Database Security

La seguridad y protección de bases de datos está recibiendo mayor atención y presupuesto de las organizaciones debido al incremento en las brechas de datos (*data breaches*).

Data Breaches

Se refiere a una brecha o ruptura que permite el acceso no autorizado a los datos.

Un incidente reciente es el de Epsilon, una firma de marketing de correos electrónicos que envía billones de correos anualmente. La brecha involucra la exposición de los correos de los clientes de muchas marcas reconocidas.

La brecha de seguridad promedio puede costar a una compañía entre $90 y $365 por registro perdido. Sin embargo, hay que tomar en cuenta otros factores como los costos legales de la pérdida.

Las brechas en los datos son caras, incluso en el límite de los $90. Considere, por ejemplo, el caso típico de una brecha de datos. En 2008, la Health Net Federal Services reportó que cientos de doctores en 11 estados tuvieron expuesta su información personal en el sitio web de una compañía. El número total de registros era de 103,000. ¿Eso cuánto costó? En el límite bajo ($90), el costo fue de $9.3 millones, pero en el límite superior el costo ascendía a los $31.4 millones.

Por esta razón tiene sentido invertir algo de tiempo y dinero en mejorar la seguridad y proteger los datos en los sistemas de bases de datos.

Database Security Basics

La regla principal es que todos los recursos de bases de datos son controlados por el DBMS. No otras autorizaciones por defecto deben ser dadas a ningún otro usuario solo porque ingresó (login) al DBMS. Por lo tanto, para que un usuario esté capacitado para realizar cualquier operación o función, debe existir alguna de las siguientes condiciones:

- Al usuario se le otorgó la habilidad de realizar esa operación, o
- La operación se le concedió a todos los usuarios de forma genérica

Usando las funciones de seguridad del DBMS, el DBA puede establecer el ambiente de forma que solo ciertos usuarios o aplicaciones puedan realizar ciertas operaciones en ciertos datos en la base de datos. Diferentes checks pueden ser establecidos para cada tipo de acceso a cada tipo de información, y que a diferentes usuarios se le puedan asignar diferentes derechos de acceso para acceder a diferentes objetos de la BD.

Muchos aspectos de seguridad de bases de datos requieren diferentes utilidades, procedimientos y comandos para implementar. Cuando los usuarios requieren acceso a múltiples bases de datos, en múltiples servidores distribuidos a lo largo de diferentes ubicaciones físicas, la seguridad de BD se vuelve una tarea complicada, ya que los comandos se deben repetir uno a uno en cada BD, no hay un repositorio central para modificar y eliminar

fácilmente la configuración de seguridad del usuario en múltiples bases de datos simultáneamente.

Aunque el DBA es típicamente el responsable de administrar la seguridad de base de datos, algunas organizaciones transfieren la responsabilidad de esta tarea a una función diferente de seguridad que controla toda la seguridad de TI de la compañía.

En un nivel alto, la seguridad de bases de datos se resume a responder los siguientes puntos:

- ¿Quién es? (Autenticación)
- ¿Quién puede hacerlo? (Autorización)
- ¿Quién puede verlo? (Encriptación)
- ¿Quién lo hizo? (Auditoría)

La fuerte autenticación es la piedra angular de cualquier plan de implementación de seguridad. Es imposible controlar la autorización y rastrear el uso sin ésta. Antes de otorgar la autorización para usar los recursos de BD, un inicio de sesión (login) debe establecerse por cada usuario del DBMS. Los logins comúnmente se refieren como cuentas o IDs de usuario.

Cuando el DBMS controla la adición de inicios de sesión, el DBA debe proporcionar cierta información sobre el inicio de sesión cuando se crea. La siguiente información debe estar relacionada con cada usuario:

- Contraseña.
- Base de datos por defecto. BD a la cual el usuario se va a conectar inicialmente durante el login.
- Lenguaje por defecto.
- Nombre del usuario.
- Detalles adicionales: sobre el usuario para el que se creó el inicio de sesión: correo electrónico, número de teléfono, ubicación de la oficina, unidad de negocios, etc.

Las contraseñas deben cambiarse con regularidad a través del tiempo para que sea difícil para los hackers ganar acceso al DBMS. Los usuarios que no cambien su contraseña deben ser deshabilitados hasta que lo hagan.

Cuando un usuario del DBMS ya no requiere más acceso al DBMS o abandona la compañía, el DBA debe eliminar ese login del usuario del sistema lo más pronto posible. Sin embargo, el login no puede eliminarse si el usuario se encuentra actualmente usando la BD o si éste es el propietario de algún objeto de la BD.

Por esta razón, es importante limitar el número de usuarios que pueden crear objetos en la BD a sólo los DBAs. Además, el DBMS puede ofrecer la capacidad de bloquear el login, lo cual lo deshabilita pero no lo elimina, por lo que puede ser habilitado en el futuro. Seguir las siguientes reglas:

- Bloquear logins que puede ser necesario reactivarlos
- Eliminar logins que no se ocuparán ser reactivados

El uso de la BD por parte de un invitado (guest usage) también puede ser permitido configurando un nombre de usuario especial que permita a los usuarios acceder a la BD como un invitado.

Otorgando y Revocando Autoridad (Grant y Revoke)

El DBA controla la seguridad y autorización de base de datos usando el Lenguaje de Control de Datos (DCL), el cual es uno de los tres subtipos de SQL. Los comandos DCL se usan para controlar cuales usuarios tienen acceso a cuales objetos y comandos. Los comandos DCL incluyen dos tipos:

- *GRANT* asigna un permiso a un usuario de la BD
- *REVOKE* eliminar un permiso de un usuario

Para usar el comando GRANT, el usuario debe ser propietario del objeto de la BD, se le otorgó autoridad de grupo de alto nivel o se le asignó la WITH GRANT OPTION ("con permiso de otorgar") cuando se le otorgó el permiso. Esta última opción permite al usuario pasar la autoridad para otorgar los privilegios a otros.

Generalmente, el uso de esta cláusula depende del tipo de práctica de administración:

- La **administración descentralizada** es más fácil de establecer pero más difícil de controlar. Mientras más usuarios obtengan la autoridad para otorgar privilegios, el enfoque de autoridad de la BD se amplía.
- La **administración centralizada** es más fácil de administrar pero coloca la carga en el administrador centralizado como el único árbitro de privilegios (el único que los otorga).

Tipos de privilegios

Existen diferentes tipos de privilegios que pueden ser otorgados o revocados de los usuarios de BD.

Los siguientes tipos de privilegios son los más comunes que proveen los actuales DBMS:

- **Tabla**. Controla quien puede acceder y modificar los datos dentro de tablas.
- **Objeto de base de datos**. Controla quien puede crear nuevos objetos de base de datos y eliminar objetos existentes.
- **Sistema**. Controla quien puede realizar ciertos tipos de actividades generales del sistema.
- **Programa**. Controla quien puede crear, modificar y usar programas de la base de datos.
- **Procedimiento almacenado**. Controla quien puede ejecutar funciones específicas y procedimientos almacenados.

Otorgando privilegios de tabla

Los *privilegios de tabla* pueden ser otorgados para permitir a usuarios acceder tablas, vistas y columnas dentro de tablas y vistas. Los siguientes pueden ser otorgados.

- SELECT
- INSERT
- UPDATE
- DELETE
- ALL: todos los anteriores

Por ejemplo, para permitir al usuario user7 eliminar filas de la tabla Titles, se puede utilizar el comando:

GRANT DELETE on Titles to user7;

Algunos privilegios de tabla pueden ser especificados a nivel de columna. Hacer esto puede ser deseable cuando ciertos usuarios deben tener la capacidad de modificar columnas específicas de una tabla pero no otras columnas. Solo los privilegios SELECT y UPDATE pueden ser otorgados a nivel de columna. Por ejemplo, para permitir al usuario user7 actualizar solo la columna au_id en la tabla Titles, se usa:

```
GRANT UPDATE on Titles (au_id) to user7;
```

Granting Database Object Privileges

Los *privilegios de objeto de base de datos* controlan qué usuarios tienen permiso para crear estructuras de bases de datos. Generalmente, el DBMS provee opciones para otorgar privilegios CREATE en cada tipo de objeto, incluyendo bases de datos, tablespaces, tablas, índices, triggers, etc.

Por ejemplo, para permitir a los usuarios user5 y user9 crear tablas e índices, se usa:

```
GRANT  CREATE table,
       CREATE index
TO user5,
   user9;
```

La capacidad de crear objetos es usualmente reservada al DBA. Si estos privilegios se le otorgan a otros, el número de objetos puede llegar a ser difícil de controlar, por lo que el DBA debería dejarse estas características sólo para él o para usuarios importantes como SAs.

Granting System Privileges

Los *privilegios de sistema* controlan qué usuarios pueden usar ciertas funciones del DBMS y ejecutar ciertos comandos del DBMS. Los privilegios del sistema disponibles varían de DBMS a DBMS pero pueden incluir la habilidad para archivar logs, apagar y reiniciar los servidores de base de datos, manejar el almacenamiento, etc.

Por ejemplo, para permitir al usuario user6 empezar rastros (traces) de rendimiento, usar:

```
GRANT TRACE
TO    user6;
```

Granting Program and Procedure Privileges

Otorgar el privilegio EXECUTE permite al usuario ejecutar un programa o stored procedure. Por ejemplo, para permitir a los usuarios user1 y user9 ejecutar un stored procedure llamado proc1, usar:

```
GRANT EXECUTE on proc1
TO    user1, user9;
```

Granting to PUBLIC

Como alternativa de otorgar permiso a un usuario de base de datos, el DBA puede escoger otorgar una autorización en particular a PUBLIC, donde el DBMS permite a cualquiera que ingrese al DBMS una autoridad en particular.

Por ejemplo, para otorgar a cualquiera la autoridad para eliminar filas de la tabla Titles, usar:

```
GRANT DELETE on titles to PUBLIC;
```

Los DBAs deben ejercer precaución cuando otorgan cualquier privilegio a PUBLIC.

Cuando un privilegio se otorga a PUBLIC, el DBA pierde el control del objeto o recurso: cualquiera puede acceder o usar el objeto. Inevitablemente, los usuarios abusarán de los recursos PUBLIC.

PUBLIC Authority and the System Catalo

Un dilema en común enfrentado por muchos DBAs es si otorgar o no acceso PUBLIC a tablas del catálogo del sistema. Por una parte, la metadata contenida en el catálogo es información útil que necesitan muchos DBAs, diseñadores o analistas. Por otra parte, el catálogo frecuentemente contiene información sensible que puede ser explotada por hackers.

Revoking Privileges

El comando REVOKE es usado para remover privilegios que fueron anteriormente otorgados. Por supuesto, los privilegios serán automáticamente revocados cuando el objeto de la base de datos se elimina.

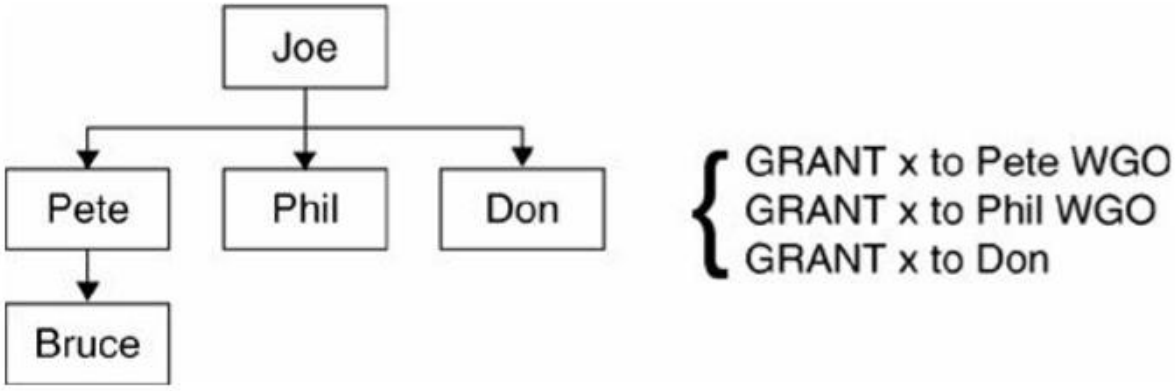
Por ejemplo, para revocar la habilidad de actualizar la columna au_id de la tabla Titles al usuario user7, usar:

```
REVOKE UPDATE on titles (au_id) from user7;
```

Revocar un privilegio PUBLIC no elimina automáticamente ese privilegio de cualquier usuario al cual le fue otorgado ese privilegio de forma separada en un comando GRANT.

Cascading REVOKEs

Cuando los privilegios son revocados, el DBMS debe decidir si revokes adicionales son necesarios. Cuando un revoke causa que el DBMS revoque privilegios relacionados adicionales, se conoce como *cascading REVOKEs*.



Considere la jerarquía de autoridad de la figura 14.2.

Para minimizar el impacto de REVOKEs en cascada, evite otorgar privilegios usando la opción WITH GRANT OPTION. Mientras menos sean los usuarios que pueden otorgar subsecuentes privilegios, más fácil es manejar y administrar una viable infraestructura de seguridad del DBMS.

Chronology and Revokes

En algunos DBMS es posible otorgar un privilegio a todos los usuarios excepto a alguno en específico usando los siguientes comandos, en orden:

```
GRANT DELETE on titles to public;
COMMIT;
REVOKE DELETE on titles from userx;
```

Estos comandos permiten eliminar a todos los usuarios y quitar el privilegio de eliminar a el usuario especificado

Por lo tanto, en este caso, el derecho a eliminar lo tendrán todos los usuarios menos el usuario ‘userx’.

Label-Based Access Control

Un número creciente de DBMSs ofrecen control de acceso basado en etiquetas (LBAC) para asegurar que cada pieza de datos es asegurada de forma que solo usuarios autorizados puedan realizar ciertas funciones. Se puede configurar LBAC para especificar quien puede leer y modificar data de filas individuales y/o columnas.

Por ejemplo, se podría especificar un escenario de autorización donde los empleados pueden ver sus propios datos pero no los de los demás, lo cual se puede lograr usando LBAC.

Un administrador debe configurar el LBAC creando componentes de seguridad de etiquetas, que son objetos usados para representar las condiciones que determinan si un usuario puede acceder a una parte de los datos. Una política de seguridad, compuesta por una o más etiquetas de seguridad, se usa para describir el criterio para determinar quién puede acceder qué data.

Una vez creado, la etiqueta de seguridad se puede asociar con columnas individuales y filas en una tabla para proteger los datos que ahí se encuentran. Un administrador permite a los usuarios acceder a data protegida otorgándoles etiquetas de seguridad. Cuando un usuario intenta acceder a data protegida, la etiqueta de seguridad de ese usuario es comparada con la etiqueta de seguridad que protege los datos.

Un administrador de seguridad también puede otorgar excepciones (exemptions) a los usuarios. Una excepción permite a los usuarios acceder a data protegida que sus etiquetas de seguridad les impide acceder. Juntas, las etiquetas de seguridad y las excepciones se denominan *credenciales LBAC*.

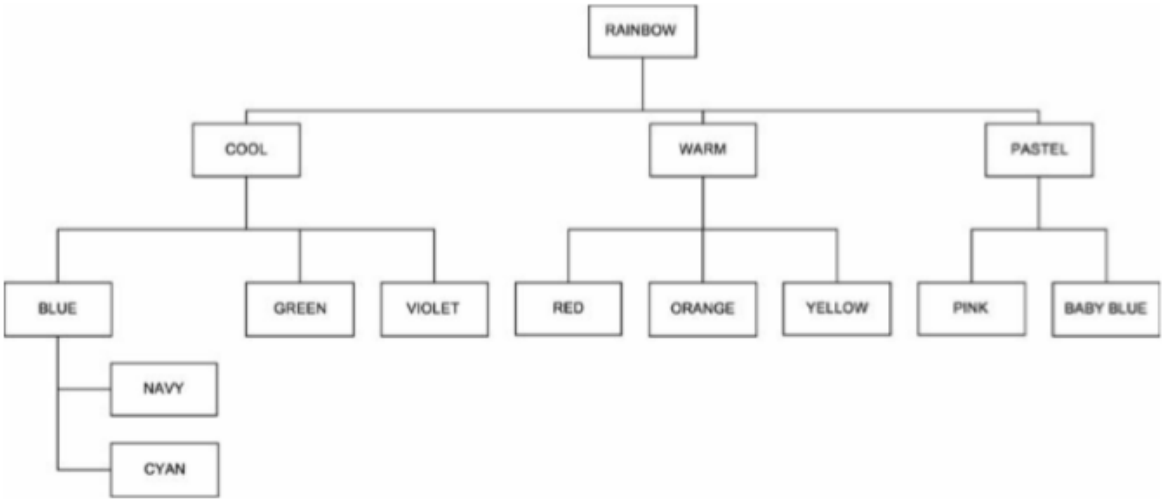


Figure 14.3. Sample LBAC security hierarchy

Para usar LBAC, se necesita añadir un nombre de columna especial que funcione como etiqueta de seguridad.

Asignar una columna de etiqueta de seguridad en una tabla, y llenarla con las especificaciones apropiadas para la jerarquía hace posible si un usuario o no tiene permitido el acceso a una

fila en particular. Entonces, asignar una fila a la etiqueta de seguridad RAINBOW hace la data accesible a cualquiera (ya que está en el nivel más alto de la jerarquía), mientras que asignar una fila a NAVY sería mucho más particular y prohibitiva.

Security Reporting

Una vez otorgados, el DBA necesita monitorear y reportar los privilegios que mantienen los usuarios. La seguridad de la base de datos se mantiene en el catálogo del sistema. El DBA puede usar SQL para recuperar la información necesaria de las tablas apropiadas del catálogo.

Asegúrese de proteger adecuadamente la seguridad del catálogo. Sólo el DBA, SA o administrador de seguridad requiere acceso a la información de seguridad de la BD almacenada en el catálogo.

Authorization Roles and Groups

En adición a otorgar privilegios a usuarios individuales, el DBMS puede proveer la habilidad de asignar a los usuarios:

- Privilegios específicos a un rol, lo cual es después otorgado a otros, o
- Privilegios incorporados de grupos específicos

Roles

Un *rol* se puede usar para otorgar uno o más privilegios preasignados a un usuario. Un rol es, esencialmente, una colección de privilegios. El DBA puede crear un rol y establecer un determinado grupo de privilegios a éste. Luego, el rol se puede asignar a distintos usuarios. Por ejemplo, considere la siguiente secuencia de sentencias:

```
CREATE role MANAGER;
COMMIT;
GRANT select, insert, update, delete on employee to MANAGER;
GRANT select, insert, update, delete on job_title to MANAGER;
GRANT execute on payroll to MANAGER;
COMMIT;
GRANT MANAGER to user1;
COMMIT;
```

El script crea un nuevo rol llamado MANAGER, otorga privilegios de ciertas tablas y procedimientos a ese rol, y luego asigna al usuario user1 el rol MANAGER. Después, a otros usuarios se les puede asignar ese rol.

Groups

Autoridad a nivel de grupo es similar a los roles. Sin embargo, cada DBMS provee grupos incorporados que no se pueden cambiar. Los siguientes grupos son los comunes en la mayoría de DBMSs:

- *System administrator:* . Abreviado SA o SYSADM, es el grupo más poderoso dentro del DBMS. Un usuario al que se le otorgue esta autoridad típicamente puede ejecutar todos los comandos de base de datos, acceder a todas las bases de datos y sus objetos.
- *Database administrator*. DBAADM o DBA, brinda todos los privilegios sobre una base de datos específica, además de la capacidad de acceder, pero no modificar, los datos en una tabla. También, puede eliminar y alterar cualquier objeto en la BD.
- *Database maintenance*. DBMAINT, incluye específicos privilegios para darle mantenimiento a objetos específicos de la BD.
- *Security administrator:* Tiene los privilegios que permiten otorgar y revocar seguridad a lo largo del DBMS. Cualquier actividad relacionada con seguridad puede ser realizada por el administrador de seguridad, incluyendo login y administración de contraseñas, auditoría, configuración de seguridad, como también grants y revokes.
- *Operations control*. OPER o SYSOPR, tiene la autoridad para realizar tareas operacionales de base de datos como backup y recovery.

Limit the Number of SA Users

Una organización debería limitar el número de usuarios a los cuales se les puede asignar el grupo SA (administrador de sistema), ya que éstos son muy poderosos y pueden llevar a cabo casi cualquier tarea de base de datos.

Group-Level Security and Cascading REVOKEs

Dependiendo del grupo, algunos usuarios a quienes se les asignó autoridad a nivel de grupo pueden otorgar privilegios a otros usuarios. Si la autoridad a nivel de grupo es revocada de ese usuario, cualquier otro privilegio que ese usuario otorgó también será revocado.

Antes de revocar la autoridad a nivel de grupo de un usuario, asegúrese de conocer el impacto de los REVOKEs en cascada.

Other Database Security Mechanisms

Using Views for Security

La mayoría de seguridad de base de datos es realizada usando las herramientas de seguridad nativas del DBMS. Sin embargo, es posible simplificar algunos aspectos de seguridad de bases de datos creando vistas para proteger los datos.

Una organización ha implementado una tabla Empleado que guarda información pertinente acerca de los empleados. Mientras que la seguridad de la aplicación se mantiene en este escenario, la seguridad personal no, ya que el usuario puede acceder a detalles personales como el salario.

En lugar de esto, una vista se puede crear que omita la información sensible de la tabla Empleado, mostrando así al usuario solo información básica, no sensible.Eje crear una tabla que omita info sensible:

```
CREATE view emp_all
AS
SELECT first_name, last_name, middle_initial,
       street_address, state, zip_code
FROM   employee;
```

Eliminar (omitir) columnas de una tabla base para crear una vista se conoce como *restricción vertical*.

La restricción vertical usando vistas es una alternativa a especificar columnas mientras se otorgan privilegios de tabla. Además, es más fácil de implementar y administrar.

Las vistas también se pueden usar para proveer seguridad a nivel de filas basado en el contenido de los datos. Esto es conocido como *restricción horizontal* y se implementa codificando una cláusula WHERE apropiada en la vista (para seleccionar solo filas específicas, que cumplan la condición). Eje

```
CREATE view emp_dept20
AS
SELECT first_name, last_name, middle_initial,
       street_address, state, zip_code
FROM   employee
WHERE  deptno = 20;
```

Using Stored Procedures for Security

Los Stored Procedures pueden ser usados para proveer un nivel adicional de seguridad. El privilegio para ejecutar el stored procedure debe ser explícitamente otorgado o revocado.

Los stored procedures se pueden codificar para que accedan solo a filas/columnas específicas de un subconjunto de los datos. Si no se otorgan privilegios de acceso a los datos en la tabla, los usuarios solo podrán acceder a los datos ejecutando los stored procedures, por lo tanto estableciendo un requisito de seguridad.

Logic-Oriented Security

Algunas veces es necesario implementar seguridad basada en algún algoritmo. Por ejemplo, ¿qué pasa si solo un subconjunto de los usuarios puede acceder a los datos a una hora determinada durante el día? Este criterio puede ser codificado en el stored procedure. Cuando el procedure es ejecutado, verifica el usuario y la hora del día antes de permitir el acceso a los datos o tabla.

Encryption

La encriptación de los datos es el proceso donde los datos son transformados usando un algoritmo que los hace ilegibles a cualquiera que no tenga la llave de desencriptación. Sin la llave no se puede acceder a la información.

Existen dos tipos de situaciones donde se puede implementar la encriptación: *data in "transit"* y *data at "rest"*. el cifrado de datos "rest" protege los datos almacenados en la base de datos, mientras que el cifrado de datos "en tránsito" se utiliza para los datos que se transfieren a través de una red

Oracle Virtual Private Database:

proporciona control de acceso a nivel de fila a través de su tecnología Virtual Private Database (VPD). VPD se habilita asociando una o más políticas de seguridad con tablas o vistas. Cuando se accede a la tabla, ya sea directa o indirectamente, la base de datos consultará una función que implementa la política. La política es básicamente un predicado SQL.

Con VPD, un usuario puede recuperar y manipular solo datos que coinciden con la cláusula WHERE en la política

Data at Rest Encryption

Encriptar data at rest "datos en descanso, estáticos" se usa para prohibir el acceso "detrás de escena" no autorizado a los datos. Considere, por ejemplo, una base de datos que contiene un plan militar de alto secreto. Por supuesto, estos datos deben ser protegidos usando los métodos convencionales de seguridad y autorización. Pero ¿qué pasa si los datos son accedidos fuera del control del DBMS? Cuando la data at rest se encripta, aun cuando un hacker gana acceso "detrás de escena" a los datos, sin la llave de desencriptación los datos no significarían nada, serían ilegibles.

Data in Transit Encryption: Encriptar los datos "en tránsito" (cuando navegan por la red) se usa para prohibir el acceso no autorizado a paquetes que circulan por la red. Si los datos se encriptan antes de ser enviados por la red y desencriptados en su destino, los datos se mantienen protegidos a lo largo del viaje por la red. Alguien que intente acceder a los datos de forma no autorizada recibirá solo los datos encriptados que, nuevamente, no significarían nada sin la llave de desencriptación.

Encryption Techniques:

Data Encryption Standard (DES), Triple DES (TDES) y (TDES), and Advanced Encryption Standard (AES): son écnicas y algoritmos de cifrado compatibles con los DBMS modernos

Oracle Transparent Data Encryption:

le permite encriptar columnas de tablas individuales o un espacio de tablas completo. Cuando un usuario inserta datos en una columna cifrada, el cifrado de datos transparente cifra

automáticamente los datos. Cuando los usuarios seleccionan la columna, los datos se descifran automáticamente.

Los datos se cifran con una wallet, que es un archivo del sistema operativo ubicado fuera de la base de datos. La base de datos usa wallet para almacenar la clave de cifrado. se crea usando el comando ALTER SYSTEM. wallet está encriptada y requiere una contraseña como clave de encriptación. Para acceder al contenido de la wallet debe conocer la contraseña.

Inyección de SQL

Otro aspecto de seguridad de base de datos es diseñar la aplicación de forma apropiada con el objetivo de evitar ataques de inyección de SQL. La inyección de SQL es una forma de hackeo Web donde sentencias SQL son especificadas en campos de una forma Web que causa que una aplicación pobremente diseñada muestre información sensible o no autorizada al hacker.

Antes de que un ataque de inyección de SQL funcione, el software de aplicación usado en el sitio Web debe ser vulnerable a un ataque de este tipo. La inyección de SQL depende de programas que no filtran adecuadamente caracteres literales de escape incorporados en sentencias SQL. De esta forma, el SQL es "inyectado" desde la Web a la base de datos, causando la ejecución de la sentencia y el acceso (e incluso modificación) de datos que no se pretendían.

Considere el siguiente caso. Un sitio Web donde se recupera el ID de usuario y contraseña de acuerdo con su email. Donde la consulta a la BD es la siguiente:

```
SELECT userid, password
FROM   uid_pwd_table
WHERE  field = '$EMAIL';
```

La variable '$EMAIL' representa lo que el usuario ingresó en el campo del sitio Web. Un hacker inteligente puede intentar un ataque de inyección de SQL ingresando:

```
anything' OR '1'='1
```

De esta forma, la consulta final se vería de la siguiente:

```
SELECT userid, password
FROM   uid_pwd_table
WHERE  field = 'anything' OR '1'='1';
```

Lo que causaría la devolución de todos los IDs y contraseñas de todos los usuarios de la tabla 'uid_pwd_table', ya que la segunda condición siempre evaluaría a "true".

SQL Injection Prevention

Usar interpretadores de lenguaje de consultas bien diseñados y programar las aplicaciones apropiadamente puede prevenir ataques de inyección SQL. Cuando sea posible, usar SQL estático.

Asegurarse siempre validar la entrada del usuario testeando el tipo, longitud, formato y rango. Testear el contenido de variables string y permitir solo valores esperados para ser procesados. Cualquier entrada (input) que contenga datos binarios, caracteres de escape y caracteres de comentado siempre se deben rechazar.

Evitar la concatenación de entradas del usuario que no hayan sido validadas. La concatenación de hileras de texto es el punto de entrada principal de ataques de inyección de SQL. Además, considere usar stored procedures para validar la entrada del usuario.

Los usuarios de Microsoft SQL Server deben cuidado con el procedimiento Transact-SQL xp_cmdshell. Este procedimiento genera un shell de comandos de Windows y pasa una cadena para su ejecución. Por lo tanto, es completamente posible que un ataque de inyección SQL no solo robe o dañe datos, sino que cargue y ejecute su propio código, pruebe la red e incluso inicie ataques contra otros sitios desde la víctima de la inyección SQL. El uso de xp_cmdshell está deshabilitado de forma predeterminada.

Static versus Dynamic SQL

Usar SQL estático en lugar de SQL dinámico puede mejorar la seguridad de las aplicaciones de bases de datos y la data. SQL estático es altamente programado en la aplicación y no permite que cambie en tiempo de ejecución. SQL dinámico es flexible y puede cambiar en tiempo de ejecución.

The DB2 BIND Command: El comando BIND puede considerarse como un compilador para las instrucciones SQL, lee las instrucciones SQL integradas en el programa de la aplicación y produce una ruta de acceso ejecutable a los datos según lo indicado por las instrucciones SQL que están vinculadas. Una vez que el SQL está estáticamente vinculado, no se puede cambiar sin volver a someterse al proceso BIND.

DB2 se proporciona con las opciones ENABLE y DISABLE, que se pueden especificar para controlar el entorno en el que se puede ejecutar el plan o paquete que se está vinculando

Auditing

La auditoría es una característica del DBMS que permite a los DBAs seguir el rastro del uso de los recursos y privilegios de la base de datos. Cuando la autoría está habilitada, el DBMS produce una pista o rastro de auditoría para cada operación de la BD. Cada operación de BD auditada produce un rastro de información, incluyendo qué objeto de la BD fue impactado, quién hizo la operación, y cuándo.

La auditoría ocurre post-actividad, no prohíbe ningún acceso. Los datos de auditoría ayudan a promover la integridad de los datos al habilitar la detección de *brechas de seguridad*, también conocida como *detección de intrusos*.

External Security:

En adición a la seguridad de base de datos, el DBA debe asegurarse que ciertos recursos usados por el DBMS sean protegidos de ser accesados fuera del control del DBMS.

Cuando se usan mecanismos de seguridad externa para proteger recursos relacionados con la base de datos, el DBA debe enfocarse principalmente en conjuntos de datos. Los conjuntos de datos incluyen:

- Archivos de datos del catálogo del sistema
- Logs activos y archivados
- Conjuntos de datos del usuario para tablespaces
- Conjuntos de datos del usuario para índices
- Archivos de datos de auditoría
- Archivos de seguimiento de rendimiento
- Archivos de programa y script (código fuente y código ejecutable)

El cifrado de datos es una técnica de seguridad que codifica datos legibles en un formato codificado, lo que hace que los archivos sean ilegibles sin la clave de cifrado

Un nivel adicional de protección se puede lograr comprimiendo los datos, sin embargo, esto no es suficiente protección. Además, considere usar software de encriptación de datos, si se encuentra disponible.

Job Scheduling and Security

Muchas organizaciones programan sus tareas para que corran en momentos determinados, y cuando esas tareas involucran el acceso a la base de datos, se debe otorgar una autoridad al *scheduler* (realiza tareas que fueron programadas). Cuando software de scheduling es usado para controlar la sumisión y programación de programas en lote y scripts, el DBA debe determinar la mejor forma de otorgar seguridad de base de datos al scheduler.

No es una muy buena idea otorgar autorización SYSADM al planificador de trabajos (**Job Scheduling**). Hacerlo permitiría que cualquier trabajo realice cualquier tarea de base de datos, creando problemas de seguridad potencialmente graves

Non-DBMS DBA Security

El DBA deberá poseer un nivel bastante alto de autoridad del sistema operativo para realizar el trabajo de administrar y administrar las bases de datos y datos de la organización.

DBMS Fixpacks and Maintenance

El DBMS en sí debe ser parcheado periódicamente para corregir defectos de seguridad y otros errores de software en el código DBMS. Si no se aplican los parches a medida que se

envían desde el proveedor de DBMS a través de fixpacks, descargas y otras secuencias de mantenimiento, pueden causar problemas de seguridad en sus bases de datos.

Oracle Database Patches

Hay dos tipos de parches para los que se deben preparar los DBA: actualizaciones críticas de parches y alertas de seguridad

Critical Patch Updates: son el medio principal para liberar soluciones de seguridad para productos Oracle a clientes con contratos de soporte válidos

Security Alerts: son emitidos por Oracle para correcciones de vulnerabilidades consideradas demasiado críticas para esperar la próxima actualización de parche crítico.