MANUAL SQL SERVER MANAGEMENT STUDIO – RICSE POMALAZA SOFÍA

MANUAL, **MODULO 4,** SEGURIDAD EN SQL SERVER:

1. Introduccion a la Seguridad en SQL Server:

   SQL Server implementa seguridad en dos niveles:
   - Seguridad de servidor: controla quién puede conectarse al servidor.
   - Seguridad de base de datos: controla qué puede hacer cada usuario dentro de una base de datos.

   El objetivo principal es proteger los datos de accesos no autorizados, controlar permisos, y garantizar integridad y confidencialidad.

   a) Conceptos de seguridad básicos:

   - **Login**: Credencial para conectarse al servidor SQL.
   - **Usuario**: Entidad que representa al login dentro de una base de datos específica.
   - **Permisos**: Derechos para realizar acciones (como SELECT, INSERT, EXECUTE).
   - **Principales**: Usuarios, roles o aplicaciones que pueden recibir permisos.
   - **Securables**: Objetos sobre los que se aplican los permisos (bases, tablas, procedimientos, etc.).

   b) Auntenticacion y autorización:

   - **Autenticación**: Proceso de verificar la identidad del usuario que intenta conectarse.
     - Tipos:

- **Autenticación de Windows**: Usa las credenciales del sistema operativo.
- **Autenticación de SQL Server (mixta)**: Usa usuario y contraseña definidos en SQL Server.

- **Autorización**: Proceso de determinar qué acciones puede realizar un usuario después de autenticarse. Se basa en los permisos asignados a usuarios o roles.

c) Roles y permisos:

Roles: Agrupaciones de permisos que se asignan a varios usuarios. Facilitan la administración.

Tipos:

- **Roles fijos de servidor** (ej. sysadmin, securityadmin): controlan acciones a nivel de servidor.
- **Roles fijos de base de datos** (ej. db_owner, db_datareader): controlan acciones en bases específicas.
- **Roles definidos por el usuario**: puedes crearlos según tus necesidades.

```
CREATE USER Juan FOR LOGIN JuanLogin;
EXEC sp_addrolemember 'db_datareader', 'Juan';
```

Ejemplo de asignación:

Permisos comunes:

- SELECT, INSERT, UPDATE, DELETE (tablas)
- EXECUTE (procedimientos)
- ALTER, CREATE, DROP (objetos)

2. Protección de Datos:
   SQL Server ofrece varias herramientas para proteger los datos sensibles frente a accesos no autorizados, robos o alteraciones. Las principales técnicas son:

   A. Cifrado de datos:

El cifrado protege los datos convirtiéndolos en una forma ilegible para usuarios no autorizados.

Tipos de cifrado en SQL Server:

1. **Cifrado a nivel de columna** (Cell-Level Encryption):
   - Cifra valores específicos de columnas.
   - Usa funciones como EncryptByPassPhrase y DecryptByPassPhrase.

```
-- Ejemplo: cifrado de una columna
SELECT EncryptByPassPhrase('clave123', 'DatoConfidencial');
```

2. **Transparent Data Encryption (TDE)**:

- Cifra toda la base de datos (archivos MDF y LDF).
- La aplicación no nota la diferencia: el cifrado es transparente.

```
-- Habilitar TDE requiere certificados y claves simétricas
ALTER DATABASE modulo_tienda SET ENCRYPTION ON;
```

3. **Always Encrypted**:

- Cifra datos a nivel de cliente (fuera del motor de SQL Server).
- SQL Server nunca ve los datos en texto plano.

B. Controles de acceso:

Controlan **quién puede ver o modificar qué datos**. Se basa en:

- **Permisos de seguridad** (GRANT, DENY, REVOKE).
- **Roles** (servidor y base de datos).
- **Inicios de sesión (logins)** y **usuarios**.

```
-- Ejemplo: otorgar permiso solo de lectura
GRANT SELECT ON Productos TO UsuarioSoloLectura;
```

También puedes aplicar:

- **Vistas filtradas** o VIEW WITH SCHEMABINDING para restringir acceso.
- **Políticas de Row-Level Security (RLS)** para filtrar automáticamente las filas que puede ver un usuario.

C. Auditoria de seguridad

La auditoría permite **registrar quién accede, qué cambia y cuándo lo hace**, para detectar accesos sospechosos o violaciones.

Opciones comunes:

1. **SQL Server Audit** (ediciones Enterprise y algunas Standard):
   - Registra acciones específicas (como login, lectura, escritura).
   - Puedes exportar los registros a un archivo.
2. **Triggers de auditoría**:
   - Disparadores (AFTER INSERT, AFTER UPDATE, etc.) que registran cambios en tablas.

```
-- Ejemplo de trigger para auditar cambios
CREATE TRIGGER tr_Auditoria_Productos
ON Productos
AFTER UPDATE
AS
INSERT INTO LogCambios (IdProducto, Fecha, Usuario)
SELECT IdProducto, GETDATE(), SYSTEM_USER
FROM inserted;
```

3. Prevención de Ataques:

   La seguridad en SQL Server no solo implica proteger datos, sino **prevenir ataques maliciosos** que puedan comprometer el sistema, como inyecciones, accesos no autorizados o explotación de errores.

   A. Vulnerabilidad Comunes en SQL Server:

- **Inyección SQL**: ocurre cuando una aplicación permite que un atacante inyecte código SQL malicioso.

  - Ejemplo:

```
SELECT * FROM Usuarios WHERE Usuario = 'admin' --' AND Clave = ''
```

  - **Credenciales débiles**: usuarios con contraseñas fáciles o sin expiración.
  - **Permisos excesivos**: dar permisos de sysadmin o db_owner a usuarios que no lo requieren.
  - **Bases de datos sin cifrado** o sin respaldo, accesibles desde el sistema de archivos.
  - **Puertos abiertos o servicios expuestos a internet** sin seguridad.

B. Medidas de Proteccion Contra Ataques

Usar consultas parametrizadas para prevenir inyecciones SQL:

```
-- En .NET, por ejemplo:
cmd.CommandText = "SELECT * FROM Usuarios WHERE Usuario = @usuario";
```

  - **Reforzar autenticación**:
    - Usar autenticación de Windows siempre que sea posible.
    - En autenticación SQL, forzar contraseñas seguras.

  - **Principio de menor privilegio**:
  - Dar solo los permisos estrictamente necesarios.

```
GRANT SELECT ON Productos TO UsuarioConsulta;
```

- **Cifrado de datos sensibles** con TDE o Always Encrypted.
- **Auditorías y registros** activos para detectar accesos sospechosos.
- **Firewall y configuración de red**: cerrar puertos no usados, limitar conexiones remotas.
- **Deshabilitar funciones no usadas** como xp_cmdshell, que pueden ser explotadas.

```
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;
EXEC sp_configure 'xp_cmdshell', 0;
RECONFIGURE;
```

C. Plan de respuesta a incidentes

Un buen plan de seguridad incluye saber **qué hacer si algo falla**. Debe incluir:

1. **Detección**:
   - Revisar logs de auditoría, fallos de login, alertas del sistema.
   - Herramientas: SQL Server Audit, Extended Events, SIEM externo.
2. **Contención**:
   - Revocar accesos comprometidos.
   - Cambiar contraseñas, suspender sesiones activas.
3. **Evaluación**:
   - Analizar qué se afectó (datos, usuarios, rendimiento).
   - Verificar integridad con DBCC CHECKDB.
4. **Recuperación**:
   - Restaurar backups si es necesario.
   - Aplicar parches de seguridad.
5. **Prevención futura**:
   - Corregir la causa raíz.
   - Capacitar al equipo y actualizar políticas.