UTB
Universidad Tecnológica Boliviana

La paz, 30 de noviembre de 2023

Técnicas y mecanismos para la seguridad de sistemas de información con bases de datos

Asignatura	:	Base de datos I
Docente	:	Ing. Angela Guadalupe Santos Quisbert
Estudiante	:	Fabio Camacho Encinas
Carrera	:	Ingeniería de sistemas

Contenido

1. Introducción

El presente informe describe tres mecanismos frecuentemente utilizados para fortalecer la protección y seguridad de un sistema de información con bases de datos: Autorización de acceso a bases de datos, control de acceso aplicando listas de control de acceso en el ámbito del diseño de la red en la que se implementa una base de datos, y prevención de vulnerabilidades a inyecciones SQL en un sistema de información.

2. Autorización de acceso a bases de datos

En la administración y diseño de un sistema de información aplicar los conceptos de autorización es un aspecto fundamental que garantiza la seguridad e integridad del sistema: Definir claramente los tipos de usuario, el nivel de confidencialidad de los objetos contenidos dentro de la base de datos del sistema, los permisos designados a cada tipo de usuario y que operaciones dichos usuarios pueden realizar con los objetos en cuestión.

2.1. Seguridad e integridad

La seguridad e integridad son términos que con frecuencia se utilizan juntos, en realidad son conceptos muy distintos. La seguridad se refiere a la protección de datos contra la revelación, alteración o destrucción no autorizada; la integridad se refiere a la exactitud o validez de los datos. En otras palabras, la seguridad implica asegurar que los usuarios están autorizados para llevar a cabo lo que tratan de hacer y la integridad implica asegurar que lo que tratan de hacer es correcto.

Existen tres aspectos principales de la seguridad en las bases de datos:

- **La confidencialidad:** Se cumple cuando sólo las personas autorizadas (en su sentido amplio podríamos referirnos también a sistemas) pueden conocer los datos o la información correspondiente.
- **La integridad:** Consiste en que sólo las personas autorizadas puedan variar (modificar o borrar) los datos; además, deben quedar pistas para control posterior y para auditoría.

- **La disponibilidad:** Se cumple si las personas autorizadas pueden acceder a tiempo a la información. El disponer de la información después del momento necesario puede equivaler a la no disponibilidad.

El problema de la seguridad tiene varios aspectos, entre ellos:

- **Controles físicos:** El lugar donde se encuentran los servidores debe estar protegido físicamente contra la entrada de intrusos.
- **Cuestiones de política interna:** El propietario del sistema decide quién puede tener acceso a qué datos.
- **Control de acceso a datos**: En el caso de utilización de contraseñas, se deben indicar las características de las contraseñas, y la frecuencia de cambio.

Por lo que la seguridad en las bases de datos trata de garantizar la seguridad contra accesos no autorizados, para lo cual se pueden utilizar varios mecanismos:

- **Identificación y autenticación** de usuarios a través de identificadores de inicio de sesión y contraseñas.
- **Uso de técnicas de cifrado**, para proteger datos en BD distribuidas o con acceso por red o Internet.
- **Diferentes tipos de usuarios**, en especial para el administrador de la BD con permisos para: creación de cuentas, concesión y revocación de privilegios y asignación de los niveles de seguridad.

2.2. Autorización a nivel de SGBD

Los sistemas de gestión de bases de datos (SGBD) usualmente cuentan con mecanismos para la creación y administración de sujetos, objetos y privilegios de autorización dentro del servidor que alberga la base de datos en cuestión, lo que permite definir los parámetros de seguridad del sistema de bases de datos que implementa estos mecanismos.

2.3. Sujetos de autorización

Pueden ser usuarios, grupos de usuarios y roles. Son entidades del sistema a las que se les puede conceder autorizaciones; por ejemplo, una persona individual Juan Pérez, grupos de personas todo un departamento de ventas, personas operando en algo en particular, programas de aplicación y computadoras remotas.

2.4. Objetos de autorización

Los objetos de autorización pueden ser las tablas y sus atributos. Son componentes pasivos del sistema a los que se les debe dar protección ante accesos no autorizados; por ejemplo, las bases de datos en sí, las afinidades, las hileras, las columnas, las vistas.

Un término importante de mencionar es la ***granularidad***, que se utiliza para referirse al tamaño de los objetos de seguridad; un sistema que sólo permite o impide acceso a la totalidad de la base de datos como una unidad tiene una granularidad grande, en comparación con uno que permite el acceso o lo impide a un atributo en particular, tiene una granularidad pequeña.

2.5. Privilegios o acciones de autorización

A continuación, se mencionan términos relacionados con la autorización de las bases de datos que a lo largo de este módulo se estarán utilizando.

Los privilegios o acciones de autorización pueden ser leer, escribir, borrar, ejecutar, seleccionar, insertar o actualizar. Son los tipos de operaciones que se pueden ejecutar sobre los objetos del sistema, identifica lo que el sujeto puede hacer con el objeto, las cuales pueden ser: lectura, inserción, eliminación, modificación, destrucción y concesión.

La inserción y la creación parecieran similares, pero la inserción significa agregar datos a una estructura existente y creación significa la elaboración de dicha estructura.

Eliminar significa quitar los datos y destruir significa eliminar datos y estructuras. La acción de conceder se refiere a dar el permiso de modificar datos a otro sujeto. Las limitantes de las autorizaciones especifican limitaciones en los permisos en relación con el sujeto, el objeto y la acción.

El método más común de imponer el control de acceso discrecional en un sistema de base de datos se basa en conceder y revocar privilegios.

2.6. Tipos de privilegios discrecionales

En SQL2, el concepto identificador de autorización se utiliza para hacer referencia a una cuenta de usuario o grupo de cuentas de usuario.

El SGBD debe ofrecer acceso selectivo a cada relación de la base de datos según sus cuentas específicas, también es posible que se controlen las operaciones, con lo que tener una cuenta no necesariamente confiere a su titular toda la funcionalidad que puede ofrecer el SGBD.

Hay dos niveles de asignación de privilegios para usar el sistema de base de datos:

1.- Nivel cuenta	En este nivel el ABD específica los privilegios particulares que tiene cada cuenta, independientemente de las relaciones de la base de datos.
2.- Nivel relación o tabla	En este nivel podemos controlar los privilegios para tener acceso a cada relación o vista individual de la base datos.

2.6.1. Privilegios a nivel cuenta

Se aplican capacidades conferidas a la cuenta misma y puede incluir los siguientes privilegios:

El privilegio **CREATE SCHEMA o CREATE TABLE**, para crear esquemas o unas relaciones de la base de datos.
El privilegio **CREATE VIEW** para crear la vista.
El privilegio **ALTER** para aplicar cambios a esquemas tales como agregar o eliminar atributos de las relaciones.
El privilegio **DROP** para eliminar las relaciones o vistas.
El privilegio **MODIFY** para insertar, eliminar o modificar tuplas.
El privilegio **SELECT** para obtener información de la base de datos.

Estos privilegios de nivel de cuenta se aplican a la cuenta en general y su definición se deja a los manejadores del SGBD y no al SQL2.

2.6.2. Privilegios a nivel relación

Los privilegios a nivel relación se especifican para cada usuario las relaciones individuales a las que se puede aplicar cada tipo de instrucción

La concesión y revocación de privilegios sigue por lo general el modelo de autorización para privilegios discrecionales, denominado modelo de matriz de acceso, donde las filas de la matriz M representan sujetos (usuarios, cuentas, programas) y las columnas representan objetos (relaciones, registros, columnas, vistas, operaciones).

Cada posición M(i, j) de la matriz representa los tipos de privilegios (lectura, escritura, actualización) que el sujeto i tiene para el objeto j.
Para controlar concesión y revocación de privilegios de la relación R de una base de datos se le asigna una cuenta propietario, que es la cuenta con que se creó la relación.

2.7. Revocación de privilegios

Cuando se desea conceder temporalmente algún privilegio a un usuario y luego revocárselo una vez terminada su tarea, se necesita un mecanismo para revocar privilegios.

En SQL se incluye una instrucción **REVOKE** para cancelar privilegios.

2.8. Otorgación de privilegios (GRANT OPTION)

Supongamos que el propietario A de una relación R concede un privilegio de R a otra cuenta B con la opción (**GRANT OPTION**), esto significa que B también podrá conceder este privilegio de R a otras cuentas.

El control de acceso obligatorio suele combinar los mecanismos de control de acceso discrecional con las políticas de seguridad adicional que clasifiquen los datos y los usuarios de acuerdo con ciertas clases de seguridad.

La necesidad de una seguridad multinivel existe en aplicaciones gubernamentales, militares y de espionaje, así como en otras industriales y corporativas.

Las clases de seguridad usuales son:

máximo secreto	**(TS:Top secret)**
secreto	**(S:secret)**
confidencial	**(C:confidential)**
no confidencial	**(U:unclassified)**

Donde **TS** es el nivel más alto y **U** el más bajo.

Existen otros esquemas de clasificación de seguridad más complejos, en los que las clases de seguridad se clasifican en un entramado.

Se utilizará el sistema con cuatro niveles de clasificación de seguridad. **TS>=S>=C>=U.**

El modelo utilizado en la seguridad multinivel es el Modelo Bell-LaPadula, el cual asigna a cada sujeto (usuario, cuenta, programa) y objeto (relación, tupla, columna, vista, operación) una de las clasificaciones de seguridad TS, S,C,U.

La base de datos estadística sirve para producir cifras estadísticas sobre diversas poblaciones. Puede contener datos confidenciales sobre muchos individuos y estos datos deben protegerse para que los usuarios no tengan acceso a ellos.

Una población es un conjunto de tuplas de una relación (tabla) que satisfacen alguna condición de selección.

En las consultas estadísticas se aplica la función estadística de una población de tuplas; por lo tanto, las bases de datos estadísticas:

- Dan importancia de la privacidad de los datos individuales.

- Consultas: funciones agregadas (totalizadoras).
- Interrogaciones anómalas: con respuestas referidas a un individuo.

Fuente:

https://cursos.clavijero.edu.mx/cursos/070_bdII/modulo3/contenidos/tema3.1.html?opc=1

3. Control de acceso

En ingeniería de redes, el control de acceso se refiere a la gestión y regulación de quién tiene permisos para acceder a recursos específicos en una red o sistema. Puede aplicarse a varios niveles, desde el acceso físico a un edificio o sala de servidores hasta el acceso lógico a sistemas y datos, como una base de datos. El objetivo principal del control de acceso es garantizar la seguridad y proteger la integridad y confidencialidad de la información.

3.1. Listas de control de acceso

Lista de Control de Acceso (ACL) se refiere a un conjunto específico de reglas utilizadas para filtrar el tráfico de red, especialmente en configuraciones de seguridad informática. Las ACL también permiten el acceso a objetos específicos del sistema, como directorios o archivos, a usuarios autorizados y deniegan el acceso a usuarios no autorizados.

Las ACL se encuentran principalmente en dispositivos de red con capacidad de filtrado de paquetes, como routers y switches.

3.1.1. Tipos de ACL

Existen cuatro tipos de ACL que desempeñan diferentes funciones en una red: estándar, reflexiva, extendida y dinámica:

3.1.1.1. ACL estándar

Este tipo sólo permite evaluar las direcciones IP de origen de los paquetes. No son tan potentes como las ACL extendidas pero utilizan menos potencia de cálculo. También utilizan los números 1300-1999 o 1-99 para que el router pueda identificar la dirección específica como la dirección IP de origen.

3.1.1.2. ACL extendidas

Estos tipos de ACL permiten bloquear el origen y el destino para hosts específicos o para toda la red. Con las ACL extendidas es posible filtrar el tráfico en función de los protocolos (IP, TCP, ICMP y UDP).

3.1.1.3. ACL reflexivas

También conocidas como ACL de sesión IP, las ACL reflexivas utilizan detalles de sesión de capa superior para filtrar el tráfico.

3.1.1.4. ACL dinámicas

Como sugiere el término, las ACL dinámicas son fiables en ACL extendidas, Telnet y autenticación. Conceden a los usuarios acceso a un recurso sólo si el usuario autentica el dispositivo mediante telnet.

3.2. ACL y aplicación para seguridad en bases de datos

Control de Acceso a Nivel de Red: En entornos de red, las ACL suelen utilizarse para filtrar el tráfico entrante y saliente en dispositivos como routers y switches. Esto también puede aplicarse a los servidores de bases de datos para controlar qué direcciones IP o rangos de direcciones tienen permiso para acceder a la base de datos. Limitar el acceso a direcciones IP específicas ayuda a prevenir intentos no autorizados de conexión.Control de Acceso a Nivel de Objeto:

Las ACL también se aplican a objetos específicos dentro de la base de datos, como tablas, vistas, procedimientos almacenados, etc. Esto significa que puedes especificar qué usuarios o roles tienen permisos para realizar operaciones específicas en esos objetos. Por ejemplo, puedes permitir que ciertos usuarios tengan solo permisos de lectura en una tabla, mientras que otros

tengan permisos de escritura.

Fuente: **https://itglobal.com/es-es/company/glossary/access-control-list/**

4. Prevención contra inyecciones SQL

4.1. Definición de Inyección SQL

SQL Injection ó **Inyección SQL** es una vulnerabilidad que permite al atacante enviar o "inyectar" instrucciones SQL de forma maliciosa y malintencionada dentro del código SQL programado para la manipulación de bases de datos, de esta forma todos los datos almacenados estarían en peligro. La finalidad de este ataque es poder modificar del comportamiento de consultas a través de parámetros no deseados, pudiendo así falsificar identidades, obtener y divulgar información de la base de datos (contraseñas, correos, información relevante, entre otros), borrar la base de datos, cambiar el nombre a las tablas, anular transacciones, el atacante puede convertirse en administrador de la misma.

Esto ocurre normalmente a la mala filtración de las variables en un programa que tiene o crea SQL, generalmente cuando solicitas a un usuario entradas de cualquier tipo y no se encuentran validadas, como por ejemplo su nombre y contraseña, pero a cambio de esta información el atacante envía una sentencia **SQL** invasora que se ejecutará en la base de datos.

4.2. Ejemplo de ataque SQL Injection

Existen muchas formas de ataques, uno de los más frecuentes es donde se valida una consulta como verdadera. Por ejemplo:

```
SELECT * FROM usuarios WHERE username = 'atacante' AND password
= 'mi_clave' OR 1=1;
```

Podemos observar que esta consulta está formada por el condicional **OR** que devolverá verdadero al cumplirse al menos una de las dos expresiones por lo que siempre será verdadero ya que **1 = 1**, cuando esto se ejecuta la base de datos arroja el total de registros en la tabla aunque el nombre de usuario y contraseña sean incorrectos puesto que la condición **OR 1=1** siempre se cumple.

Fuente : https://openwebinars.net/blog/que-es-sql-injection/

4.3. Medidas de prevención

Existen distintos tipos de medidas que se deben considerar cuando se necesita comunicar un sistema con una base de datos relacional a través del uso de SQL.

A continuación, se mencionan algunas medidas básicas que se deberían implementar:

- Uso de consultas parametrizadas: Prepared Statements.
- Uso de procedimientos: Stored Procedures.
- Validación de entrada de datos de usuarios
- Escapar todas las entradas permitidas de los usuarios.

4.3.1. Uso de consultas parametrizadas: Prepared Statements.

Preparar las declaraciones sql y almacenarlas en una variable antes de la ejecución es una forma simple y segura de programar. Al hacerlo de forma anticipada se evita que un atacante inserte declaraciones en nuestra base de datos, como se puede ver en el segundo ejemplo.

Los lenguajes más usados poseen métodos para parametrizar de manera segura las declaraciones que precisan datos de entrada de un usuario. Alguno de estos lenguajes son:

- Java EE – usar PreparedStatement() en nuestros parámetros dinámicos (variable Bind en inglés)
- .NET – usar consultas parametrizadas como SqlCommand() o OleDbCommand() en nuestros parámetros dinámicos.
- PHP – es posible usar PDO para base de datos genéricas con una fuerte parametrización de las consultas o en caso de usar un driver específico para una base de datos es necesario buscar una función segura para preparar nuestra declaración, por ejemplo, para MySQL es necesario usar bind_param().

4.3.2. Uso de procedimientos almacenados: Stored Procedures.

Al ejecutar procedimientos directamente en la base de datos es necesario tener el cuidado de no incluir ninguna generación de declaración SQL dinámica no segura. Estos procedimientos deben tener validación de las entradas y un adecuado “escape” de las mismas.

4.3.3. Validación de entrada de datos de usuarios

En todos los casos, y no solo para prevenir SQL injections, es necesario la correcta validación de la entrada de datos por parte de los usuarios.

No es recomendable que se usen variables dinámicas para nombres de tablas o columnas, así como tampoco para indicar el orden de clasificación (ASC o DESC). En caso de necesitarlas se deben usar validaciones previas como convertir las entradas a variables booleanas o usar funciones SWITCH, Sort of, etc.

4.3.4. Escapar todas las entradas permitidas de los usuarios: Escape characters.

Escapar entradas de usuarios para convertirlas en otro formato como strings, debe ser usado con cuidado ya que no evita todas las inyecciones. Esta técnica escape characters depende de cada motor de base de datos por lo que es importante implementar un control que impida un posible bypass de esta medida.

4.4. Ejemplos

A continuación dejamos algunos ejemplos de códigos que permitirían ataques de SQLi.

Ejemplo 1: Separar resultados en páginas, usando PHP y Postgres.

```
<?php
$índice    = $argv[0];
$consulta  = "SELECT id, name FROM products ORDER BY name LIMIT 20 OFFSET
$índice;";
$resultado = pg_query($conexión, $consulta);
```

Un usuario normalmente utiliza los botones “siguiente” y “atrás” para navegar entre los resultados, lo cual modificaría el valor decimal de la variable “índice” en la URL.

Este comportamiento no generaría problemas, pero si un agente malicioso decide agregar el siguiente código en la URL:

```
0;
insert into pg_shadow(usename,usesysid,usesuper,usecatupd,passwd)
    select 'crack', usesysid, 't','t','crack'
    from pg_shadow where usename='postgres';
--
```

Entonces la variable `$consulta` quedaría de la siguiente manera:

```
$consulta  = "SELECT id, name FROM products ORDER BY name LIMIT 20
OFFSET 0;

insert into pg_shadow(usename,usesysid,usesuper,usecatupd,passwd)

    select 'superusuario', usesysid, 't','t','password'

    from pg_shadow where usename='postgres';

--;"
```

y como resultado estaría creando el usuario “superusuario” con privilegios, por lo tanto quedaría habilitado para llevar a cabo actividades maliciosas como las que se describieron anteriormente en el artículo.

Una forma de corregir este problema sería el uso de PDO (PHP Data Objects):

```
<?php

$stmt  = $pdo->prepare("SELECT id, name FROM products ORDER BY name
LIMIT 20 OFFSET :índice;");

$índice = $argv[0];

$stmt->bindParam(':indice', $indice);

$stmt->execute()

?>
```

Ejemplo 2: Bypass de autenticación

Supongamos una aplicación web que tiene un formulario de autenticación que acepta como entradas un usuario y una contraseña.

Este formulario está siendo procesado por un código que contiene la siguiente declaración SQL:

```
consulta = "SELECT * FROM users WHERE username = "'" + username + "'
AND password = '" + password + "'"
```

Como se puede apreciar la consulta a la base de datos está construída mediante el uso de sentencias SQL y se asignan a las variables de forma directa los valores introducidos por el usuario.

En este caso un usuario malintencionado podría ingresar como usuario:

```
admin y como contraseña pass’ OR ‘1’=’1.
```

La consulta a la base de datos final sería la siguiente:

```
consulta = SELECT * FROM users WHERE username = 'admin' AND
(password = ' pass' OR '1'='1 ')
```

Esta consulta traería todos los datos asociados al usuario privilegiado ‘admin’ porque la condición booleana siempre será verdadera.

Fuente: https://blog.lacnic.net/ciberseguridad/como-prevenir-un-ataque-de-inyeccion-de-sql

5. Conclusión y aportes

Diseñar e implementar una base de datos es una tarea que debe ser emprendida con mucho rigor y seriedad, especialmente en el aspecto de seguridad. Las bases de datos muy a menudo contienen información muy valiosa, ya sea para un emprendimiento comercial, entidad financiera, servicios estatales o empresas de desarrollo e investigación tecnológica. La perdida, daño a la integridad o trafico malintencionado de esta información puede afectar de manera considerable la economía y la vida de muchas personas.

Con frecuencia, la información dentro de una base de datos es vulnerable al acceso o cambios no previstos, tanto por los mismos operadores dentro de una organización que implementa una base de datos, como de usuarios terceros no autorizados intentando acceder a información confidencial.

La gestión de la autorización por medio de roles dentro de un sistema de información es un mecanismo efectivo para asegurar la integridad de los datos, definiendo claramente que usuarios pueden acceder a determinados objetos y que operaciones estos pueden realizar con la información.

La implementación de listas de control de accesos es útil para filtrar el trafico en la red donde opera un sistema de información con una base de datos. El uso de ACLs permite bloquear hosts, conceder accesos, definir rangos de direcciones IP especificas para la conexión con un servidor de base de datos, entre otras configuraciones útiles para robustecer la seguridad en la estructura de red de un sistema de información.

Al mismo tiempo, es también importante que los sistemas de información con bases de datos relacionales sean diseñados con mecanismos, técnicas o arquitecturas que neutralicen los intentos de vulneración por medio de inyecciones SQL.

Para este propósito, es efectivo que el software diseñado para el sistema de gestión de información implemente procedimientos almacenados, donde las instrucciones que realiza una base de datos, están almacenadas en el gestor de base de datos y no así a nivel cliente.

También es útil la aplicación de arquitecturas de diseño de software que impidan que el usuario final se comunique directamente con el servidor de base de datos, siendo una opción

viable optar por diseños con arquitectura en capas, donde las capas del usuario se comunican directamente con las capas lógicas del software de administración de información, pero no directamente con la capa de datos.