GRUPO DE SEGURIDAD INFORMÁTICA

Fundamentos de la Seguridad Informática

Bases de Datos

Seguridad en Bases de Datos

- Bases de Datos almacenan datos y proveen información a sus usuarios
- Seguridad de BD: protección de datos sensibles y mecanismos que permiten extraer información en forma controlada
- Diferencia con seguridad de SOs: Seguridad de BD controla acceso a la información más que a los datos
- Foco en sujetos que requieren acceso a la base

Plan

- Requerimientos de Protección
- BD Relacionales
- Modelos de seguridad
  - DAC
    - Modelo de SQL
    - Control de acceso usando vistas
  - MAC: Relaciones Multinivel
  - RBAC
- Seguridad de BD Estadísticas
- Privacidad

GRUPO DE SEGURIDAD INFORMÁTICA

Objetivos de Ataque

- Datos: los valores cargados en la base
- Cotas: cotas mínimas o máximas de un valor numérico pueden representar información valiosa
- Resultados negativos: si una BD contiene el número de condenas criminales, la información que una persona en particular no posee 0 condenas es información sensible
- Existencia: la existencia de datos puede ser una información sensible
- Valor probable: capacidad de inferir información a partir del resultado de otras *queries*

Bases de Datos Relacionales

- Una BD relacional es una BD que es percibida por sus usuarios como una colección de tablas (y solamente tablas)
- Una relación *R* es un subconjunto del conjunto D_1 x … x D_n donde D_1,..., D_n son los dominios de n atributos
- Los elementos en la relación son *n*-uplas (v_1,..., v_n) con $v_i \in D_i$; el valor del i-esimo atributo tiene que ser un elemento del dominio D_i.
- Elementos de una tupla son usualmente llamados campos
- Un valor especial *null* indica que un campo no contiene ningún valor

Tipos de Relaciones

- Relaciones base (relaciones reales): relaciones autónomas e identificadas por un nombre, no son derivadas a partir de otras relaciones y tienen su propios datos
- Vistas: relaciones derivadas e identificadas por un nombre, son definidas en términos de otras relaciones, no almacenan datos propios
- *Snapshots*: relaciones derivadas e identificadas por un nombre, son definidas en términos de otras relaciones, almacenan datos propios
- Resultados de *Queries*: pueden o no ser identificadas con un nombre, no persisten an la BD

Claves

- Las tuplas en una relación deben ser unívocamente identificables
- Una clave primaria *K* de una relación *R* debe satisfacer las siguientes propiedades:
  - Unicidad: En todo momento no existen dos tuplas de *R* que tengan el mismo valor para *K*
  - Minimalidad: Si *K* es una clave compuesta, ninguno de sus componentes puede ser omitido sin destruir unicidad
- Toda relación debe tener una clave primaria
- Cuando la clave primaria *K* de una relación es a su vez un atributo de otra, entonces K es una clave foránea de esta última

GRUPO DE SEGURIDAD INFORMÁTICA

Reglas de Integridad

- Regla de Integridad de una Entidad: ningún componente de la clave primaria de una relación puede contener el valor null
- Regla de Integridad Referencial: la BD no puede contener valores inconsistentes de claves foráneas

SQL

- Structured Query Language (SQL): lenguaje estándar para describir formas de recuperar y actualizar la información almacenada en una BD relacional
- Operaciones SQL :
  - SELECT: recupera datos de una relación
  - UPDATE: actualiza campos de una relación
  - DELETE: borra tupla de una relación
  - INSERT: agrega tuplas a una relación

Modelo de Seguridad de SQL

- Fuertemente basado en modelo de autorización de System R
- DAC, basado en
  - usuarios: autenticados durante logon
  - acciones: incluyen SELECT, UPDATE, DELETE e INSERT
  - objetos: tablas, vistas, columnas (atributos) de tablas y vistas
- Usuarios invocan acciones sobre los objetos, el DBMS decide si el acceso está autorizado
- Cuando un objeto es creado se le asigna un propietario
  - Inicialmente sólo el propietario tiene acceso al objeto
  - Para que otros usuarios puedan tener acceso al objeto el propietario les debe otorgar un privilegio

    (grantor, grantee, objeto, acción, grantable)

Otorgando y Revocando Privilegios

- Gestión de privilegios usando GRANT y REVOKE

```
Table Diary(Name,Day,Flight,Status)

  GRANT SELECT, UPDATE (Day,Flight)
     ON TABLE Diary
     TO U1 U2
```

- Revocación selectiva de privilegios

```
  REVOKE UPDATE
     ON TABLE Diary
     FROM U1
```

- Delegación de privilegios usando GRANT option

```
  GRANT SELECT
    ON TABLE Diary
    TO U1
    WITH GRANT OPTION
```

Control de Acceso usando Vistas

- Vistas: relaciones derivadas

```
CREATE VIEW view_name [ ( column [, column ] ... ) ]
    AS subquery
    [ WITH CHECK OPTION ]
```

- Permiten implementar en forma natural control de acceso *content-based*
- Condiciones de acceso descriptas usando una subquery en la definición de la vista

```
CREATE VIEW business_trips AS
    SELECT * FROM Diary
    WHERE Status = `business'
    WITH CHECK OPTION;
```

Ventajas del uso de Vistas

- Permiten definir reglas de control de acceso con un nivel de descripción muy cercano a los requerimientos de los aplicativos
- Permiten expresar y aplicar políticas tanto context-dependent como content-dependent
- Permiten implementar invocación controlada
- Vistas aseguradas pueden reemplazar a etiquetas de seguridad
- Los datos pueden ser fácilmente re-clasificados

Más Ejemplos

```
CREATE VIEW Top_of_the_Class AS
      SELECT * FROM Students
      WHERE Grade < (SELECT Grade FROM Students
                     WHERE Name = current_user());
```

Despliega los estudiantes cuyo promedio de calificaciones es menor al de la persona que está usando la vista

```
CREATE VIEW My_Journeys AS
      SELECT * FROM Diary
      WHERE Customer = current_user());
```

Despliega los viajes reservados por el cliente que está usando la vista

La opción CHECK

- INSERT y UPDATE pueden interferir con control de acceso basado en vistas
- Vistas pueden ser no actualizables debido a que no contengan la información que es necesaria para mantener la integridad de la correspondiente relación (tabla) base
  - Ejemplo: una vista que no contiene la clave primaria de la tabla base no puede ser usada para actualizaciones
- *Blind writes*: updates que sobreescriben una entrada existente

La opción CHECK (II)

La siguiente actualización debe ser permitida?

```
UPDATE business_trips
     SET Status = 'private'
     WHERE Name = 'Alice' AND Day = 'Thu';
```

- Para las vistas definidas con la CHECK OPTION, UPDATE e INSERT sólo pueden efectuar modificaciones sobre la BD que satisfacen la definición de la vista
- Blind writes son posibles sólo si la CHECK option es omitida

Desventajas

- Chequeo de acceso puede volverse complicado y lento
- La corrección de las definiciones de la vista debe ser verificada (realmente capturan la política de seguridad?)
- Completitud y consistencia también debe ser verificada, las vistas pueden colisionar o pueden no llegar a cubrir toda la BD
- Puede llegar a ser difícil determinar quién tiene acceso a un determinada dato. Vistas no son un mecanismo muy adecuado cuando el interés es proteger los ítems de datos más que controlar las acciones de los usuarios

MAC en Bases de Datos

- MAC especifica el acceso que sujetos tienen sobre los objetos basado en una clasificación de seguridad de los sujetos y los objetos
- Este tipo de seguridad es usualmente referenciado como Multilevel Security (MLS)
- DBMSs que satisfacen propiedades de MLS son denominados *multilevel secure database management systems* (MLS/DBMSs)
- La mayoría de los MLS/DBMSs han sido diseñados basados en el modelo de Bell – La Padula (BLP)

Modelo Relacional Multinivel

- El modelo relacional multinivel (MRL) resulta de aplicar el modelo BLP en BD relacionales
- Usualmente son utilizados reticulados need-to-know (clases de acceso)
- Cuestiones
  - Granularidad: a qué elemento de la BD se le aplica la clasificación
  - Restricciones de Integridad

Modelo MRL

- Dada una relación, una clase de acceso puede ser asociada a
  - Toda la relación
  - Cada tupla en la relación
    - Este es el enfoque comúnmente adoptado en DBMSs comerciales
  - Cada valor de atributo de cada tupla de la relación
    - En lo que sigue se considerarán estos casos

Relaciones Multinivel (RML)

- Una relación RML es caracterizada por los siguientes componentes
  - Un esquema invariante de relación $R(A_1,C_1,\ldots,A_n,C_n,TC)$ donde
    - A_i es un atributo sobre un dominio D_i
    - C_i es un atributo de clasification para A_i; cuyo dominio es el conjunto de las clases de acceso que puede ser asociado con los valores de A_i
    - TC es el atributo de clasificación de la tupla

Relaciones Multinivel (RML) (2)

- Una relación sobre R dependiente del estado compuesta de distintas tuplas de la forma (a_1,c_1,…, a_n,c_n, tc), donde
  - a_i es un valor en el dominio D_i
  - c_i es la clase de acceso para a_i
  - tc es la clase de acceso de la tupla, que es determinada como el least upper bound de todos los c_i en la tupla
- Atributos de clasificación no pueden ser nulos

Ejemplo de una relación RML

Nombre	C_Nombre	#Dep	C_#Dep	Salario	C_Salario	TC
JP	Low	Dep1	Low	30k	Low	Low
RM	High	Dep2	High	50k	High	High
NV	Low	Dep1	Low	100k	High	High

Instancias RML

- Una relación dada puede entonces tener instancias con diferentes clases de acceso
- La instancia de clase c de una relación contiene todos los datos que son visibles para sujetos de nivel c (datos dominados por clase c)
- Todos los elementos con clases de acceso mayores a c, o incomparables con c, son enmascarados con valores nulos (o valores ficticios)

Instancia Low

Nombre	C_Nombre	#Dep	C_#Dep	Salario	C_Salario	TC
JP	Low	Dep1	Low	30k	Low	Low
NV	Low	Dep1	Low	Null	Low	Low

Condiciones de corrección de RML

- Las relaciones ML deben satisfacer las siguientes condiciones
  - Para cada tupla en una RML los atributos de la clave primaria deben tener asociados la misma clase de acceso
  - Para cada tupla en una RML la clase de acceso asociada a cada uno de los atributos que no son claves debe dominar a la clase de acceso de la clave primaria

RBAC y DBMSs

- La mayoría de los DBMSs comerciales implementan RBAC
- Sin embargo, en la mayoría de los casos sólo implementan $RBAC_0$ y algunas características de $RBAC_1$

RBAC: Comandos SQL

- `CREATE ROLE role-name`

  `IDENTIFIED BY passw | NOT IDENTIFIED;`

  - Ejemplo:

    ```
    CREATE ROLE teller IDENTIFIED BY
     cashflow;
    ```

- `DROP ROLE role-name;`

RBAC: Comandos SQL (II)

- `GRANT role TO user | role | PUBLIC`

  `[WITH ADMIN OPTION];`
  - Para poder otorgar un rol un usuario debe a su vez tener el privilegio para ese role con la opción `ADMIN`, o si no, el privilegio system `GRANT ANY ROLE`
  - La opción ADMIN le permite al receptor del privilegio poder modificar e incluso “dropear” el role
- Ejemplo: `GRANT teller TO Bob;`

RBAC: Comandos SQL (III)

- El comando grant permite otorgarle privilegios en forma explícita a un rol

```
GRANT select ON Employee TO teller;
```

- El comando SET permite habilitar o deshabilitar roles en una sesión
  - `SET ROLE role-name IDENTIFIED BY passwd;`
  - `SET ROLE ALL [EXCEPT role-name]`
  - `SET ROLE NONE;`

Seguridad de BD Estadística

- BD estadística: información recuperada usando queries estadísticas (agregaciones) sobre atributos de una tabla
- Funciones de agregación en SQL:
  - COUNT: el número de valores en una columna
  - SUM: la suma de los valores de una columa
  - AVG: el promedio de los valores de una columa
  - MAX: el valor máximo de una columna
  - MIN: el valor mínimo de una columna
- Predicado de Query de una query estadística: especifica las tuplas usadas para computar la agregación
- Conjunto Query: tuplas que satisfacen el predicado de query

Desafío de Seguridad

- La BD contiene datos que son individualmente sensibles, acceso directo a algunos de los ítems de la base no es permitido
- Las queries estadísticas son permitidas, pero las mismas van a poder leer ítems de datos individuales
- Es posible entonces inferir información, por lo tanto control de acceso individual no es suficiente
- En una BD estadística la información fluye desde los datos individuales hacia las agregaciones que se computan sobre los mismos
- Desafío: tratar de reducir ese flujo lo máximo posible

Ataques

- Agregación: el nivel de sensibilidad de una agregación computada sobre un grupo de valores puede diferir del nivel de sensibilidad de cada uno de esos valores

- Inferencia: derivación de información sensible a partir de datos no sensibles
  - Ataque Directo: agregación computada sobre un número reducido de datos puede provocar fuga de información de datos individuales
  - Ataque Indirecto: combinar información relativa a varias agregaciones
  - Tracker Attack: un tipo efectivo de ataque indirecto

Relación Estudiantes

Nombre	Sexo	Carrera	Créditos	Promedio
Alma	F	ELEC	80	6.3
Juan	M	COMP	150	5.8
Elena	F	COMP	160	7.0
Jorge	M	CIV	220	7.5
José	M	COMP	80	6.6
Flor	F	CIV	160	8.1
Gabriela	F	ELEC	230	6.8
Horacio	M	COMP	70	5.0
Ignacio	M	CIV	210	7.0

Ataque Directo

```
Q1 : SELECT COUNT(*)
     FROM Estudiantes
     WHERE Sexo = 'F' AND Carrera = 'COMP'
```

Retorna 1

```
Q2 : SELECT AVG(Promedio)
     FROM Estudiantes
     WHERE Sexo = 'F' AND Carrera = 'COMP'
```

Retorna 7.0: promedio para un estudiante (Elena)

- El criterio de selección retorna un conjunto con sólo 1 elemento
- Solución: Exigir que las queries retornen conjuntos de cardinalidad suficientemente grandes.

Ataques Tracker

- Tracker individual para una tupla dada: predicado de query *T* que permite derivar información relativa a esa tupla
- Tracker general : predicado que puede ser usado para encontrar respuestas a cualquier query
- Sea *T* un tracker general y sea *R* un predicado que identifique en forma unívoca la tupla *r* objetivo
- Ejecutar dos queries a la BD con los predicados $R \vee T$ y $R \vee \neg T$; la tupla objetivo *r* es la única usada por las dos queries
- ‘Sumar’ los dos resultados y 'Restar' el resultado de ejecutar una query sobre toda la BD, queda solamente la tupla objetivo

Tracker Attack

```
Q3 : SELECT COUNT(*)
     FROM   Estudiantes
     WHERE  Carrera = 'COMP'

Q4 : SELECT COUNT(*)
     FROM   Estudiantes
     WHERE  Carrera = 'COMP' AND Sexo = 'M'

Q5 : SELECT AVG(Promedio)
     FROM   Estudiantes
     WHERE  Carrera = 'COMP'

Q6 : SELECT AVG(Promedio)
     FROM   Estudiantes
     WHERE  Carrera = 'COMP' AND Sexo = 'M'
```

Retorna 4

Retorna 3

Retorna promedio 6.1

Retorna promedio 5.8

Promedio de Elena:
$4 * 6.1 - 3 * 5.8 = 7.0$

General Tracker

```
Q7 : SELECT SUM(Units)
     FROM   Students
     WHERE  Nombre = 'Flor' OR Carrera = 'CIV'
```

Retorna 590

```
Q8 : SELECT SUM(Units)
     FROM   Students
     WHERE  Nombre = 'Flor' OR NOT (Carrera = 'CIV')
```

Retorna 930

```
Q9 : SELECT SUM(Units)
     FROM   Students
```

Retorna 1360

Flor ha obtenido
(590 + 930) - 1360 = 160 créditos

Contramedidas

- Suprimir información obviamente sensible
- Disfrazar los datos
  - Permutar randómicamente entradas en la BD de tal forma que una query individual de un resultado erróneo aunque el resultado estadísitico sea correcto
  - Agregar perturbaciones al resultado de la query
  - Desventaja: precisión y usabilidad se ve fuertemente reducida
- Mejor diseño del esquema de la base de datos
- “Trackear” lo que el usuario sabe: registrar acciones del usuario en un log de auditoría, análisis de query buscando secuencias sospechosas de queries

Privacidad

- Las organizaciones que almacenan datos personales de sus clientes, como nombre, dirección, edad, número de tárjetas de crédito, deben satisfacer leyes y regulaciones de protección de datos
- Ejemplos:
  - OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data
  - EU Data Protection Directive
  - US: HIPAA (Health Insurance Portability and Accountability Act of 1996)
  - Uruguay: Ley Nº 18.331 - Protección de Datos Personales y Acción de Habeas Data (aprobada en agosto de 2008, regulada en setiembre de 2009)

Principios Básicos de Protección

- Principio de recolección limitada
  - Deberían existir límites para la recolección de datos personales, y los mismos deberían ser obtenidos en forma regulada y legal, con el consentimiento del sujeto involucrado
- Principio de Calidad de Datos
  - Los datos personales deben ser relevantes para el propósitos con el que serán utilizados, y de acuerdo a ese propósito, deberían ser correctos, completos y actualizados

Principios Básicos de Protección (2)

- Principio de Seguridad
  - Los datos personales deben ser protegidos usando mecanismos adecuados de seguridad para proteger a los mismos contra riesgos como pérdida o acceso, destrucción, uso, revelamiento y modificación no autorizados
- Principio de Trazabilidad
  - El controlador de los datos debe ser responsable de aplicar y satisfacer las medidas que implementan los anteriores principios

Bibliografía y Referencias

- **D. Gollman,** *Computer Security***,** Wiley, 2006.
- **P. Griffiths, B. Wade,** *An Authorization Mechanism for a Relational Database System***,** ACM Transactions on Database Systems, Vol. 1, Nro. 3, 1976.
- **E. Bertino, P. Samarati, S. Jajodia**, *An Extended Authorization Model for Relational Databases,* IEEE Transactions on Knowledge and Data Engineering, Vol. 9, Nro. 1, 1997.