SEGURIDAD EN LAS BASES DE DATOS

DEFINICIÓN DE UN ESQUEMA DE SEGURIDAD

Al concepto de seguridad también se le puede llamar privacidad.

El problema de la seguridad consiste en lograr que los recursos de un sistema sean, bajo toda circunstancia, utilizados para los fines previstos.

LA FIABILIDAD DEL SISTEMA

EL CONCEPTO DE SEGURIDAD LO MEDIMOS EN:

- La protección del sistema frente a ataques externos.
- La protección frente a caídas o fallos en el software o en el equipo.
- La protección frente a manipulación por parte de usuarios no autorizados.

PRINCIPIOS BÁSICOS PARA LA SEGURIDAD

Suponer que el diseño del sistema es público:

- El defecto debe ser: sin acceso.
- Chequear permanentemente.
- Los mecanismos de protección deben ser simples, uniformes y construidos en las capas más básicas del sistema.

MEDIDAS DE SEGURIDAD

FÍSICAS: Controlar el acceso al equipo.

- mediante tarjetas de acceso…

PERSONAL: Acceso solo de personal autorizado.

- **identificación directa de personal**…

SGBD: Uso de herramientas que proporcione el SGBD

- perfiles de usuario, vistas, restricciones de uso de vistas…

MEDIDAS DE SEGURIDAD

HAY DOS TIPOS DE SEGURIDAD:

- ***DIRECCIONAL***

Se usa para otorgar y revocar privilegios a los usuarios a nivel de archivos, registros o campos en un modo determinado (consulta o modificación).

MEDIDAS DE SEGURIDAD

HAY DOS TIPOS DE SEGURIDAD:

- ***OBLIGATORIA***

- Sirve para imponer seguridad de varios niveles tanto para los usuarios como para los datos.
- Para eso se utilizan mecanismos de protección.

REQUISITOS PARA LA SEGURIDAD DE LAS BD

- La base de datos debe ser protegida contra el fuego, el robo y otras formas de destrucción.
- Los datos deben ser reconstruibles, ya que siempre pueden ocurrir accidentes.
- Los datos deben poder ser sometidos a procesos de auditoria.
- El sistema debe diseñarse a prueba de intromisiones, no deben poder pasar por alto los controles.
- Ningún sistema puede evitar las intromisiones malintencionadas, pero es posible hacer que resulte muy difícil eludir los controles.
- El sistema debe tener capacidad para verificar que sus acciones han sido autorizadas.
- Las acciones de los usuarios deben ser supervisadas, de modo tal que pueda descubrirse cualquier acción indebida o errónea.

CARACTERÍSTICAS PRINCIPALES

El objetivo es proteger la Base de Datos contra accesos no autorizados.

LAS 3 PRINCIPALES CARÁCTERÍSTICAS DE LA SEGURIDAD EN UNA BASE DE DATOS SON:

- ***La Confidencialidad de la información***
- ***La Integridad de la información***
- ***La Disponibilidad de la información***

TIPOS DE USUARIOS

HAY DOS TIPOS DE USUARIOS:

- Usuario con derecho a crear, borrar y modificar objetos y que además puede conceder privilegios a otros usuarios sobre los objetos que ha creado.

- Usuario con derecho a consultar, o actualizar, y sin derecho a crear o borrar objetos. Privilegios sobre los objetos, añadir nuevos campos, indexar, alterar la estructura de los objetos, etc.

IDENTIFICACIÓN Y AUTENTIFICACIÓN

En un SGBD existen diversos elementos que ayudan a controlar el acceso a los datos.

En primer lugar el sistema debe identificar y autentificar a los usuarios utilizando alguno de las siguientes formas:

IDENTIFICACIÓN Y AUTENTIFICACIÓN

- Código y contraseña
- Identificación por hardware
- Conocimiento, aptitudes y hábitos del usuario
- Información predefinida (Aficiones, cultura…)

IDENTIFICACIÓN Y AUTENTIFICACIÓN

Además, el administrador deberá especificar los privilegios que un usuario tiene sobre los objetos:

- Usar una B.D.
- Consultar ciertos datos
- Actualizar datos
- Crear o actualizar objetos
- Ejecutar procedimientos almacenados
- Referenciar objetos
- Indexar objetos
- Crear identificadores

MATRIZ DE AUTORIZACIÓN

La seguridad se logra si se cuenta con un mecanismo que limite a los usuarios a su vista o vistas personales.

La norma es que la base de datos relacionales cuente con dos niveles de seguridad:

- Relación: Puede permitírsele o impedírsele que el usuario tenga acceso directo a una relación.

- Vista: Puede permitírsele o impedírsele que el usuario tenga acceso a la información que aparece en un vista.

INYECCIÓN SQL

¿QUÉ ES LA INYECCIÓN SQL?

INYECCIÓN SQL

- Ejemplo de inyección SQL

SELECT * FROM usuarios WHERE usuario =

‘ " + Usuario + “ ‘ and password =‘ "+ pass + “ ‘;

INYECCIÓN SQL

- Un usuario cualquiera colocaría su nombre y su password de la siguiente manera:

SELECT * FROM usuarios WHERE usuario =

' pepe ' and password =' 020304 '

INYECCIÓN SQL

- Hasta aquí todo normal, pero un usuario podría modificar el campo password:

SELECT * FROM usuarios WHERE usuario =

' pepe ' and password =' 020304 ' OR password LIKE '%'

INYECCIÓN SQL

- Como hemos visto la inyección SQL se ha hecho con el fin de burlar la restricción de acceso, pero se pueden realizar cosas más desastrosas en la BD, como por ejemplo:

DROP TABLE usuarios

PROTEGERSE DE INYECCIÓN SQL

- ASIGNACION DE MÍNIMOS PRIVILEGIOS
  - Debe tener los privilegios necesarios, ni mas ni menos.
- VALIDAR TODAS LAS ENTRADAS
  - Especifique el tipo de dato de entrada, si son números, asegúrese de que son solo números.
- EMPLEO DE PROCEDIMIENTOS ALMACENADOS
  - Utilizar procedimientos almacenados y aceptar los datos del usuario como parámetros en lugar de comandos sql.
- UTILIZAR COMILLAS DOBLES EN LUGAR DE SIMPLES
  - Puesto que las comillas simples finalizan las expresiones SQL, y posibilitan la entrada de expresiones de más potencia.

PROTEGERSE DE INYECCIÓN SQL

- La inyección SQL es fácil de evitar en la mayoría de los lenguajes de programación que desarrollan aplicaciones web

PROTEGERSE DE INYECCIÓN SQL

- EN PHP
  - Para MySQL, la función a usar es mysql_real_escape_string:

Ejemplo:

```
$query_result = mysql_query("SELECT * FROM usuarios
WHERE nombre = \"" .
mysql_real_escape_string($nombre_usuario) . "\"");
```

PROTEGERSE DE INYECCIÓN SQL

- EN JAVA
  - En Java, tenemos que usar la clase PreparedStatement

En vez de:

Connection con = (acquire Connection) Statement stmt = con.createStatement(); ResultSet rset = stmt.executeQuery("SELECT * FROM usuarios WHERE nombre = '" + nombreUsuario + "';");

Habría que poner:

Connection con = (acquire Connection) PreparedStatement pstmt = con.prepareStatement("SELECT * FROM usuarios WHERE nombre = ?"); pstmt.setString(1, nombreUsuario); ResultSet rset = pstmt.executeQuery();

PROTEGERSE DE INYECCIÓN SQL

- EN C#
  - El siguiente ejemplo muestra cómo prevenir los ataques de inyección de código usando el objeto SqlCommand

PROTEGERSE DE INYECCIÓN SQL

En vez de:

```
using( SqlConnection con = (acquire connection) ) { con. Open();
using( SqlCommand cmd = new SqlCommand("SELECT * FROM
usuarios WHERE nombre = '" + nombreUsuario + "'", con) ) {
using( SqlDataReader rdr = cmd.ExecuteReader() ){ ... } } }
```

Habría que usar:

```
using( SqlConnection con = (acquire connection) ) { con. Open();
using( SqlCommand cmd = new SqlCommand("SELECT * FROM
usuarios WHERE nombre = @nombreUsuario", con) ) {
cmd.Parameters.AddWithValue("@nombreUsuario",
nombreUsuario); using( SqlDataReader rdr =
cmd.ExecuteReader() ){ ... } } }
```

PREGUNTAS

...