TDDD17
Information Security

Topic: Database Security

Olaf Hartig

olaf.hartig@liu.se

Acknowledgement: Many of the slides in this slide set are adaptations from slides made available for the database textbook by Elmasri and Navathe.

LINKÖPING UNIVERSITY

Before we begin …

… a reminder of database-related terminology

- **Data**: known facts that can be recorded
  and that have implicit meaning
- **Database**: logically coherent collection of related data
  - Built for a specific purpose
  - Represents some aspects of the real world
- **Database management system** (**DBMS**): collection of computer programs to create and maintain a database
  - Protects DB against unauthorized access and manipulation
  - Examples of DBMSs: IBM’s DB2, Microsoft’s SQL Server,
    Oracle, MySQL, PostgreSQL

… and some more terminology …

SQL (Structured Query Language)

- Most prevalent database language
- Commands for defining databases (i.e., their structure)
- Commands for manipulating the data
  - INSERT, UPDATE, DELETE
- Commands for expressing **queries** (i.e., questions to be answered based in the data)
  - SELECT
- **SQL databases** represent data in a tabular form

… and last but not least …

Relational Data Model (formal foundation of SQL)

- **Relational database** is a collection of relations
- **Schema** describes the relations
  - Relation names, attribute names, attribute domains (types)
  - Integrity constraints
- **Instance** (also called state) is a set of **tuples** for each relation that represent the current content

Relation Name

Attributes

STUDENT

Tuples

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Example from "Fundamentals of Database Systems" by Elmasri and Navathe, Addison Wesley.

Introduction to DB Security

What are the threads?

- Loss of integrity: improper modification of data
  - e.g., students changing their grades
- Loss of confidentiality: unauthorized disclosure of data
  - e.g., student learns other students' grades
- Loss of availability: unavailability of database objects to authorized programs and people
  - e.g., students are denied seeing their own grades
  - “denial of service attack”

Control Measures to Provide DB Security

- Access control
  - Limiting access to the database (or parts thereof)
  - Requires authentication (e.g., through login and password)
  - Usually with auditing (i.e., logging DB operations by each user)
- Inference control
  - Preventing deductions about database content
  - Summary data without ability to determine individuals' data
- Flow control
  - Preventing information from reaching unauthorized users
- Data encryption
  - Protecting sensitive data (e.g., when transmitted over network)
  - Making information unintelligible unless authorized
  - Making changes traceable to source

Access Control in a Database System

- Security policy specifies who is authorized to do what in the system
- DBMS provides access control mechanisms to help implement a security policy
- Two complementary types of such mechanisms:
  - Discretionary access control
  - Mandatory access control

Discretionary Access Control

Idea and Related Concepts

- Idea: achieve access control based on
  1. privileges (specific rights for tables, columns, etc.), and
  2. a mechanism for granting and revoking such privileges
- Authorization administration policy specifies how granting and revoking is organized
  - i.e., who may grant / revoke
  - *Centralized administration*: only some privileged users
  - *Ownership-based administration*: creator of the ob
- Administration delegation: if authorized to do so, a user may assign others the right to grant / revoke

Discretionary Access Control in SQL

- Simple examples:
  - to allow user Alice to query the table called Student
    GRANT SELECT **ON** Student **TO** Alice
  - to allow Alice to delete from the Student table
    GRANT DELETE **ON** Student **TO** Alice
  - revoke the previous privilege
    REVOKE DELETE **ON** Student **FROM** Alice
  - to allow Alice to modify any value in Employee
    GRANT UPDATE **ON** Employee **TO** Alice
  - to allow Bob to modify Salary values in Employee
    GRANT UPDATE **ON** Employee(Salary) **TO** Bob

Discretionary Access Control in SQL (cont'd)

GRANT *privileges* **ON** *objects* **TO** *users*

REVOKE *privileges* **ON** *objects* **FROM** *users*

- Possible privileges:
  - SELECT
  - INSERT (may be restricted to specific attributes)
  - UPDATE (may be restricted to specific attributes)
  - DELETE
  - REFERENCES (may be restricted to specific attributes)
- Possible objects:
  - Tables
  - Views
  - Specific attributes (for INSERT, UPDATE, REFERENCES)

What are Views?

- A **virtual** table **derived** from other (possibly virtual) tables, i.e. always up-to-date

```
CREATE VIEW research_colleagues_view AS
  SELECT Fname, Lname, Email
   FROM    EMPLOYEE
   WHERE Dept = 'Research';

CREATE VIEW dept_view AS
 SELECT Dept, COUNT(*) AS C, AVG(Salary) AS S
  FROM    EMPLOYEE
  GROUP BY Dept;
```

Discretionary Access Control in SQL (cont'd)

GRANT *privileges* **ON** *objects* **TO** *users* **[WITH GRANT OPTION]**

REVOKE [GRANT OPTION FOR] *privileges* **ON** *objects* **FROM** *users*

- WITH GRANT OPTION allows users to pass on privilege (with or without passing on grant option)
  - When a privilege is revoked from user *X*, it is also revoked from all users who were granted this privilege *solely* from *X*

Example

- Assume we do

GRANT UPDATE **ON** Emp **TO** Alice

GRANT UPDATE **ON** Emp **TO** Bob **WITH GRANT OPTION**

- Next, Bob does

GRANT UPDATE **ON** Emp **TO** Alice, Eve

- Now, Bob, Alice, and Eve have the privilege
- Assume we now do

REVOKE UPDATE **ON** Emp **FROM** Alice

- Alice still has the privilege (thanks to Bob)
- Let's do

REVOKE UPDATE **ON** Emp **FROM** Bob

- Now, neither of them has the privilege anymore

Granularity of Privileges in SQL

- Seen so far, object-level privileges
  - Objects: tables, views, attributes
  - SQL does not support tuple-specific privileges

- System-level privileges
  - CREATE / ALTER / DROP tables or views
  - Creator of an object gets all (object-level) permissions on that object
  - Not supported by standard SQL but by DBMS-specific extensions of SQL

Mandatory Access Control

Idea

- Achieve access control based on system-wide policies that cannot be changed by individual users
- Basis: partially ordered set of security classes
  - e.g., TopSecrect > Secret > Confidential > Unclassified
- DB objects (e.g., tables, columns, rows) are assigned such a class
- Subjects (users, programs) are assigned a clearance for such a class
- Subject's clearance must match class of object

Bell-LaPadula Model

- Rule 1 (no read-up): subject *S* can read object *O* only if clearance(*S*) ≥ class(*O*)
  - e.g., reading secret data requires at least *secret* clearance
  - Goal: protect classified data
- Rule 2 (no write-down): subject *S* can write object *O* only if clearance(*S*) ≤ class(*O*)
  - e.g., person with *confidential* clearance cannot write *unclassified* object
  - Goal: flow control (information never flows from a higher to a lower class)

Trojan Horse Attack

- Assume **discretionary access control**
- Suppose user *Bob* has privileges to read a secret table *T*
- User *Mallory* wants to see the data in *T*
  (but does not have the privileges to do so)
- *Mallory* creates a table *T'* and gives INSERT privileges to *Bob*
- *Mallory* tricks *Bob* into copying data from *T* to *T'* (e.g., by extending the “functionality” of a program used by *Bob*)
- *Mallory* can then see the data that comes from *T*

Trojan Horse Attack

- Let's try to use **mandatory access control** instead
- Suppose user *Bob* has privileges to read a secret table *T*

  clearance(*Bob*) = *secret*

- User *Mallory* wants to see the data in *T*
  (but does not have the privileges to do so)

  clearance(*Mallory*) < *secret*

- *Mallory* creates a table *T'* and gives INSERT privileges to *Bob*

  class(*T'*) := clearance(*Mallory*), i.e., class(*T'*) < *secret*

- *Mallory* tricks *Bob* into copying data from *T* to *T'* (e.g., by extending the "functionality" of a program used by *Bob*)

  → Writing to T' fails because clearance(*Bob*) $\not\leq$ class(*T'*)

- ~~*Mallory* can then see the data that comes from *T*~~

Summary

Summary

- Three main security objectives
  - Secrecy (confidentiality)
  - Integrity
  - Availability
- Discretionary access control
  - based on notion of privileges
  - GRANT and REVOKE
  - susceptible to trojan horse attack
- Mandatory access control
  - based on notion of security classes
  - not widely supported

www.liu.se

LINKÖPING UNIVERSITY