POLITÉCNICA

Seguridad en Bases de Datos

Bases de datos

Departamento de Sistemas Informáticos

E.T.S.I. de Sistemas Informáticos

Universidad Politénica de Madrid

Índice

INTRODUCCIÓN

Introducción

- La seguridad en bases de datos es crucial para proteger la información confidencial y garantizar la integridad de los datos.
- Implementar una combinación de medidas de seguridad es fundamental para mitigar las amenazas potenciales.
- La vigilancia constante y las actualizaciones periódicas son clave para mantener la seguridad de la base de datos a largo plazo.

AMENANZAS A LA SEGURIDAD DE LAS BASES DE DATOS

Amenaza 1: Acceso No Autorizado

- **Descripción**: Intentos de acceder a la base de datos sin permisos adecuados.
- **Ejemplo**: Un empleado descontento intenta acceder a la base de datos de recursos humanos para ver información salarial de sus colegas sin tener autorización.

Amenaza 2: Inyección de SQL

- **Descripción**: Ataques que aprovechan vulnerabilidades en consultas SQL para obtener acceso no autorizado.
- **Ejemplo**: Un atacante inserta código SQL malicioso en un campo de entrada de un formulario web para manipular la base de datos y extraer información confidencial, como nombres de usuario y contraseñas.

Amenaza 3: Fugas de Información

- **Descripción**: Divulgación no autorizada de datos sensibles.
- **Ejemplo**: Un empleado descarga una lista de clientes de la base de datos y la comparte con un competidor, violando la política de privacidad y comprometiendo la confidencialidad de los datos.

Amenaza 4: Modificaciones No Autorizadas

- **Descripción**: Alteración de datos por parte de usuarios no autorizados.
- **Ejemplo**: Un hacker compromete las credenciales de un administrador de la base de datos y modifica registros financieros para desviar fondos a una cuenta bancaria controlada por él mismo.

ESTRATEGIAS DE SEGURIDAD

Estrategia 1: Autenticación y Autorización

En las bases de datos en fundamental el proceso de autenticación (verificar la identidad del usuario) como de autorización (controlar los permisos de acceso de los usuarios).

Para gestionarlo SQL dispone de mecanismos de creación de usuarios.

La creación de un usuario permite a una persona o una aplicación acceder a la base de datos y realizar diversas operaciones, como consultar datos, insertar registros, actualizar información o eliminar información, dependiendo de los permisos otorgados al usuario

Creación de usuarios

Para crear un usuario debemos estar conectados a la base de datos con un usuario que disponga de permisos suficientes para llevar a cabo tal acción

La sentencia para crear usuarios es:

```
CREATE USER 'nombre_de_usuario' IDENTIFIED BY 'contraseña_del_usuario';
```

Donde:

- `nombre_de_usuario`: es el nombre que se dará al nuevo usuario (debe ser único).
- `contraseña_del_usuario`: es la contraseña que se utilizará para acceder (usar reglas estándar de contraseñas).

Asignación de permisos

```
GRANT PRIVILEGE ON schema.tabla TO 'nombre_de_usuario' WITH GRANT OPTION;
```

Asigna los permisos que consideremos necesarios a un usuario

- `PRIVILEGE`: Uno o más de los siguientes valores que permiten ejecución de las sentencias homónimas: `CREATE`, `ALTER`, `DROP`, `INSERT`, `UPDATE`, `DELETE`, `SELECT`; `*` para todos los permisos
- `schema.tabla`: El `schema` y la tabla(s) sobre la que aplicar los permisos, siendo `*` equivalente a *todos* (e.g. `s.*` todas tabla de `s`, `*.*` toda tabla de todo `schema`)
- `WITH GRANT OPTION` (opcional): , que puede omitirse, otorga al usuario la posibilidad de asignatar permisos iguales o inferiores a los suyos a otros usuarios

`FLUSH PRIVILEGES` tras asignar permisos fuerza su refresco en algunos SGBD

Asignación de permisos sobre vistas

Las vistas toman un papel crucial en la privacidad de las bases datos cuando se combinan con una gestión de permisos adecuada

Por ejemplo, ante una vista creada como:

```
CREATE VIEW miBD.miVista AS SELECT ...
```

Es posible establecer un permiso tal que:

```
GRANT SELECT ON miBD.miVista TO 'nombre_de_usuario';
```

De tal forma que `nombre_de_usuario` solo pueda consultar la información proporcionada por `miVista` del `miBD`

Revocación de permisos

Al igual que podemos crear permisos, podemos revocarlos. Para ello empleamos:

```
REVOKE PRIVILEGE ON base_de_datos.tabla FROM 'nombre_de_usuario';
```

Que funciona de forma análoga a `GRANT`.

Consultar los permisos de un usuario

Podemos consultar los permisos de un usuario con:

```
SHOW GRANTS FOR 'nombre_de_usuario';
```

Eliminar usuario

Para eliminar un usuario usaremos la siguiente sentencia:

```
DROP USER 'nombre_de_usuario';
```

Estrategia 2. Encriptación de datos

La encritación de datos consiste en codificación de datos para proteger su confidencialidad.

En MySQL existen diferente mecanismos que permiten esta encriptación.

Encriptación en la aplicación

- Encripta los datos antes de enviarlos a MySQL desde la aplicación.
- Utiliza algoritmos como AES o RSA.
- Desencripta los datos cuando se recuperan.

Funciones de Encriptación de MySQL

- `AES_ENCRYPT(str, key)`: Encripta una cadena de texto utilizando el algoritmo AES.
- `AES_DECRYPT(str, key)`: Desencripta una cadena de texto encriptada utilizando el algoritmo AES.
- Puedes usar estas funciones en las consultas SQL para encriptar y desencriptar datos:

```
SELECT CAST(AES_DECRYPT(AES_ENCRYPT('hola', '1234'), '1234') AS CHAR); # hola
```

- `AES_ENCRYPT` genera un binario (`BLOB`) con los datos codificados y `AES_DECRYPT` los decodifica. `CAST` permite volver a transformar los datos *string* para poder mostrarlos

Columnas Encriptadas (MySQL Enterprise Edition)

- Define columnas como encriptadas.
- MySQL encripta automáticamente los datos al insertarlos en estas columnas.
- Los datos se desencriptan automáticamente al recuperarlos.

SSL/TLS

- Habilita SSL/TLS para cifrar la comunicación entre la aplicación y MySQL y asegura una transferencia segura de datos entre la aplicación y el servidor de MySQL.
- Primero debes obtener un certificado SSL/TLS válido de una autoridad de certificación confiable o generarlo tú mismo.
- Después abre el archivo de configuración de MySQL (`my.cnf` o `my.ini`) y agrega o modifica las siguientes líneas:

```
[mysqld]
ssl-ca=/ruta/al/archivo/ca-cert.pem
ssl-cert=/ruta/al/archivo/server-cert.pem
ssl-key=/ruta/al/archivo/server-key.pem
```

Si usas Docker puedes customizar el fichero de configuración con `-v /my/custom:/etc/mysql/conf.d` al arrancar el contenedor.

SSL/TLS

- A continuación reinicia el servidor MySQL para que los cambios en la configuración surtan efecto.
- Verifica si SSL/TLS está habilitado ejecutando la siguiente consulta SQL:

```
SHOW VARIABLES LIKE 'have_ssl'; # Cono  SSL/TLS  habilitado have_ssl debería ser YES.
```

- Finalmente conectate a MySQL utilizando SSL/TLS desde el cliente, especifica la opción `--ssl-mode` al iniciar sesión:

```
mysql --ssl-mode=REQUIRED -u usuario -p
```

Estrategia 3. Auditoría de Seguridad

- La auditoría de seguridad en bases de datos es el proceso de monitoreo y registro de actividades relacionadas con el acceso y uso de la base de datos. Permite:
  - Identificar y Prevenir Brechas de Seguridad: La auditoría permite identificar intentos de acceso no autorizado, inyecciones de SQL, modificaciones no autorizadas y otras actividades maliciosas que podrían comprometer la seguridad de la base de datos.
  - Garantizar el Cumplimiento Normativo: Muchas regulaciones, como GDPR, HIPAA y PCI DSS, requieren que las organizaciones implementen medidas de auditoría de seguridad para proteger los datos personales y sensibles.
  - Detectar Comportamientos Anómalos: El monitoreo constante de la actividad de la base de datos permite detectar patrones y comportamientos anómalos que podrían indicar una amenaza de seguridad, como accesos inusuales o intentos de acceso a datos sensibles.
  - Investigación de Incidentes: En caso de un incidente de seguridad, la auditoría proporciona un registro detallado de las actividades ocurridas en la base de datos, lo que facilita la investigación y la respuesta rápida ante incidentes.

Estrategia 4. Actualizaciones y Parches

- Mantener el software de la base de datos actualizado para protegerse contra vulnerabilidades conocidas.
- Por ejemplo, la *release* MySQL 8.0.35 incluye entre sus mejoras de seguridad: *Binary packages that include curl rather than linking to the system curl library have been upgraded to use curl 8.4.0. Important issues fixed in curl version 8.4.0*. Es decir, actualizo sus dependencias hacia la librería *curl* con contenía importantes problemas de seguridad.

MEJORES PRÁCTICAS

Mejores Prácticas

- **Principio de Menor Privilegio**: Asignar los permisos mínimos necesarios para cada usuario.
- **Validación de Datos de Entrada**: Filtrar y validar los datos ingresados para prevenir inyecciones de SQL.
- **Respaldo Regular**: Realizar copias de seguridad de la base de datos de forma regular para evitar pérdida de datos en caso de un incidente de seguridad.