Security and Authorization

CS430/630
Lecture 18

Definitions
- Security policy
  - specifies who is authorized to do what
- Security mechanism
  - allows to enforce a chosen security policy
- Terminology
  - Users = Subjects or Principals
  - Data = Objects
- Two important functions needed to achieve security
  - Authentication (AuthN)
  - Authorization (AuthZ)

Authentication
- Establishing the identity of the user, or who the user is
- Subjects (users) present authentication credentials
  - Username/Password combination – “what user knows”
  - Digital certificates (cryptographic tokens) – “what user has”
  - Biometrics – “what user is”
- Some credential types stronger than others
  - For high-security applications, multi-factor authentication
  - E.g., password + fingerprint

Authorization
- Once we know who the user is, what can s/he access?
  - What objects (data) the subjects is allowed access to?
  - What kind of operations is the subject allowed to perform?
    - Read-only, modify, append
  - Authorization also referred to as access control
- Two main categories of access control
  - Discretionary: object owner decides authorization policy for its
    objects (Unix system)
  - Mandatory: system-wide rules that dictate who gets to access what
    (multi-level security, Bell-LaPadula)

Discretionary Access Control
- Based on the concept of access rights or privileges
  - Privileges for objects (tables and views)
  - Mechanisms for granting and revoking privileges
- Object creator automatically gets all privileges on it
  - DBMS keeps track of who subsequently gains and loses privileges
  - DBMS ensures that only requests from users who have the necessary privileges (at the time the request is issued) are allowed

GRANT Command
- The following privileges can be specified:
  - SELECT
  - can read all columns
  - including those added later via ALTER TABLE command
  - INSERT (col-name)
  - can insert tuples with non-null or non-default values in this column
  - INSERT means same right with respect to all columns
  - DELETE
  - can delete tuples
  - REFERENCES (col-name)
  - can define foreign keys (in other tables) that refer to this column
GRANT Command (contd)

- If a privilege is granted with **GRANT OPTION**, the grantee can pass privilege on to other users
- Special **ALL PRIVILEGES** privilege
- Only owner can execute CREATE, ALTER, and DROP

Examples

**GRANT INSERT, SELECT ON Sailors TO Horatio**
- Horatio can query Sailors or insert tuples into it

**GRANT DELETE ON Sailors TO Yuppy WITH GRANT OPTION**
- Yuppy can delete tuples, and also authorize others to do so

**GRANT INSERT (rating) ON Sailors TO Dustin**
- Dustin can insert (only) the rating field of Sailors tuples

REVOKE Command

- **REVOKE**
- Revokes privileges
- **CASCADE**: when a privilege is revoked from X, it is also revoked from all users who got it solely from X
  - Privilege is said to be **ABANDONED**
  - A graph with the granting relationship is maintained
- **RESTRICT**: if revoke causes some privilege to be abandoned, it is **NOT** executed

Authorization Graph

- Keeps track of active authorization on objects
  - Each authorization ID (user) corresponds to a node
  - Granting a privilege adds labeled edge to graph
  - Removing privilege deletes one or more edges from graph
  - Special “**System**” node that originates all privileges
  - Note: it is possible to have multiple edges between same pair of nodes (with same direction)!

- How to determine if access is allowed for an ID?
  - There must be a path from System to that ID formed of privileges equal (or stronger) than the one required

Authorization Graph

- Joe: **CREATE TABLE** ...
- **Authorization Graph**

Authorization Graph

- Joe: **GRANT SELECT ON T TO Art WITH GRANT OPTION**
Art: GRANT SELECT ON T TO Bob WITH GRANT OPTION

Authorization Graph

Joe → Art: SELECT, Yes
Art → Bob: SELECT, Yes
Cal

Bob: GRANT SELECT ON T TO Art WITH GRANT OPTION

Authorization Graph

Joe → Art: SELECT, Yes
Art → Bob: SELECT, Yes
Cal

Joe: GRANT SELECT ON T TO Cal WITH GRANT OPTION

Authorization Graph

Joe → Art: SELECT, Yes
Art → Bob: SELECT, Yes
Cal

Cal: GRANT SELECT ON T TO Bob WITH GRANT OPTION

Authorization Graph

Joe → Art: SELECT, Yes
Art → Bob: SELECT, Yes
Cal

Joe: REVOKE SELECT on T FROM Art CASCADE

Authorization Graph

Joe → Art: SELECT, Yes
Art → Bob: SELECT, Yes
Cal

Art, Bob can still access T!
No “temporal order” memorized
Security at the Level of a Field!

- Can create a view that only returns one field of one tuple
  - Then grant access to that view accordingly
- Allows for arbitrary granularity of control, but:
  - Tidious to specify and maintain policies
  - Performance is unacceptable
    - Too many view creations and look-ups
- Another solution
  - Attach labels to subjects and objects
  - Create rules of access based on labels

Mandatory Access Control

- Based on system-wide policies that cannot be changed by individual users (even if they own objects)
  - Each DB object is assigned a security class
  - Each subject (user or user program) is assigned a clearance for a security class
  - Rules based on security classes and clearances govern who can read/write which objects.
- Many commercial systems do not support mandatory access control
  - Some specialized versions do
    - e.g., those used in military applications
Bell-LaPadula Model

- Security classes:
  - Top secret (TS)
  - Secret (S)
  - Confidential (C)
  - Unclassified (U):
    - TS > S > C > U

- Each object (O) and subject (S) is assigned a class
  - S can read O only if class(S) >= class(O) (Simple Security Property or No Read Up)
  - S can write O only if class(S) <= class(O) (^-Property or No Write Down)

Intuition

- Idea is to ensure that information can never flow from a higher to a lower security level
- The mandatory access control rules are applied in addition to any discretionary controls that are in effect