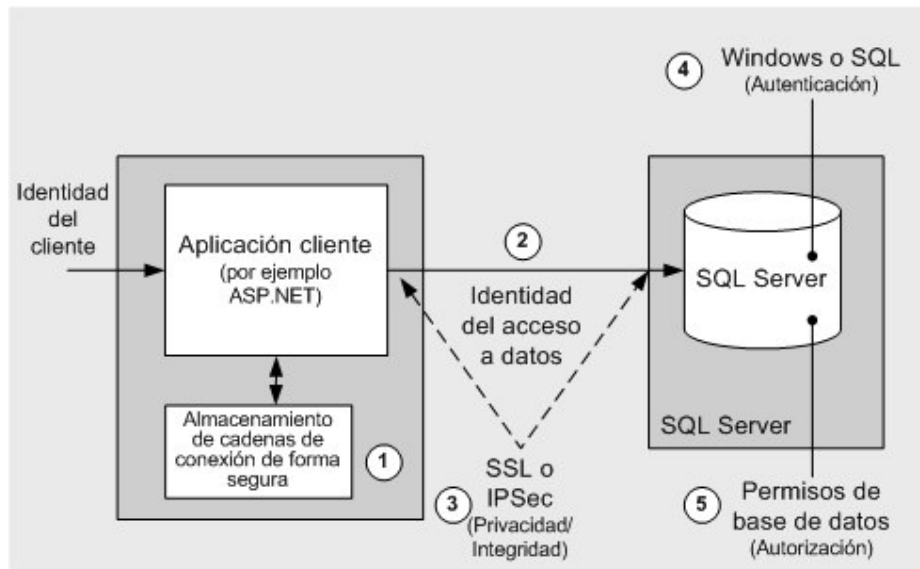
- Elegir entre la autenticación del sistema operativo Microsoft® Windows® y la autenticación de SQL al conectarse a SQL Server™.

- Almacenar cadenas de conexión de forma segura.
- Decidir si se va a transmitir el contexto de seguridad del llamador original a la base de datos.
- Aprovechar la agrupación de la conexión.
- Protegerse frente a los ataques de inyección SQL.
- Almacenar credenciales de forma segura en una base de datos.

En el capítulo también se presentan los diversos compromisos relacionados con el uso de funciones; por ejemplo, las funciones de la base de datos frente a la lógica de funciones aplicada en el nivel medio. Por último se incluye un conjunto de recomendaciones básicas para el acceso a datos.

Introducción a la seguridad del acceso a datos

La ilustración 12.1 muestra los principales problemas de seguridad asociados al acceso a datos.



{Insert figure: CH12 - Data Access Security Overview.gif}

Ilustración 12.1

Principales problemas de seguridad del acceso a datos

A continuación se resumen los principales problemas que se muestran en la ilustración 12.1 y que se tratan en el resto del capítulo:

1. **Almacenar cadenas de conexión a bases de datos de forma segura.** Es especialmente importante si la aplicación utiliza la autenticación de SQL para conectarse a SQL Server o si se conecta a bases de datos que no son de Microsoft que requieren credenciales explícitas para iniciar la sesión. En estos casos, las cadenas de conexión contienen nombres de usuario y contraseñas de texto sin cifrar.
2. **Usar una identidad o unas identidades apropiadas para tener acceso a la base de datos.** El acceso a datos puede realizarse utilizando la identidad del proceso de llamada, una o varias identidades de servicio o la identidad del llamador original (con suplantación/delegación). La elección depende del modelo del acceso a datos: subsistema de confianza o suplantación/delegación.

3. **Asegurar datos que se extienden en la red.** Por ejemplo, asegurar credenciales de inicio de sesión y datos confidenciales intercambiados con SQL Server.

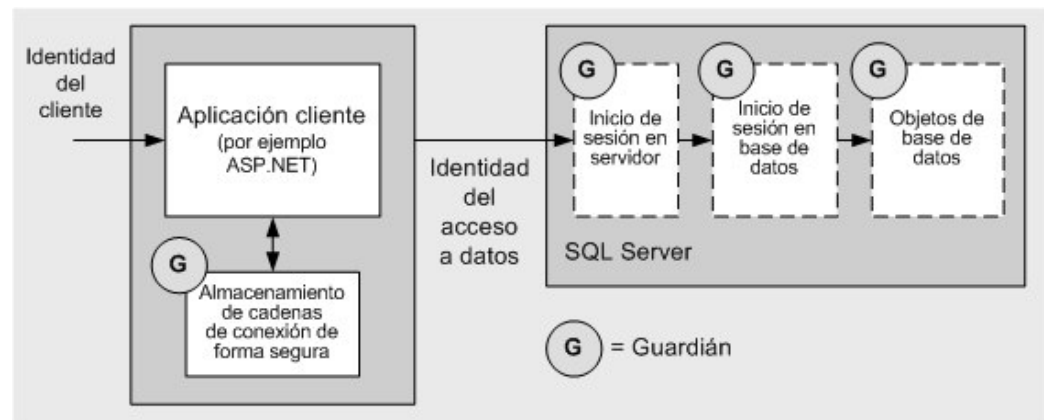
Nota: Las credenciales de inicio de sesión sólo se exponen en la red si se utiliza la autenticación de SQL y no la autenticación de Windows.

SQL Server 2000 admite SSL, con certificados de servidor. También se puede utilizar IPSec para cifrar el tráfico entre el equipo cliente (por ejemplo, un servidor Web o de aplicaciones) y un servidor de base de datos.

4. **Autenticar llamadores en la base de datos.** SQL Server admite la autenticación de Windows (con NTLM o Kerberos) y la autenticación de SQL (con el mecanismo de autenticación integrado de SQL Server).
5. **Autorizar llamadores en la base de datos.** Los permisos se asocian a objetos de base de datos individuales. Pueden asociarse a usuarios, grupos o funciones.

Equipos selectores de SQL Server

En la ilustración 12.2 se destacan los principales equipos selectores para el acceso a datos de un servidor SQL Server.



{Insert figure: CH12 - Data Access GateKeepers.gif}

Ilustración 12.2

Equipos selectores de SQL Server

Los principales equipos selectores son:

- El almacén de datos elegido que se utiliza para mantener la cadena de conexión a bases de datos.
- El inicio de sesión de SQL Server (según lo establecido por el nombre de servidor especificado en la cadena de conexión).
- El inicio de sesión de la base de datos (según lo establecido por el nombre de la base de datos especificado en la cadena de conexión).
- Los permisos asociados a cada uno de los objetos de la base de datos. Pueden asignarse a usuarios, grupos o funciones.

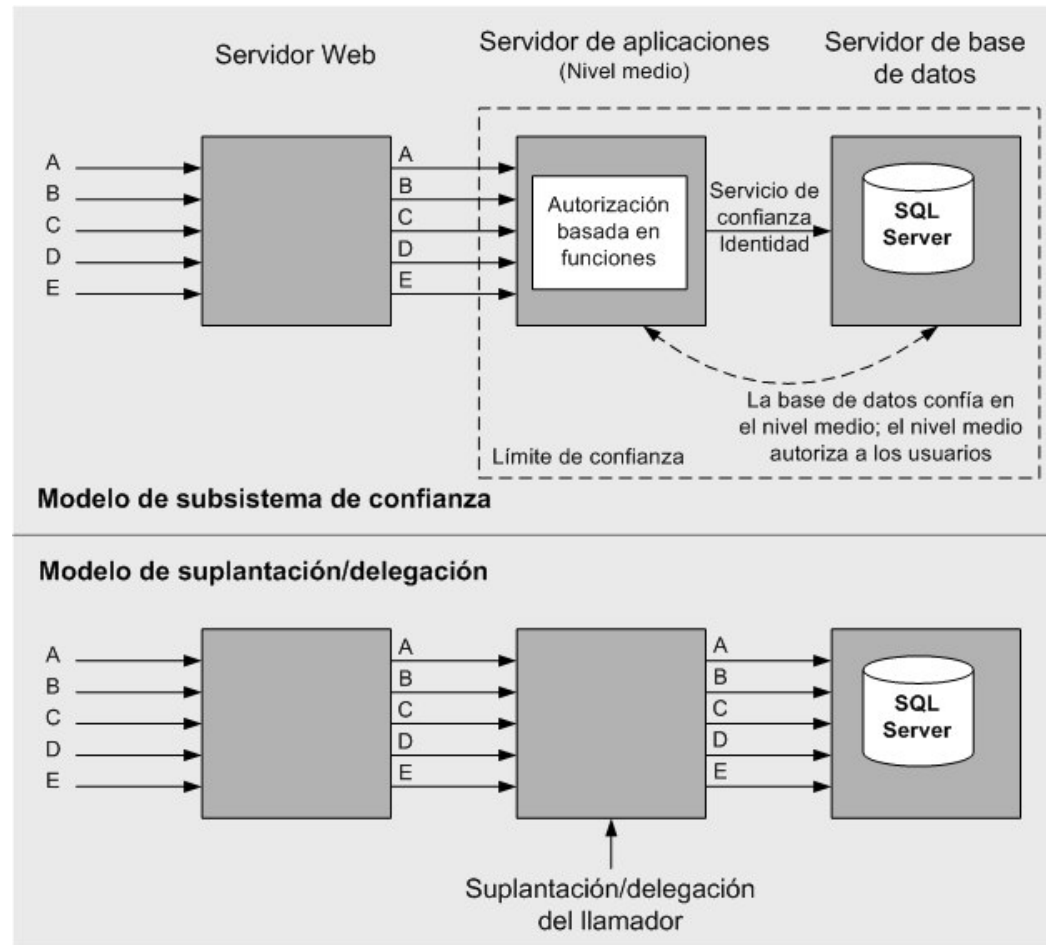
Subsistema de confianza frente a suplantación/delegación

La granularidad del acceso a la base de datos es uno de los principales factores que hay que tener en cuenta. Hay que tener en cuenta si se necesita autorización de usuario en la base de datos (lo que requiere el modelo de suplantación/delegación) o

si se puede utilizar la lógica de función de aplicación en el nivel medio de la aplicación para autorizar usuarios (lo que requiere el modelo de subsistema de confianza).

Si la base de datos requiere autorización de usuario, es necesario suplantar al llamador original. Si bien se admite el modelo de suplantación/delegación, se recomienda utilizar el modelo de subsistema de confianza, en el que el llamador original se comprueba en la puerta IIS/ASP.NET, se asigna a una función y, a continuación, se autoriza según la pertenencia a una función. A continuación, los recursos del sistema para la aplicación se autorizan en la aplicación o en la función mediante cuentas de servicio o con la identidad del proceso de la aplicación (por ejemplo, la cuenta ASPNET).

En la ilustración 12.3 se muestran ambos modelos.



{Insert figure: CH12 - Trusted Sub-system vs. Impersonation-Delegation Models.gif}

Ilustración 12.3

Modelos de subsistema de confianza y de suplantación/delegación para tener acceso a la base de datos

Hay varios factores fundamentales que hay que tener en cuenta cuando se conecta a SQL Server para tener acceso a datos. Dichos factores se resumen a continuación y se explican en las secciones siguientes:

- **¿Qué tipo de autenticación se debe utilizar?** La autenticación de Windows ofrece una seguridad mejorada, pero los servidores de seguridad y los problemas de dominios no confiables pueden obligarle a utilizar la autenticación de SQL. Si es así, debe asegurarse de que el uso de la

autenticación de SQL en la aplicación es lo más segura posible, tal y como se explica más adelante en la sección "Autenticación de SQL" de este capítulo.

- **Función de usuario único frente a funciones de varios usuarios.**
¿Necesita la aplicación tener acceso a SQL mediante una única cuenta con un conjunto fijo de permisos en la base de datos o se requieren varias cuentas (basadas en funciones) según el usuario de la aplicación?
- **Identidad del llamador.** ¿Necesita la base de datos recibir la identidad del llamador original a través del contexto de llamada para llevar a cabo la autorización o auditoría, o se puede utilizar una o varias conexiones de confianza y pasar la identidad del llamador original en el nivel de la aplicación?
Para que el sistema operativo transmita la identidad del llamador original, necesita suplantación/delegación en el nivel medio. Esto reduce considerablemente la eficacia de la agrupación de la conexión. La agrupación de la conexión sigue habilitada pero produce numerosos grupos pequeños (para cada uno de los contextos de seguridad) en los que la reutilización de las conexiones es escasa o nula.
- **¿Se intercambian datos confidenciales con el servidor de base de datos?**
Aunque en la autenticación de Windows no se extienden las credenciales de usuario al servidor de base de datos a través de la red, si los datos de la aplicación son confidenciales (por ejemplo, detalles de los empleados o datos de nóminas), deberían asegurarse mediante IPsec o SSL.

Autenticación

En esta sección se trata la forma en que debe autenticarse a los clientes en SQL Server y el modo en que debe elegirse una identidad para tener acceso a la base de datos en aplicaciones cliente, antes de conectarse a SQL Server.

Autenticación de Windows

La autenticación de Windows es más segura que la autenticación de SQL por los siguientes motivos:

- Las credenciales se administran automáticamente y no se transmiten a través de la red.
- Se evita tener que incrustar nombres de usuario y contraseñas en cadenas de conexión.
- La seguridad de inicio de sesión se mejora mediante los periodos de caducidad y la longitud mínima de las contraseñas, así como los bloqueos de cuenta después de varias solicitudes de inicio de sesión no válidas. De esta forma se reduce la amenaza de los ataques de diccionario.

La autenticación de Windows se utiliza en los siguientes escenarios:

- Ha utilizado el modelo de subsistema de confianza y se conecta a SQL Server mediante una identidad fija única. Si se conecta desde ASP.NET, se supone que la aplicación Web no está configurada para la suplantación.
En este escenario, utilice la identidad del proceso ASP.NET o una identidad de componente revisado (obtenida de la cuenta que se utilizó para ejecutar una aplicación del servidor de Servicios Empresariales).
- Delega, de forma intencionada, el contexto de seguridad del llamador original mediante el uso de la delegación (y se prepara para sacrificar la escalabilidad de la aplicación anteponiendo la agrupación de la conexión a la base de datos).

Cuando utilice la autenticación de Windows para conectarse a SQL Server, tenga en cuenta los siguientes puntos importantes:

- Utilice el principio de privilegio mínimo en la cuenta del proceso ASP.NET. Evite conceder a la cuenta del proceso ASP.NET el privilegio "Actuar como parte del sistema operativo" para habilitar las llamadas a la API **LogonUser**.
- Determine cuál es el código que requiere privilegios adicionales e inclúyalo en componentes revisados, que se ejecutan en aplicaciones de Servicios Empresariales fuera del proceso.

Más información

Para obtener más información acerca del acceso a los recursos de la red desde ASP.NET y de cómo elegir y configurar una cuenta adecuada para ejecutar ASP.NET, consulte el capítulo 8, "[Seguridad de ASP.NET](#)."

Utilizar la autenticación de Windows

Cuando utiliza la autenticación de Windows para conectarse a SQL Server desde una aplicación ASP.NET (o servicio Web o componente remoto alojado por ASP.NET), dispone de las siguientes opciones:

- Utilizar la identidad del proceso ASP.NET.
- Utilizar identidades fijas en ASP.NET.
- Utilizar componentes revisados.
- Utilizar la API **LogonUser** y suplantar una identidad específica.
- Utilizar la identidad del llamador original.
- Utilizar la cuenta de usuario de Internet anónimo.

Recomendación

Se recomienda configurar la identidad del proceso ASP.NET local cambiando la contraseña por un valor conocido en el servidor Web y crear una cuenta reflejada en el servidor de base de datos mediante la creación de un usuario local con el mismo nombre y contraseña. A continuación se ofrecen más detalles acerca de las diversas opciones.

Utilizar la identidad del proceso ASP.NET

Si se conecta a SQL Server directamente desde una aplicación ASP.NET (o servicio Web o componente remoto alojado por ASP.NET), utilice la identidad del proceso ASP.NET. Se realiza con frecuencia y la aplicación define el límite de confianza, es decir, que la base de datos confía a la cuenta ASP.NET el acceso a objetos de la base de datos.

Tiene tres opciones:

- Utilizar cuentas locales ASPNET reflejadas.
- Utilizar cuentas locales personalizadas reflejadas.
- Utilizar una cuenta de dominio personalizada.

Utilizar cuentas locales ASPNET reflejadas

Es la opción más simple y la que se suele utilizar cuando se es propietario de la base de datos de destino (y se puede controlar la administración de las cuentas locales del servidor de base de datos). Con esta opción se utiliza la cuenta local ASPNET con privilegios mínimos para ejecutar ASP.NET y crear a continuación una cuenta duplicada en el servidor de base de datos.

Nota: Esta opción tiene la ventaja añadida de que funciona en dominios no confiables y a través de servidores de seguridad. Es posible que el servidor de seguridad no abra puertos suficientes para admitir la autenticación de Windows.

Utilizar cuentas locales personalizadas reflejadas

Esta opción es igual que la anterior excepto en que no se utiliza la cuenta ASPNET personalizada. Lo que significa dos cosas:

- Será necesario crear una cuenta local personalizada con los permisos y privilegios adecuados.
Para obtener más información, consulte "[Cómo crear una cuenta personalizada para ejecutar ASP.NET](#)" en la sección Referencia de esta guía.
- Ya no utiliza la cuenta personalizada creada por el proceso de instalación de .NET Framework. Puede ser que en su empresa se siga la política de no utilizar cuentas de instalación personalizadas. Esto podría aumentar la seguridad de la aplicación.
Para obtener más información, consulte la página de Sans Top 20, "G2—Accounts with No Passwords or Weak Passwords" (<http://www.sans.org/top20.htm>) (en inglés).

Utilizar una cuenta de dominio personalizada

Esta opción es similar a la anterior, excepto en que se utiliza una cuenta de dominio con privilegios mínimos en lugar de una cuenta local. Se supone que los equipos cliente y servidor están en los mismos dominios de confianza. La principal ventaja es que los equipos no comparten las credenciales, sino que únicamente proporcionan acceso a la cuenta de dominio. Además, la administración es más sencilla con cuentas de dominio.

Implementar la identidad del proceso ASPNET reflejada

Para utilizar cuentas reflejadas para conectarse desde ASP.NET a una base de datos, es necesario realizar las acciones siguientes:

- Utilice el Administrador de usuarios en el servidor Web para restablecer la contraseña de la cuenta ASPNET a un valor de contraseña rigurosa conocido.

Importante: Si cambia la contraseña de ASPNET por un valor conocido, la contraseña de la Autoridad de seguridad local (LSA) del equipo local ya no coincidirá con la contraseña de la cuenta almacenada en la base de datos del Administrador de cuentas de seguridad (SAM) de Windows. Si necesita recuperar el valor predeterminado AutoGenerate, debe realizar lo siguiente:

Ejecute `Aspnet_regiis.exe` para restablecer la configuración predeterminada de ASP.NET. Para obtener más información, consulte el artículo Q306005, "[HOWTO: Repair IIS Mapping After You Remove and Reinstall IIS](#)" (en inglés), en Microsoft Knowledge Base. Cuando lo haya hecho, obtendrá una nueva cuenta y un nuevo identificador de seguridad (SID) de Windows. Los permisos de esta cuenta tienen establecidos valores predeterminados. Como resultado, es necesario volver a aplicar de forma explícita los permisos y privilegios que había establecido originalmente en la antigua cuenta ASPNET.

- Establezca de forma explícita la contraseña en el archivo `Machine.config`.

```
<processModel userName="machine" password="YourStrongPassword" .
```

Debería proteger el archivo `Machine.config` de accesos no autorizados mediante las ACL de Windows. Por ejemplo, restrinja `Machine.config` en la cuenta de usuario de Internet anónimo de IIS.

- Cree una cuenta reflejada (con el mismo nombre y contraseña) en el servidor de base de datos.
- En la base de datos SQL, cree un inicio de sesión en el servidor para la cuenta ASPNET local y, a continuación, asigne el inicio de sesión a una cuenta de usuario de la base de datos requerida. Cree a continuación una función de

usuario de la base de datos, agregue el usuario de la base de datos a la función y configure en la función los permisos de base de datos adecuados. Para obtener más información, consulte "Crear una cuenta de base de datos con privilegios mínimos" más adelante en este capítulo.

Conectar a SQL Server mediante la autenticación de Windows

Para conectar a SQL Server mediante la autenticación de Windows:

- En la aplicación cliente, utilice una cadena de conexión que contenga "Trusted Connection=Yes" o "Integrated Security=SSPI". Ambas cadenas son equivalentes y ambas obtienen como resultado la autenticación de Windows (suponiendo que SQL Server esté configurado para la autenticación de Windows). Por ejemplo:

```
"server=MySQL; Integrated Security=SSPI; database=Northwind"
```

Nota: La identidad del cliente que realiza la solicitud (es decir, el cliente autenticado por SQL Server) viene determinada por el testigo de suplantación de subprocesos del cliente (si el subproceso está actualmente en suplantación) o por el testigo del proceso actual del cliente.

Utilizar identidades fijas en ASP.NET

En esta opción se configura la aplicación ASP.NET para que suplante una identidad fija especificada, mediante el uso del siguiente elemento en el archivo Web.config.

```
<identity impersonate="true"
    userName="YourAccount"
    password="YourStrongPassword" />
```

Pasa a ser la identidad predeterminada que se utiliza al conectarse a los recursos de la red, incluidas las bases de datos.

No se recomienda utilizar esta opción con .NET Framework versión 1.0 por dos motivos:

- Los nombres de usuario y las contraseñas aparecen como texto sin cifrar en el espacio Web (es decir, en el archivo Web.config en un directorio virtual).
- ASP.NET (en Windows 2000) requiere el privilegio "Actuar como parte del sistema operativo". Esta restricción no se aplica a Microsoft Windows Server 2003.

Para obtener más información acerca de este privilegio firme, consulte la columna sobre seguridad en la copia de agosto de 1999 de Microsoft Systems Journal (<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnmsj99/html/security0899.asp>) (en inglés).

.NET Framework versión 1.1 proporcionará una mejora para este escenario en Windows 2000. En concreto:

- Se cifrarán las credenciales.
- El proceso IIS realizará el inicio de sesión, de forma que ASP.NET no requiera el privilegio "Actuar como parte del sistema operativo".

Utilizar componentes revisados

Puede desarrollar un componente revisado para que contenga específicamente código de acceso a datos. Con los componentes revisados, es posible tener acceso a la base de datos alojando el componente en una aplicación servidor de Servicios

Empresariales (COM+) que se ejecute bajo una identidad específica o bien se puede escribir código que utilice la API **LogonUser** para llevar a cabo la suplantación.

El uso de componentes revisados fuera del proceso aumenta la seguridad ya que los saltos de proceso dificultan la tarea del atacante, especialmente si los procesos se ejecutan con identidades diferentes. La otra ventaja es que se puede aislar del resto de la aplicación el código que requiera más privilegios.

Llamar a LogonUser y suplantar la identidad específica de Windows

No debería llamar a **LogonUser** directamente desde ASP.NET. En Windows 2000, esta opción requiere proporcionar al proceso ASP.NET la identidad "Actuar como parte del sistema operativo".

Se prefiere la opción de llamar a **LogonUser** fuera del proceso ASP.NET mediante un componente revisado en una aplicación de servidor de Servicios Empresariales, tal y como se ha explicado anteriormente.

Utilizar la identidad del llamador original

Para que funcione, es necesario utilizar la delegación Kerberos y suplantar el llamador en la base de datos, directamente desde ASP.NET o desde un componente revisado.

Desde ASP.NET, agregue el siguiente código al archivo Web.config de la aplicación.

```
<identity impersonate="true" />
```

Desde un componente revisado, llame a **ColImpersonateClient**.

Utilizar la cuenta anónima de usuario de Internet

Como variación de la opción anterior, en los escenarios en los que la aplicación utiliza la autenticación por formularios o de Passport (lo que implica la autenticación anónima de IIS), puede habilitar la suplantación en el archivo Web.config de la aplicación para poder utilizar la cuenta anónima de usuario de Internet para tener acceso a la base de datos.

```
<identity impersonate="true" />
```

Con IIS configurado para la autenticación anónima, esta configuración provoca que el código de la aplicación Web se ejecute con el testigo de suplantación del usuario de Internet anónimo. En un entorno de alojamiento Web, tiene la ventaja de que permite realizar por separado la auditoría y el seguimiento del acceso a la base de datos desde varias aplicaciones Web.

Más información

- Para obtener más información y detalles acerca de la implementación utilizando la identidad del llamador original, consulte "[Transmitir el llamador original a la base de datos](#)" en el capítulo 5, "Seguridad de intranet."
- Para obtener más información acerca de cómo configurar IIS para utilizar la cuenta de usuario anónimo, consulte el capítulo 8, "[Seguridad de ASP.NET](#)."

¿Cuándo no se puede utilizar la autenticación de Windows?

En determinados escenarios de aplicaciones no es posible utilizar la autenticación de Windows. Por ejemplo:

- El cliente y el servidor de la base de datos están separados por un servidor de seguridad que no permite la autenticación de Windows.

- La aplicación necesita conectarse a una o varias bases de datos con varias identidades.
- Se conecta a bases de datos que no son de SQL Server.
- No hay una forma segura de ejecutar código en ASP.NET como un usuario de Windows específico. No puede (o no podrá) reenviar el contexto de seguridad del llamador original o desea utilizar una cuenta de servicio dedicada en lugar de conceder inicios de sesión a usuarios finales.
Especificar un nombre de usuario y una contraseña en el archivo Machine.config (en el elemento **<processModel>**) o en el archivo Web.config (en el elemento **<identity>**) para ejecutar el proceso de trabajo de ASP.NET o la aplicación es menos seguro que seguir pasos explícitos para proteger credenciales de SQL estándar.

En dichos escenarios deberá utilizar la autenticación de SQL (o el mecanismo de autenticación nativo de la base de datos) y debe:

- Proteger las credenciales de usuario de la base de datos en el servidor de aplicaciones.
- Proteger las credenciales de usuario de la base de datos durante el tránsito del servidor a la base de datos.

Si utiliza la autenticación de SQL, hay diversas formas en que puede hacer más segura la autenticación de SQL. Se describen en la siguiente sección.

Autenticación de SQL

Si la aplicación necesita utilizar la autenticación de SQL, debe tener en cuenta los siguientes puntos importantes:

- Utilice una cuenta con privilegios mínimos para conectarse a SQL.
- Las credenciales se transmiten por la red, por lo que deben ser seguras.
- La cadena de conexión de SQL (que contiene credenciales) debe ser segura.

Tipos de cadenas de conexión

Si se conecta a una base de datos de SQL Server utilizando credenciales (nombre de usuario y contraseña), la cadena de conexión tendrá este aspecto:

```
SqlConnection string = "Server=YourServer;
    Database=YourDatabase;
    uid=YourUserName;pwd=YourStrongPassword;"
```

Si necesita conectarse a una determinada instancia de SQL Server (una característica sólo disponible en SQL Server 2000 o posterior) instalada en el mismo equipo, la cadena de conexión tendrá este aspecto:

```
SqlConnection string = "Server=YourServer\Instance;
    Database=YourDatabase;uid=YourUserName;
    pwd=YourStrongPassword;"
```

Si desea conectarse a SQL Server utilizando credenciales de red, utilice el atributo Integrated Security (o el atributo **Trusted Connection**) y omita el nombre de usuario y la contraseña:

```
SqlConnection string = "Server=YourServer;
    Database=YourDatabase;
    Integrated Security=SSPI;"
```

- o bien -

```
SqlConnectionString = "Server=YourServer;  
Database=YourDatabase;  
Trusted_Connection=Yes; "
```

Si se conecta a una base de datos de Oracle utilizando credenciales explícitas (nombre de usuario y contraseña), la cadena de conexión tendrá este aspecto:

```
SqlConnectionString = "Provider=MSDAORA;Data Source=YourDatabaseAlias;  
User ID=YourUserName;Password=YourPassword; "
```

Más información

Para obtener más información acerca del uso de archivos de vínculo de datos universal (UDL) en la conexión, consulte el artículo Q308426, "[HOW TO: Use Data Link Files with the OleDbConnection Object in Visual C# .NET](#)" (en inglés), en Microsoft Knowledge Base.

Elegir una cuenta SQL para las conexiones

No utilice las cuentas integradas **sa** o **db_owner** para el acceso a datos. En su lugar, utilice cuentas con privilegios mínimos con una contraseña rigurosa.

Evite la siguiente cadena de conexión:

```
SqlConnectionString = "Server=YourServer\Instance;  
Database=YourDatabase; uid=sa; pwd=; "
```

Utilice cuentas con privilegios mínimos con una contraseña rigurosa, por ejemplo:

```
SqlConnectionString = "Server=YourServer\Instance;  
Database=YourDatabase;  
uid=YourStrongAccount;  
pwd=YourStrongPassword; "
```

Tenga en cuenta que esto no soluciona el problema de almacenar las credenciales en texto sin cifrar en los archivos Web.config. Lo que ha hecho hasta ahora es limitar el alcance de los posibles daños en caso de que la seguridad se vea comprometida, mediante el uso de una cuenta con privilegios mínimos. Para aumentar la seguridad, debería cifrar las credenciales.

Nota: Si seleccionó un orden que distingue mayúsculas de minúsculas al instalar SQL Server, el Id. de inicio de sesión también distingue mayúsculas de minúsculas.

Pasar credenciales a través de la red

Cuando se conecta a SQL Server con autenticación de SQL, el nombre de usuario y la contraseña se envían a través de la red como texto sin cifrar. Esto puede significar un problema de seguridad importante. Para obtener más información acerca de cómo asegurar el canal entre una aplicación o servidor Web y un servidor de base de datos, consulte "Comunicación segura" más adelante en este capítulo.

Asegurar las cadenas de conexión de SQL

Los nombres de usuario y las contraseñas no deberían almacenarse como texto sin cifrar en los archivos de configuración. Para obtener detalles acerca de cómo almacenar cadenas de conexión de forma segura, consulte "Almacenar cadenas de conexión a bases de datos" más adelante en este capítulo.

Autenticar con bases de datos que no son de SQL Server

Los problemas habituales que puede encontrar al conectarse a bases de datos que no son de SQL Server son similares a los escenarios en los que necesita utilizar la autenticación de SQL. Puede que necesite suministrar credenciales explícitas si los recursos de destino no admiten la autenticación de Windows. Para asegurar este tipo de escenario, debe almacenar de forma segura la cadena de conexión y debe asegurar además la comunicación a través de la red (para evitar que se intercepten las credenciales).

Más información

- Para obtener más información acerca del almacenamiento de cadenas de conexión a bases de datos, consulte "[Almacenar cadenas de conexión a bases de datos de forma segura](#)" más adelante en este capítulo.
- Para obtener más información acerca de cómo asegurar el canal hacia el servidor de base de datos, consulte "[Comunicaciones seguras](#)" más adelante en este capítulo.

Autorización

SQL Server proporciona varias opciones de autorización basadas en funciones. Giran en torno a los siguientes tres tipos de funciones admitidas por SQL Server:

- **Funciones de base de datos definidas por el usuario.** Se utilizan para agrupar a los usuarios que tienen los mismos privilegios de seguridad en la base de datos. Se agregan cuentas de grupos o usuarios de Windows a las funciones de base de datos de usuario y se establecen los permisos en cada uno de los objetos de la base de datos (procedimientos almacenados, tablas, vistas, etc.) a través de las funciones.
- **Funciones de aplicación.** Son similares a las funciones de base de datos de usuario en que se utilizan al establecer los permisos de objetos. Sin embargo, a diferencia de las funciones de base de datos de usuario, no contienen usuarios ni grupos. En su lugar, las activa una aplicación mediante un procedimiento almacenado integrado. Una vez activas, los permisos concedidos a la función determinan la capacidad del acceso a datos de la aplicación.
Las funciones de aplicación permiten a los administradores de la base de datos conceder a determinadas aplicaciones el acceso a objetos de base de datos concretos. Esto se realiza en contraposición a conceder permisos a los usuarios.
- **Funciones de base de datos fijas.** SQL Server también proporciona funciones de servidor fijas como `db_datareader` y `db_datawriter`. Estas funciones integradas se incluyen en todas las bases de datos y pueden utilizarse para conceder a un usuario conjuntos de permisos específicos de lectura (y otros de uso frecuente) en la base de datos.

Para obtener más información acerca de los diversos tipos de funciones (y de las funciones de servidor fijas que son similares a las funciones de base de datos fijas pero se aplican en el servidor y no en la base de datos), consulte los libros en

pantalla de SQL Server

(<http://www.microsoft.com/sql/techinfo/productdoc/2000/books.asp>) (en inglés).

Utilizar varias funciones de base de datos

Si la aplicación tiene varias categorías de usuarios y los usuarios de una misma categoría requieren los mismos permisos en la base de datos, la aplicación requiere varias funciones.

Cada función requiere a su vez un conjunto de permisos distinto en la base de datos. Por ejemplo, los miembros de una función Usuario de Internet pueden requerir permisos de sólo lectura en la mayoría de las tablas de una base de datos, mientras que los miembros de una función Administrador u Operador pueden requerir permisos de lectura y escritura.

Opciones

Para implementar estos escenarios, hay dos opciones principales de autorización basada en funciones en SQL Server:

- **Funciones de base de datos SQL Server definidas por el usuario.** Se utilizan para asignar permisos a objetos de la base de datos para los grupos de usuarios que tienen los mismos permisos de seguridad en la base de datos. Cuando se utilizan funciones de base de datos definidas por el usuario, se realizan comprobaciones en la puerta, se asignan usuarios a las funciones (por ejemplo, en una aplicación Web ASP.NET o en un componente revisado de nivel medio en una aplicación de servidor de Servicios Empresariales) y se utilizan varias identidades para conectarse a la base de datos, cada una de las cuales se asigna a una función de base de datos definida por el usuario.
- **Funciones de aplicación SQL.** Son similares a las funciones de base de datos definidas por el usuario en que se utilizan al asignar permisos a objetos de base de datos. No obstante, a diferencia de las funciones de base de datos definidas por el usuario, no contienen miembros y se activan desde aplicaciones individuales mediante un procedimiento almacenado integrado. Cuando se utilizan funciones de aplicación, se realizan comprobaciones en la puerta, se asignan usuarios a las funciones, se conecta a la base de datos con una identidad de servicio única y de confianza y se activa la función de aplicación SQL apropiada.

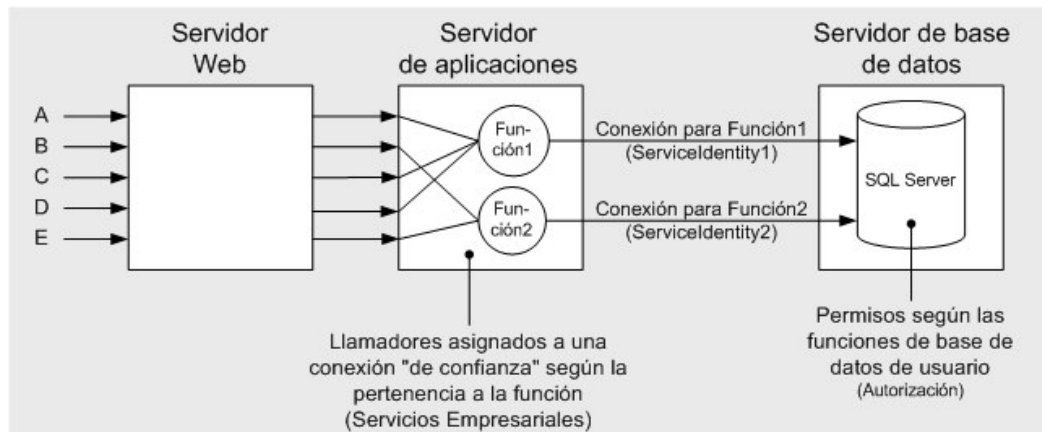
Funciones de base de datos definidas por el usuario

Si elige utilizar funciones de base de datos definidas por el usuario, debe:

- Crear varias cuentas de servicio para utilizarlas en el acceso a la base de datos.
- Asignar cada cuenta a una función de base de datos definida por el usuario.
- Establecer los permisos de base de datos necesarios para cada función de la base de datos.
- Autorizar a los usuarios en la aplicación (aplicación Web ASP.NET, servicio Web o componente de nivel medio) y utilizar a continuación la lógica de la aplicación en el nivel de acceso a datos para determinar la cuenta con la que debe conectarse a la base de datos. Se basa en la pertenencia a una función del llamador.

De forma declarativa, puede configurar métodos individuales para autorizar únicamente a los usuarios que pertenezcan a un conjunto de funciones. A continuación, debe agregar comprobaciones de funciones imperativas en el código del método para determinar la pertenencia precisa a la función, lo que determina a su vez la conexión que se debe utilizar.

La ilustración 12.4 muestra esta opción.



{Insert figure: CH12 - SQL User Database Roles.gif}

Ilustración 12.4

Conectar a SQL Server con varias funciones de bases de datos de usuarios de SQL

Para utilizar la autenticación de Windows preferida en este escenario, debe desarrollar código (con la API **LogonUser**) en un componente revisado fuera del proceso para suplantar una de las identidades del conjunto de identidades de Windows.

Con la autenticación de SQL, utiliza una cadena de conexión distinta (que contiene nombre de usuario y contraseñas distintas) según la lógica basada en funciones de la aplicación.

Más información

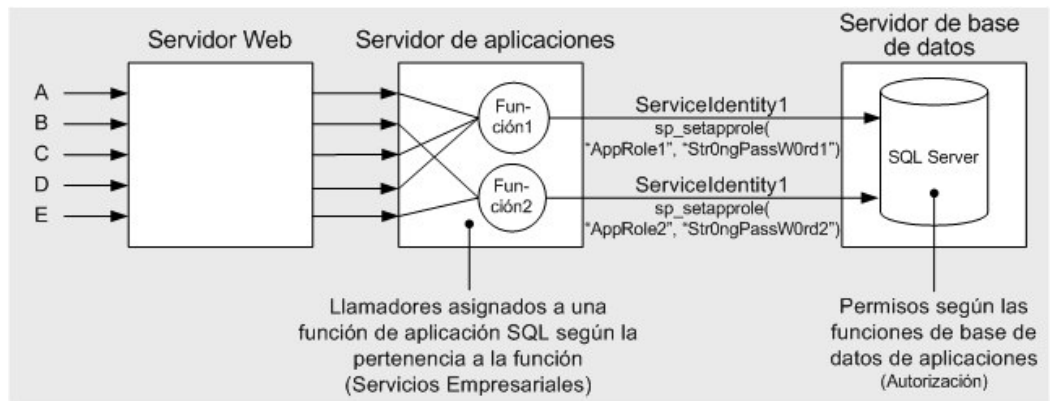
Para obtener más información acerca del almacenamiento seguro de cadenas de conexión a bases de datos, consulte "[Almacenar cadenas de conexión a bases de datos de forma segura](#)" más adelante en este capítulo.

Funciones de aplicación

Con las funciones de aplicación SQL debe:

- Crear una cuenta de servicio única para utilizarla en el acceso a la base de datos (puede ser la cuenta de proceso que se utilizó para ejecutar el proceso de trabajo de ASP.NET o una aplicación de Servicios Empresariales).
- Crear un conjunto de funciones de aplicación SQL en la base de datos.
- Establecer los permisos de base de datos necesarios para cada función de la base de datos.
- Autorizar a los usuarios en la aplicación (aplicación Web ASP.NET, servicio Web o componente de nivel medio) y utilizar a continuación la lógica de la aplicación en el nivel de acceso a datos para determinar la función de aplicación que debe activarse en la base de datos. Se basa en la pertenencia a una función del llamador.

La ilustración 12.5 muestra esta opción.



{Insert figure: CH12 - SQL Application Roles.gif}

Ilustración 12.5

Utilizar varias funciones de aplicación SQL

En la ilustración 12.5, la identidad **ServiceIdentity1** que se utiliza para tener acceso a la base de datos se obtiene normalmente del proceso de trabajo de ASP.NET o de una identidad del proceso de la aplicación de servidor de Servicios Empresariales.

En esta opción, se utiliza la misma identidad de servicio (y por lo tanto la misma conexión) para conectarse a SQL Server. Las funciones de aplicación SQL se activan con el procedimiento almacenado integrado **sp_setapprole**, que se basa en la pertenencia a la función del llamador. Este procedimiento almacenado requiere el nombre de la función y una contraseña.

Si utiliza esta opción, debe almacenar de forma segura las credenciales del nombre de la función y la contraseña. Para obtener más consejos y técnicas de almacenamiento de secretos, consulte "Almacenar cadenas de conexión a bases de datos de forma segura" más adelante en este capítulo.

Limitaciones de las funciones de aplicación SQL

Debe tener en cuenta los siguientes puntos principales antes de elegir el uso de las funciones de aplicación SQL:

- Necesita administrar credenciales de las funciones de aplicación SQL. Debe llamar al procedimiento almacenado **sp_setapprole**; para ello, pase un nombre de la función y una contraseña para cada conexión. Si está activando una función de aplicación SQL desde código administrado, no será seguro tener incrustada en el ensamblado una contraseña con texto sin cifrar.
- Las credenciales de la función de aplicación SQL se pasan a la base de datos como texto sin cifrar. Debería asegurarlas en la red mediante IPsec o SSL entre el servidor de aplicaciones y el servidor de base de datos.
- Una vez que se activa una función de aplicación SQL en una conexión, ya no puede desactivarse. Permanece activa hasta que se cierra la conexión. Tampoco es posible intercambiar dos o más funciones en la misma conexión.
- Utilice funciones de aplicación SQL únicamente cuando la aplicación utilice una identidad fija única para conectarse a la base de datos. En otras palabras, utilícelas únicamente cuando la aplicación utilice el modelo de subsistema de confianza.

Si cambia el contexto de seguridad de la conexión (por ejemplo, si se utilizara el contexto del llamador original para conectarse a la base de datos), las funciones de aplicación SQL no funcionan junto con la agrupación de la conexión.

Para obtener más información, consulte el artículo Q229564, "[PRB: SQL Application Role Errors with OLE DB Resource Pooling](#)" (en inglés), en Microsoft Knowledge Base.

Comunicación segura

En la mayoría de los escenarios de aplicaciones es necesario asegurar el vínculo de comunicación entre el servidor de aplicaciones y la base de datos. Es necesario poder garantizar:

- **La confidencialidad de los mensajes.** Los datos deben estar cifrados para garantizar que conservan la privacidad.
- **La integridad de los mensajes.** Los datos deben estar firmados para garantizar que no se han alterado.

En algunos escenarios todos los datos que se pasan entre el servidor de aplicaciones y el servidor de base de datos deben ser seguros, mientras que en otros escenarios deben asegurarse determinados elementos de datos que se envían a través de conexiones concretas. Por ejemplo:

- En una aplicación de recursos humanos de una intranet, algunos de los datos de empleados que se pasan entre el cliente y el servidor de base de datos son confidenciales.
- En escenarios de Internet como, por ejemplo, aplicaciones de banca seguras, todos los datos que se pasan entre el servidor de aplicaciones y el servidor de base de datos deben ser seguros.
- Si utiliza la autenticación de SQL, debería asegurar además el vínculo de comunicación para garantizar que los nombres de usuario y las contraseñas no sean vulnerables al software de supervisión de la red.

Opciones

Tiene dos opciones para asegurar el vínculo de red entre un servidor de aplicaciones y un servidor de base de datos:

- IPSec
- SSL (con un certificado de servidor en el equipo con SQL Server)

Nota: Debe ejecutarse SQL Server 2000 para admitir el uso de SSL. Las versiones anteriores no lo admiten. El cliente debe tener instaladas bibliotecas cliente de SQL Server 2000.

Elegir una de las opciones

El uso de IPSec o SSL depende principalmente de varios factores del entorno como, por ejemplo, las consideraciones del servidor de seguridad, las versiones de sistema operativo y de base de datos, etc.

Nota: IPSec no pretende ser un sustituto de la seguridad en la aplicación. Actualmente se utiliza como defensa en mecanismos de profundidad o para asegurar aplicaciones no seguras sin cambiarlas, así como para proteger a protocolos no TLS (como SSL) de ataques a través de la red.

Más información

- Para obtener más información acerca de la configuración de IPSec, Consulte ["Cómo utilizar IPSec para garantizar una comunicación segura entre dos servidores"](#) en la sección Referencia de esta guía.
- Para obtener más información acerca de la configuración de SSL, consulte ["Cómo utilizar SSL para proporcionar comunicaciones seguras con SQL Server 2000"](#) en la sección Referencia de esta guía.
- Para obtener más información acerca de SSL y IPSec en general, consulte el capítulo 4, ["Comunicación segura"](#).

Conectar con privilegios mínimos

Conectarse a la base de datos con privilegios mínimos significa que la conexión que se establece sólo tiene los privilegios mínimos necesarios en la base de datos. En pocas palabras, no se conecta a la base de datos con la cuenta **sa** ni con la cuenta propietaria de la base de datos. Idealmente, si el usuario actual no está autorizado para agregar ni actualizar registros, la cuenta correspondiente que se utiliza en su conexión (que puede estar agregada a una identidad que represente una función concreta) no puede agregar ni actualizar registros en la base de datos.

Cuando se conecta a SQL Server, la opción elegida debe admitir la granularidad necesaria que requiere la autorización de la base de datos. Necesita tener en cuenta en qué confía la base de datos. Puede confiar en:

- La aplicación
- Funciones definidas de la aplicación
- El llamador original

La base de datos confía en la aplicación

Piense en una aplicación financiera a la que autoriza el acceso a una base de datos. La aplicación financiera es responsable de administrar la autenticación de usuarios y de autorizar el acceso. En tal caso, puede administrar las conexiones a través de una única cuenta de confianza (que corresponde a un inicio de sesión SQL o a una cuenta de Windows asignada a un inicio de sesión SQL). Si utiliza la autenticación de Windows, normalmente implicaría permitir el acceso a la base de datos por parte de la identidad del proceso de la aplicación que llama (como el proceso de trabajo de ASP.NET o una identidad de aplicación de servidor de Servicios Empresariales).

Desde el punto de vista de la autorización, esta opción es muy poco precisa, ya que la conexión se ejecuta como una identidad que tiene acceso a todos los objetos y recursos de la base de datos que necesita la aplicación. Las ventajas de esta opción son que puede utilizar la agrupación de la conexión y que simplifica la administración porque sólo autoriza una única cuenta. La desventaja es que todos los usuarios tienen los mismos privilegios de conexión.

La base de datos confía en distintas funciones

Puede utilizar grupos de conexiones a la base de datos independientes y de confianza que correspondan a las funciones definidas por la aplicación; por ejemplo, una conexión para cajeros, otra para administradores, etc.

Estas conexiones pueden utilizar o no la autenticación de Windows. La ventaja de la autenticación de Windows es que controla la administración de credenciales y no las envía a través de la red. Sin embargo, si bien es posible utilizar la autenticación de Windows en el proceso o en la aplicación (por ejemplo, cuando se utiliza una única conexión a la base de datos), el hecho de que se necesiten varias identidades (una por cada función) presenta problemas adicionales.

Muchas aplicaciones utilizan la API **LogonUser** para establecer un testigo de acceso de Windows. Esta opción presenta dos problemas:

- Ahora tiene un problema de administración de credenciales (la aplicación debe almacenar de forma segura el nombre de usuario y la contraseña de la cuenta).
- La API **LogonUser** requiere que la cuenta del proceso de llamada tenga el privilegio "Actuar como parte del sistema operativo". Esto significa que está obligado a conceder este privilegio a la cuenta del proceso ASP.NET, lo cual no es recomendable. En su lugar puede utilizar la autenticación de SQL, pero entonces necesita proteger las credenciales en el servidor y en la red.

Nota: La restricción de **LogonUser** desaparece en Windows Server 2003.

La base de datos confía en el llamador original

En este caso, necesita transmitir el llamador original a través de varios niveles hasta la base de datos. Esto significa que los clientes necesitan credenciales de red para poder saltar de un equipo al siguiente. Esto requiere la delegación Kerberos.

Aunque esta solución proporciona un nivel de autorización preciso en la base de datos, porque se conoce la identidad del llamador original y se pueden establecer permisos por usuario en los objetos de la base de datos, afecta al rendimiento y la escalabilidad de la aplicación. La agrupación de la conexión se vuelve ineficaz (aunque sigue habilitada).

Crear una cuenta de base de datos con privilegios mínimos

En los pasos siguientes se muestra un ejemplo sencillo de cómo crear una cuenta de base de datos con privilegios mínimos. Aunque la mayoría de los administradores de bases de datos están familiarizados con estos pasos, puede ser que muchos desarrolladores no lo estén y recurran a la cuenta propietaria de la base de datos o a la cuenta **sa** para provocar el funcionamiento de las aplicaciones.

Esta práctica puede crear dificultades al pasar de un entorno de desarrollo a un entorno de pruebas y, a continuación, a un entorno de producción, ya que la aplicación pasa de un entorno abierto a uno más controlado, lo que evita que la aplicación funcione correctamente.

Para empezar, debe crear un inicio de sesión SQL para una cuenta SQL o una cuenta de Windows (de usuario o grupo). A continuación, debe agregar dicho inicio de sesión a una función de usuario de base de datos y asignar permisos a dicha función.

► Para configurar una cuenta de acceso a datos para SQL

1. Cree una nueva cuenta de usuario y agregue la cuenta a un grupo de Windows. Si administra varios usuarios, debe utilizar un grupo. Si trabaja con una única cuenta de aplicación (como una cuenta del proceso ASP.NET duplicada), puede elegir no agregar la cuenta a un grupo de Windows.
2. Cree un inicio de sesión de SQL Server para el usuario o grupo.
 - a. Inicie el **Administrador corporativo**, busque el servidor de base de datos y, a continuación, expanda la carpeta **Seguridad**.
 - b. Haga clic con el botón secundario en **Inicios de sesión** y, a continuación, haga clic en **Nuevo inicio de sesión**.
 - c. Escriba el nombre del grupo de Windows en el campo **Nombre**, y haga clic en **Aceptar** para cerrar el cuadro de diálogo **Propiedades de inicio de sesión de SQL Server**.
3. Cree un nuevo usuario de base de datos en la base de datos seleccionada, que se asigne al inicio de sesión de SQL Server.
 - a. Utilice el **Administrador corporativo** y expanda la carpeta **Bases de datos** y, a continuación, expanda la base de datos requerida en la que el inicio de sesión necesita acceso.
 - b. Haga clic con el botón secundario en **Usuarios** y, a continuación, haga clic en **Nuevo usuario de base de datos**.
 - c. Seleccione el nombre de inicio de sesión creado anteriormente.
 - d. Especifique un nombre de usuario.

- e. Configure permisos del modo en que se explica a continuación.
- 4. Conceda al usuario de la base de datos permisos de selección en las tablas a las que es necesario tener acceso y permisos de ejecución en cualquier procedimiento almacenado importante.

Nota: Si el procedimiento almacenado y la tabla son propiedad de una misma persona y el acceso a la tabla se realiza únicamente a través del procedimiento almacenado (y no es necesario tener acceso directo a la tabla), basta con conceder únicamente permisos de ejecución en el procedimiento almacenado. Esto se debe al concepto de la cadena de propiedad. Para obtener más información, consulte los libros en pantalla de SQL Server.

- 5. Si desea que la cuenta de usuario tenga acceso a todas las vistas y tablas de la base de datos, agréguelas a la función **db_datareader**.

Almacenar cadenas de conexión a bases de datos de forma segura

Hay varias ubicaciones y opciones posibles para almacenar cadenas de conexión a bases de datos, con diversos grados de seguridad y flexibilidad de configuración.

Opciones

La siguiente lista presenta las principales opciones de almacenaje de cadenas de conexión:

- Cifradas con DPAPI
- Texto sin cifrar en el archivo Web.config o en el archivo Machine.config
- Archivos UDL
- Archivos de texto personalizado
- Registro
- Catálogo de COM+

Utilizar DPAPI

Los sistemas operativos Windows 2000 y posteriores proporcionan la API de protección de datos Win32® (DPAPI) para cifrar y descifrar datos. DPAPI forma parte de la API de criptografía (Crypto API) y se implementa en Crypt32.dll. Está formada por dos métodos: **CryptProtectData** y **CryptUnprotectData**.

DPAPI es especialmente útil porque puede eliminar el principal problema de administración que tienen las aplicaciones que utilizan criptografía. Aunque el cifrado garantiza la seguridad de los datos, es necesario seguir algunos pasos adicionales para garantizar la seguridad de la clave. DPAPI utiliza la contraseña de la cuenta de usuario asociada al código que llama a las funciones de DPAPI para obtener la clave de cifrado. Como resultado, el sistema operativo (y no la aplicación) administra la clave.

¿Por qué no LSA?

Muchas aplicaciones utilizan la Autoridad de seguridad local (LSA) para almacenar secretos. DPAPI presenta las siguientes ventajas con respecto a LSA:

- Para utilizar LSA, un proceso requiere privilegios administrativos. Supone un problema de seguridad, ya que aumenta en gran medida los posibles daños de un atacante que logre poner en peligro el proceso.
- LSA proporciona únicamente un número limitado de ranuras para almacenamiento secreto, muchas de las cuales ya utiliza el sistema.

Almacén de equipo frente a almacén de usuario

DPAPI puede funcionar con el almacén de equipo o el almacén de usuario (que requiere la carga de un perfil de usuario). DPAPI tiene como valor predeterminado el almacén de usuario, aunque se puede especificar que se utilice el almacén de equipo pasando el indicador CRYPTPROTECT_LOCAL_MACHINE a las funciones de DPAPI.

La opción de perfil de usuario proporciona un nivel adicional de seguridad ya que limita los usuarios que pueden tener acceso al secreto. Únicamente el usuario que cifra los datos puede descifrarlos. Sin embargo, el uso del perfil de usuario requiere un mayor desarrollo cuando se utiliza DPAPI desde una aplicación Web ASP.NET, ya que es necesario realizar pasos explícitos para cargar y descargar un perfil de usuario (ASP.NET no carga automáticamente un perfil de usuario).

La opción de almacén de equipo requiere un menor desarrollo, ya que no es necesario administrar perfiles de usuario. Sin embargo, a menos que se utilice un parámetro de entropía adicional, es menos seguro porque cualquier usuario del equipo puede descifrar los datos. (La entropía es un valor aleatorio que dificulta el descifrado del secreto). El problema del uso de un parámetro de entropía adicional es que la aplicación debe almacenarlo de forma segura, lo que supone otro problema de administración de claves.

Nota: Si utiliza DPAPI con el almacén de equipo, la cadena cifrada es específica de un determinado equipo, por lo que debe generar los datos cifrados en cada equipo. No copie los datos cifrados entre equipos de una batería de servidores o clúster.

Si utiliza DPAPI con el almacén de usuario, puede descifrar los datos en cualquier equipo con un perfil de usuario itinerante.

Soluciones de implementación de DPAPI

En esta sección se describen dos soluciones de implementación que muestran cómo utilizar DPAPI desde una aplicación Web ASP.NET para asegurar una cadena de conexión (o cualquier tipo de secreto). En esta sección se describen las siguientes soluciones de implementación:

- **Utilizar DPAPI desde Servicios Empresariales.** Esta solución le permite utilizar DPAPI con el almacén de usuario.
- **Utilizar DPAPI directamente desde ASP.NET.** Esta solución le permite utilizar DPAPI con el almacén de equipo, que facilita el desarrollo de la solución, ya que puede llamarse a DPAPI directamente desde una aplicación Web ASP.NET.

Utilizar DPAPI desde Servicios Empresariales

Una aplicación Web ASP.NET no puede llamar a DPAPI y utilizar el almacén de usuario porque se requiere la carga de un perfil de usuario. La cuenta ASPNET que se utiliza habitualmente para ejecutar aplicaciones Web es una cuenta no interactiva, por lo que no tiene un perfil de usuario. Además, si la aplicación ASP.NET está suplantada, el subproceso de la aplicación Web se ejecuta como el usuario autenticado actual, que puede variar de una solicitud a la siguiente.

De este modo, una aplicación Web ASP.NET que desee utilizar DPAPI tiene los siguientes problemas:

- Hay errores en las llamadas a DPAPI desde una aplicación ASP.NET que se ejecute bajo la cuenta ASPNET predeterminada. El motivo es que la cuenta ASPNET no tiene un perfil de usuario, ya que no se utiliza para inicios de sesión interactivos.

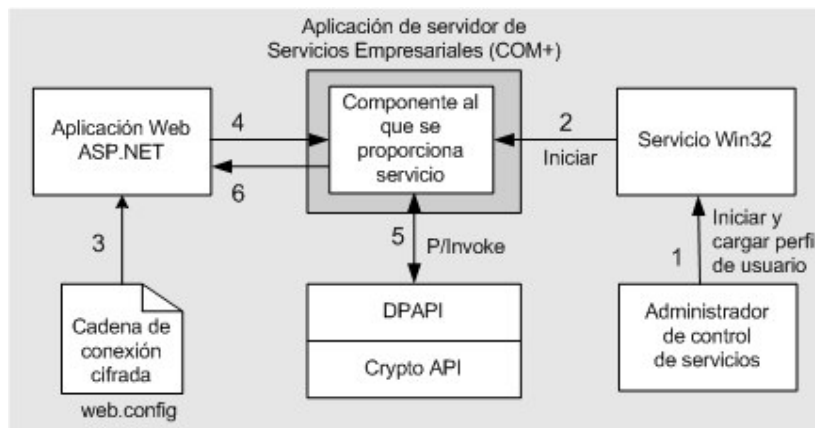
- Si se configura una aplicación Web ASP.NET para que suplante a sus llamadores, el subproceso de la aplicación ASP.NET tiene un testigo de suplantación del subproceso asociado. El inicio de sesión asociado al testigo de suplantación es un inicio de sesión de red (que se utiliza en el servidor para representar al llamador). Los inicios de sesión de red no producen la carga de perfiles de usuario.

Para solucionar este problema, puede crear un componente revisado (en una aplicación servidor de Servicios Empresariales (COM+) fuera del proceso) para llamar a DPAPI. Puede garantizar que la cuenta que se utiliza para ejecutar el componente tiene un perfil de usuario y puede utilizar un servicio Win32 para cargar automáticamente el perfil.

Nota: Para evitar el uso de un servicio Win32, se puede realizar llamadas a las funciones de administración de perfiles Win32 (**LoadUserProfile** y **UnloadUserProfile**) en el componente revisado.

Esta opción tiene dos inconvenientes. En primer lugar, las llamadas a estas API bajo solicitud afectarían considerablemente el rendimiento. En segundo lugar, estas API requieren que el código de llamada tenga privilegios administrativos en el equipo local, lo que contraviene el principio de privilegios mínimos de la cuenta del proceso de Servicios Empresariales.

La ilustración 12.6 muestra la solución DPAPI para Servicios Empresariales.



{Insert figure: CH12 - DPAPIHowTo.gif}

Ilustración 12.6

La aplicación Web ASP.NET utiliza una aplicación servidor COM+ para interactuar con DPAPI

La secuencia de eventos en tiempo de ejecución de la ilustración 12.6 es la siguiente:

1. El administrador de control de servicios de Windows inicia el servicio Win32 y carga automáticamente el perfil de usuario asociado a la cuenta en la que se ejecuta el servicio. La misma cuenta de Windows se utiliza para ejecutar la aplicación de Servicios Empresariales.
2. El servicio Win32 llama a un método de inicio en el componente revisado que inicia la aplicación de Servicios Empresariales y carga el componente revisado.
3. La aplicación Web recupera la cadena cifrada del archivo Web.config. Puede almacenar la cadena cifrada mediante el elemento **<appSettings>** del archivo Web.config, tal y como se muestra a continuación. Este elemento admite pares clave-valor arbitrarios.

```
<configuration>
```

```
<appSettings>
  <add key="SqlConnString"
    value="AQAAANCMnd8BFdERjHoAwE/Cl+sBAAAABcqc/xCHxki3" />
</appSettings>
</configuration>
```

Puede recuperar la cadena cifrada con la siguiente línea de código:

```
string connString = ConfigurationSettings.AppSettings["SqlConnString"];
```

Nota: Puede utilizar el archivo Web.config o el archivo Machine.config para almacenar cadenas de conexión cifradas. Se prefiere Machine.config porque está en un directorio del sistema fuera de un directorio virtual. Esto se explica con más detalle en la sección siguiente, "Utilizar Web.config y Machine.config".

4. La aplicación llama a un método en el componente revisado para descifrar la cadena de conexión.
5. El componente revisado interactúa con DPAPI mediante P/Invoke para llamar a las funciones de DPAPI de Win 32.
6. Se devuelve la cadena descifrada a la aplicación Web.

Nota: Para almacenar cadenas de conexión cifradas en el archivo Web.config en primer lugar, escriba una aplicación de utilidad que utilice las cadenas de conexión y llame al método **EncryptData** del componente revisado para obtener la cadena cifrada. Es fundamental ejecutar la aplicación de utilidad cuando se ha iniciado una sesión con la misma cuenta que se utilizó para ejecutar la aplicación servidor de Servicios Empresariales.

Utilizar DPAPI directamente desde ASP.NET

Si utiliza el almacén de equipo (y llama a las funciones de DPAPI con el indicador CRYPTPROTECT_LOCAL_MACHINE) puede llamar a las funciones de DPAPI directamente desde una aplicación Web ASP.NET (porque no necesita un perfil de usuario).

Sin embargo, como utiliza el almacén de equipo, cualquier cuenta de Windows que pueda iniciar una sesión en el equipo tendrá acceso al secreto. Una opción para solucionarlo sería agregar entropía, pero requiere una mayor administración de claves.

Como alternativa al uso de entropía con el almacén de equipo, se pueden considerar las opciones siguientes:

- Utilizar las ACL de Windows para restringir el acceso a los datos cifrados (si los datos se almacenan en el sistema de archivos o en el Registro).
- Especificar en el código de la aplicación el parámetro de entropía para evitar el problema de administración de claves.

Más información

- Para obtener información acerca de la creación de una biblioteca de DPAPI para utilizarla en aplicaciones Web .NET, consulte ["Cómo crear una biblioteca DPAPI"](#) en la sección Referencia de esta guía.
- Para obtener un tutorial de implementación detallado que muestre el uso de DPAPI directamente desde ASP.NET, consulte ["Cómo utilizar DPAPI \(Almacén de equipo\) desde ASP.NET"](#) en la sección Referencia de esta guía.

- Para obtener un tutorial de implementación detallado que muestre el uso de DPAPI directamente desde Servicios Empresariales, consulte "[Cómo utilizar DPAPI \(Almacén de usuario\) desde ASP.NET con Servicios Empresariales](#)" en la sección Referencia de esta guía.
- Para obtener más información acerca de la protección de datos con DPAPI en Windows, consulte el artículo de MSDN, "[Windows Data Protection](#)" (en inglés).

Utilizar Web.config y Machine.config

No se recomienda almacenar contraseñas como texto sin cifrar en el archivo Web.config. De forma predeterminada, **HttpForbiddenHandler** protege el archivo de la descarga y visualización por parte de usuarios malintencionados. Sin embargo, los usuarios que tienen acceso directamente a las carpetas que contienen los archivos de configuración pueden ver el nombre de usuario y la contraseña.

Se considera a Machine.config una ubicación de almacenamiento más segura que Web.config, ya que se ubica en un directorio del sistema (con ACL) fuera de un directorio virtual de una aplicación Web. Bloquee siempre Machine.config con las ACL.

Más información

Para obtener más información acerca de la seguridad de Machine.config, consulte el capítulo 8, "[Seguridad de ASP.NET](#)."

Utilizar archivos UDL

El proveedor de datos.NET de OLE DB admite nombres de archivo UDL en su cadena de conexión. Utilice "File Name=name.udl" para hacer referencia al archivo UDL en la cadena de conexión.

Importante: Esta opción sólo se encuentra disponible si se utiliza el proveedor de datos .NET de OLE DB para conectarse a la base de datos. El proveedor de datos .NET de SQL Server no utiliza archivos UDL.

No se recomienda almacenar archivos UDL en un directorio virtual junto con otros archivos de aplicación. Debería almacenarlos fuera de la jerarquía de directorios virtuales de la aplicación Web y asegurar a continuación el archivo o la carpeta que contiene el archivo con las ACL de Windows. También debería considerar la posibilidad de almacenar los archivos UDL en un volumen lógico independiente del sistema operativo para protegerlo de errores transversales de directorio y de la canonización del archivo.

Granularidad de las ACL

Los archivos UDL (o cualquier archivo de texto) ofrecen mayor granularidad cuando se aplican las ACL en comparación con Machine.config. Las ACL predeterminadas asociadas a Machine.config conceden acceso a una gran variedad de usuarios locales y remotos. Por ejemplo, Machine.config tiene las siguientes ACL predeterminadas:

```
MachineName\ASPNET:R
BUILTIN\Users:R
BUILTIN\Power Users:C
BUILTIN\Administrators:F
NT AUTHORITY\SYSTEM:F
```

En cambio, puede bloquear el archivo UDL de su propia aplicación en mayor medida. Por ejemplo, puede restringir el acceso de forma que sólo se permita a los administradores, la cuenta SISTEMA y la cuenta del proceso ASP.NET (que requiere acceso de lectura), tal y como se muestra a continuación.

```
BUILTIN\Administrators:F
MachineName\ASPNET:R
NT AUTHORITY\SYSTEM:F
```

Nota: Dado que los archivos UDL pueden modificarse externamente en cualquier aplicación cliente ADO.NET, las cadenas de conexión que contienen referencias a archivos UDL se analizan cada vez que se abre la conexión. Esto puede afectar el rendimiento, por lo que se recomienda que para mejorar el rendimiento se utilice una conexión estática que no incluya un archivo UDL.

► Para crear un nuevo archivo UDL

1. Utilice el Explorador de Windows y busque la carpeta en la que desea crear el archivo UDL.
2. Haga clic con el botón secundario del *mouse* en la carpeta, señale **Nuevo** y, a continuación, haga clic en **Documento de texto**.
3. Escriba un nombre de archivo con la extensión de archivo .udl.
4. Haga doble clic en el nuevo archivo para mostrar el cuadro de diálogo **Propiedades de UDL**.

Más información

Para obtener más información acerca del uso de archivos UDL en los programas de herramientas de desarrollo de Microsoft C#®, consulte el artículo Q308426, "[HOW TO: Use Data Link Files with OleDbConnection Object in VC#](#)" (en inglés) en Microsoft Knowledge Base.

Utilizar archivos de texto personalizados

Muchas aplicaciones utilizan archivos de texto personalizados para almacenar cadenas de conexión. Si opta por ello, debe tener en cuenta las siguientes recomendaciones:

- Almacene los archivos personalizados fuera de la jerarquía de directorios virtuales de la aplicación.
- Considere la posibilidad de almacenar los archivos en un volumen lógico independiente del sistema operativo para protegerlo de errores transversales de directorio y de la canonización de archivos.
- Proteja el archivo con una ACL restringida que conceda acceso de lectura a la cuenta del proceso de la aplicación.
- Evite almacenar la cadena de conexión como texto sin cifrar en el archivo. En su lugar, utilice DPAPI para almacenar una cadena cifrada.

Utilizar el Registro

Puede utilizar una clave personalizada en el Registro de Windows para almacenar la cadena de conexión. Esta información se puede almacenar en el subárbol del Registro HKEY_LOCAL_MACHINE (HKLM) o HKEY_CURRENT_USER (HKCU). En las identidades del proceso que no tienen perfiles de usuario como, por ejemplo, la cuenta ASPNET, esta información debe almacenarse en HKLM para permitir que el código ASP.NET la recupere.

Si utiliza esta opción, debe:

- Utilizar las ACL para proteger la clave del Registro mediante Regedt32.exe.
- Cifrar los datos antes de almacenarlos.

Más información

Para obtener más información acerca del cifrado de datos para almacenarlos en el Registro, consulte ["Cómo almacenar una cadena de conexión cifrada en el registro"](#) en la sección Referencia de esta guía.

Utilizar el catálogo de COM+

Si la aplicación Web incluye componentes revisados, puede almacenar cadenas de conexión en el catálogo de COM+ como cadenas de constructor. Se administran fácilmente (con la herramienta Servicios de componentes) y el código de componentes las recupera con facilidad. Los Servicios Empresariales llaman al método **Construct** de un objeto inmediatamente después de crear las instancias del objeto y pasan la cadena de construcción configurada.

El catálogo de COM+ no proporciona un grado elevado de seguridad porque la información no está cifrada; no obstante, aumenta la seguridad con respecto a los archivos de configuración gracias al salto de proceso adicional.

Para evitar el acceso al catálogo a través de la herramienta Servicios de componentes, incluya únicamente la lista de usuarios deseados en las funciones **Administrador** y **Lector** en la aplicación **System**.

El siguiente ejemplo muestra el modo de recuperar una cadena de constructor de objeto de un componente revisado.

```
[ConstructionEnabled(Default="Default Connection String")]
public class YourClass : ServicedComponent
{
    private string _ConnectionString;
    override protected void Construct(string s)
    {
        _ConnectionString = s;
    }
}
```

Para mayor seguridad, puede agregar código para cifrar la cadena de construcción antes de almacenarla y descifrarla en el componente revisado.

Más información

- Para obtener más información acerca del uso de cadenas de conexión, consulte el artículo Q271284, ["HOWTO: Access COM+ Object Constructor String in a VB Component"](#) (en inglés) en Microsoft Knowledge Base.
- Para obtener un ejemplo de código completo por parte de .NET Framework SDK, consulte el ejemplo de constructor de objeto en \Archivos de programa\Microsoft Visual Studio .NET\FrameworkSDK\Samples\Technologies\ComponentServices\ObjectConstruction.

Autenticar usuarios en una base de datos

Si crea una aplicación que necesite validar las credenciales de usuario en un almacén de base de datos, tenga en cuenta los siguientes puntos:

- Almacene valores hash de contraseña unidireccionales (con un valor salt aleatorio).
- Evite la inyección SQL al validar credenciales de usuario.

Almacenar valores hash de contraseña unidireccionales (con salt)

Las aplicaciones Web que utilizan la autenticación mediante Formularios necesitan a menudo almacenar las credenciales de usuario (incluidas las contraseñas) en una base de datos. Por motivos de seguridad, no debería almacenar contraseñas (texto sin cifrar o cifrado) en la base de datos.

Debería evitar almacenar contraseñas cifradas porque eso aumenta los problemas de administración de claves; puede asegurar la contraseña con cifrado, en cuyo caso tiene que pensar en cómo almacenar la clave de cifrado. Si la clave se convierte en vulnerable, un atacante puede descifrar todas las contraseñas del almacén de datos.

La opción preferida es:

- **Almacenar un valor hash unidireccional de la contraseña.** Vuelva a calcular el valor hash cuando haya que validar la contraseña.
- **Combine el valor hash de la contraseña con un valor salt (un número aleatorio de criptografía segura).** Al combinar el valor salt con el valor hash de la contraseña, se reduce la amenaza asociada a los ataques de diccionario.

Crear un valor salt

El siguiente código muestra cómo generar un valor salt con la funcionalidad de generación de números aleatorios que proporciona la clase **RNGCryptoServiceProvider** en el espacio de nombres **System.Security.Cryptography**.

```
public static string CreateSalt(int size)
{
    RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider();
    byte[] buff = new byte[size];
    rng.GetBytes(buff);
    return Convert.ToBase64String(buff);
}
```

Crear un valor hash (con un valor salt)

El siguiente fragmento de código muestra cómo generar un valor hash a partir de una contraseña suministrada y un valor salt.

```
public static string CreatePasswordHash(string pwd, string salt)
{
    string saltAndPwd = string.Concat(pwd, salt);
    string hashedPwd =
        FormsAuthentication.HashPasswordForStoringInConfigFile(
            saltAndPwd, "SHA1");
    return hashedPwd;
}
```

Más información

Para obtener todos los detalles de implementación de esta opción, consulte "[Cómo utilizar la autenticación mediante Formularios con SQL Server 2000](#)" en la sección Referencia de esta guía.

Ataques de inyección SQL

Si utiliza la autenticación mediante Formularios en una base de datos SQL, debería tomar las precauciones que se describen en esta sección para evitar ataques de inyección SQL. La inyección SQL es el acto de pasar código SQL adicional (y dañino) a una aplicación, que suele agregarse al código SQL legítimo de la aplicación. Todas las bases de datos SQL pueden ser objeto de la inyección SQL en mayor o menor medida, pero este capítulo se centra en SQL Server.

Debería prestar especial atención a la posibilidad de recibir ataques de inyección SQL cuando procese acciones de usuario que formen parte de un comando SQL. Si el esquema de autenticación se basa en la validación de usuarios en una base de datos SQL, por ejemplo, si utiliza la autenticación mediante Formularios en SQL Server, debe protegerse de los ataques de inyección SQL.

Si crea cadenas SQL con entrada no filtrada, la aplicación puede ser objeto de una entrada del usuario malintencionada (recuerde que nunca debe confiar en las entradas del usuario). El riesgo en este caso es que al insertar la entrada del usuario en una cadena que se convierte en una instrucción ejecutable, un usuario malintencionado puede agregar comandos SQL a las instrucciones SQL pretendidas, mediante caracteres de escape.

Los fragmentos de código de las siguientes secciones utilizan la base de datos Pubs que suministra SQL Server para mostrar ejemplos de inyección SQL.

Problema

La aplicación debe ser susceptible de recibir ataques de inyección SQL al incorporar entradas del usuario u otros datos desconocidos en consultas de base de datos. Por ejemplo, los dos fragmentos de código siguientes son susceptibles de recibir un ataque.

- Crea instrucciones SQL con acciones del usuario sin filtrar.

```
SqlDataAdapter myCommand = new SqlDataAdapter(  
    "SELECT au_lname, au_fname FROM authors WHERE au_id = '" +  
    Login.Text + "'", myConnection);
```

- Llama a un procedimiento almacenado generando una única cadena que incorpore acciones del usuario sin filtrar.

```
SqlDataAdapter myCommand = new SqlDataAdapter("LoginStoredProcedure '" +  
    Login.Text + "'", myConnection);
```

Anatomía de un ataque de inyección de cadenas SQL

Cuando se aceptan los valores de entrada del usuario sin filtrar en la aplicación (como se muestra anteriormente), un usuario malintencionado puede utilizar caracteres de escape para agregar sus propios comandos.

Piense, por ejemplo, en una consulta SQL que espera que la entrada del usuario sea un número de la seguridad social como 172-32-xxxx y que tiene el siguiente aspecto:

```
SELECT au_lname, au_fname FROM authors WHERE au_id = '172-32-xxxx'
```

Un usuario malintencionado puede escribir el siguiente texto en el campo de entrada de la aplicación (por ejemplo, un control de cuadro de texto).

```
' ; INSERT INTO jobs (job_desc, min_lvl, max_lvl) VALUES ('Important Job', 25, 100) --
```

En este ejemplo, se inserta una sentencia INSERT (pero podría ejecutarse cualquier instrucción que se permita en la cuenta utilizada para conectarse a SQL Server). El código puede ser especialmente dañino si la cuenta es miembro de la función **sysadmin** (permite utilizar comandos del Shell mediante **xp_cmdshell**) y SQL Server se ejecuta en una cuenta de dominio con acceso a otros recursos de la red.

El comando anterior produce la siguiente cadena SQL combinada:

```
SELECT au_lname, au_fname FROM authors WHERE au_id = '' ; INSERT INTO jobs (job_desc, min_lvl, max_lvl) VALUES ('Important Job', 25, 100) --
```

En este caso, el carácter de comilla sencilla (') que inicia la entrada rogue termina el literal de cadena actual de la instrucción SQL. Cierra la instrucción actual únicamente si el siguiente testigo analizado no tiene sentido como continuación de la instrucción actual, pero sí como inicio de una nueva instrucción.

```
SELECT au_lname, au_fname FROM authors WHERE au_id = ' '
```

El carácter de punto y coma (;) indica a SQL que se inicia una nueva instrucción, a la que sigue el código SQL malintencionado:

```
; INSERT INTO jobs (job_desc, min_lvl, max_lvl) VALUES ('Important Job', 25, 100)
```

Nota: No es obligatorio utilizar el punto y coma para separar instrucciones SQL. Depende de la implementación o el proveedor, pero no es necesario en SQL Server. Por ejemplo, SQL Server analizará el código siguiente como dos instrucciones independientes:

```
SELECT * FROM MyTable DELETE FROM MyTable
```

Por último, la secuencia de caracteres de guión doble (--) es un comentario SQL que indica a SQL que omita el resto del texto; en este caso, se omite el carácter de comilla sencilla de cierre ('), que de otra forma provocaría un error de análisis de SQL.

El texto completo que ejecuta SQL como resultado de la instrucción anterior es el siguiente:

```
SELECT au_lname, au_fname FROM authors WHERE au_id = '' ; INSERT INTO jobs (job_desc, min_lvl, max_lvl) VALUES ('Important Job', 25, 100) --'
```

Solución

Se pueden utilizar las siguientes opciones para llamar a SQL de forma segura desde la aplicación.

- Utilice la colección **Parameters** al crear las instrucciones SQL.

```
SqlDataAdapter myCommand = new SqlDataAdapter(
    "SELECT au_lname, au_fname FROM Authors WHERE au_id= @au_id",
    myConnection);

SqlParameter parm = myCommand.SelectCommand.Parameters.Add(
    "@au_id",
    SqlDbType.VarChar, 11);

parm.Value= Login.Text;
```

- Utilice la colección **Parameters** cuando llame a un procedimiento almacenado.

```
// AuthorLogin is a stored procedure that accepts a parameter named Login
SqlDataAdapter myCommand = new SqlDataAdapter("AuthorLogin", myConnection);
myCommand.SelectCommand.CommandType = CommandType.StoredProcedure;
SqlParameter parm = myCommand.SelectCommand.Parameters.Add(
    "@LoginId", SqlDbType.VarChar,11);

parm.Value=Login.Text;
```

Si utiliza la colección **Parameters**, sea cual sea la entrada de un usuario malintencionado, dicha entrada se tratará como un literal. Otra ventaja del uso de la colección **Parameters** es que puede exigir comprobaciones de tipo y longitud. Los valores fuera del rango desencadenan una excepción. Esto es un claro ejemplo de defensa en profundidad.

- Filtre la entrada del usuario de caracteres SQL. El siguiente método muestra cómo garantizar la seguridad de cualquier literal de cadena que se utilice en una instrucción de comparación SQL sencilla (igual a, menor que, mayor que). Lo hace asegurándose de que cualquier apóstrofo que se utilice en la cadena se interpreta correctamente mediante un apóstrofo adicional. En un literal de cadena SQL dos apóstrofes consecutivos se tratan como una instancia del carácter de apóstrofo en la cadena y no como delimitadores.

```
private string SafeSqlLiteral(string inputSQL)
{
    return inputSQL.Replace("'", "''");
}

...

string safeSQL = SafeSqlLiteral(Login.Text);
SqlDataAdapter myCommand = new SqlDataAdapter(
    "SELECT au_lname, au_fname FROM authors WHERE au_id = '" +
    safeSQL + "'", myConnection);
```

Otras prácticas recomendadas

A continuación se describen algunas medidas adicionales que pueden tomarse para limitar la capacidad de aprovecharse de los puntos débiles y para limitar el alcance de los posibles daños:

- Evite la entrada no válida en la puerta (la aplicación cliente) limitando el tamaño y el tipo de entrada. Al limitar el tamaño y el tipo de entrada se reducen considerablemente los daños posibles. Por ejemplo, si el campo de

búsqueda de la base de datos tiene once caracteres y todos ellos son numéricos, exija que la entrada cumpla estas características.

- Ejecute código SQL con una cuenta con privilegios mínimos. Esto reduce considerablemente los posibles daños.
Por ejemplo, si un usuario inyectara código SQL para eliminar una tabla de la base de datos y la conexión de SQL utilizara una cuenta sin los permisos adecuados, el código SQL no se ejecutaría correctamente. Es otro de los motivos para no utilizar la cuenta **sa** o la cuenta propietaria de la base de datos en las conexiones de SQL de la aplicación.
- Cuando hay una excepción en el código SQL, no exponga al usuario final los errores SQL mostrados por la base de datos. Registre la información de errores y muestre únicamente la información descriptiva. Así evitará detalles innecesarios que podrían ayudar a un atacante.

Proteger instrucciones de búsqueda de modelos coincidentes

Si se va a utilizar la entrada en literales de cadena de una cláusula 'LIKE', los caracteres que no sean el apóstrofo también tendrán un significado especial para la búsqueda de modelos coincidentes.

Por ejemplo, en una cláusula LIKE el carácter % significa "encontrar cero o más caracteres coincidentes". Para tratar estos caracteres de la entrada como caracteres literales sin un significado especial, también debe agregar caracteres de escape. Si no se tratan de forma especial, la consulta puede devolver resultados incorrectos; un carácter de búsqueda de modelos coincidentes que no esté precedido por un carácter de escape y que esté situado al principio o cerca del principio de la cadena también podría afectar al indizado.

En SQL Server, debería utilizarse el siguiente método para garantizar la validez de la entrada:

```
private string SafeSqlLikeClauseLiteral(string inputSQL)
{
    // Make the following replacements:
    // ' becomes ''
    // [ becomes [[]
    // % becomes [%]
    // _ becomes [_]

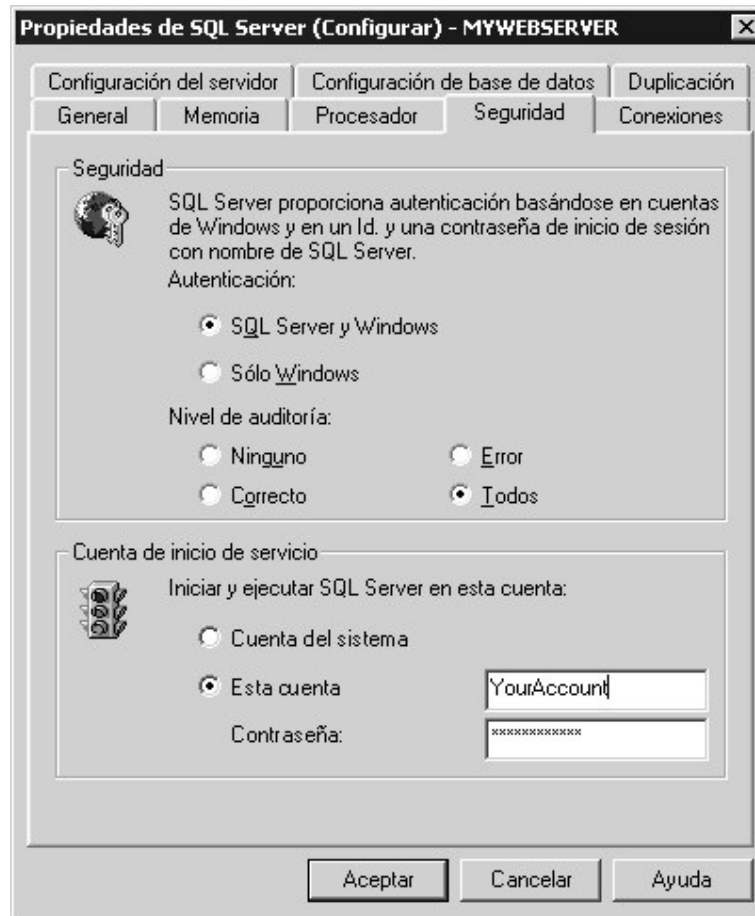
    string s = inputSQL;
    s = inputSQL.Replace("'", "''");
    s = s.Replace("[", "[[]");
    s = s.Replace("%", "[%]");
    s = s.Replace("_", "[_]");
    return s;
}
```

Auditoría

La auditoría de los inicios de sesión no está activada de forma predeterminada en SQL Server. Puede configurarla en el Administrador corporativo de SQL Server o en el Registro. En el cuadro de diálogo de la ilustración 12.7 se muestra la opción de auditoría habilitada para inicios de sesión de usuario correctos e incorrectos.

Las entradas del registro se escriben en los archivos de registro SQL que se encuentran de forma predeterminada en C:\Archivos de programa\Microsoft SQL

Server\MSSQL\LOG. Puede utilizar cualquier lector de texto para verlas, por ejemplo el Bloc de notas.



{Insert figure: SQL Server Properties Dialog.gif}

Ilustración 12.7

Cuadro de diálogo Propiedades de SQL Server con la configuración de Nivel de auditoría

También puede habilitar la auditoría de SQL Server en el Registro. Para habilitar la auditoría de SQL Server, cree la siguiente clave **AuditLevel** en el Registro y establezca como valor uno de los valores REG_DWORD que se especifican a continuación.

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSSQLServer\AuditLevel

Puede elegir uno de los siguientes valores, que le permiten capturar el nivel de detalle que desea:

- 3: captura los intentos de inicio de sesión correctos e incorrectos
- 2: captura sólo los inicios de sesión incorrectos
- 1: captura sólo los inicios de sesión correctos
- 0: no captura ningún inicio de sesión

Se recomienda activar la auditoría de inicios de sesión incorrectos porque permite determinar si alguien trata de atacar por fuerza bruta el servidor SQL Server. El registro de los intentos de inicio de sesión incorrectos no afecta apenas al rendimiento, a menos que esté siendo objeto de un ataque, en cuyo caso necesitará saberlo.

También puede ejecutar una secuencia de comandos con objetos de administración de bases de datos SQL (DMO, *Database Management Objects*). El siguiente fragmento de código muestra código VBScript de ejemplo.

```
Sub SetAuditLevel(Server As String, NewAuditLevel As SQLDMO_AUDIT_TYPE)
    Dim objServer As New SQLServer2
    objServer.LoginSecure = True      'Use integrated security
    objServer.Connect Server         'Connect to the target SQL Server
    'Set the audit level
    objServer.IntegratedSecurity.AuditLevel = NewAuditLevel
    Set objServer = Nothing
End Sub
```

Como puede ver en los libros en pantalla de SQL Server, los miembros del tipo enumerado SQLDMO_AUDIT_TYPE son:

SQLDMOAudit_All	3	Log all authentication attempts regardless of success or failure
SQLDMOAudit_Failure	2	Log failed authentication
SQLDMOAudit_Success	1	Log successful authentication
SQLDMOAudit_None	0	Do not log authentication attempts

Identidad del proceso para SQL Server

Ejecute SQL Server con una cuenta de dominio con privilegios mínimos. Al instalar SQL Server tiene la opción de ejecutar el servicio de SQL Server con la cuenta SISTEMA local o con una cuenta especificada.

No utilice la cuenta SISTEMA ni una cuenta de administrador. En su lugar, utilice una cuenta de dominio con privilegios mínimos. No necesita conceder a esta cuenta ningún privilegio concreto, pues el proceso de instalación (o el Administrador corporativo de SQL Server, si configura de nuevo el Servicio SQL después de la instalación) concede los permisos necesarios a la cuenta especificada.

Conclusión

A continuación se resumen las recomendaciones relativas al acceso a datos en las aplicaciones Web .NET:

- Utilice la autenticación de Windows para SQL Server cuando sea posible.
- Utilice cuentas con privilegios mínimos en la base de datos.
- Utilice cuentas locales con privilegios mínimos para ejecutar ASP.NET/Servicios Empresariales cuando se conecte a SQL Server.
- Si utiliza la autenticación de SQL, siga los siguientes pasos para mejorar la seguridad:
 - Utilice cuentas personalizadas con contraseñas rigurosas.
 - Limite los permisos de cada cuenta de SQL Server mediante funciones de base de datos.
 - Agregue las ACL a cualquier archivo que se utilice para almacenar cadenas de conexión.
 - Cifre las cadenas de conexión.
 - Considere la posibilidad de utilizar DPAPI para almacenar credenciales.

- Cuando utilice la autenticación mediante Formularios en SQL, tome precauciones para evitar ataques de inyección SQL.
- No almacene contraseñas de usuario en bases de datos para que los usuarios las validen. En su lugar, almacene valores hash de contraseña con un valor salt y no texto sin cifrar ni contraseñas cifradas.
- Proteja los datos confidenciales que se intercambian con SQL Server a través de la red.
 - La autenticación de Windows protege las credenciales pero no los datos de aplicación.
 - Utilice IPSec o SSL.