Security and Authorization

Chapter 21

Introduction to DB Security

- Secrecy: Users should not be able to see things they are not supposed to.
  - E.g., A student can't see other students' grades.
- Integrity: Users should not be able to modify things they are not supposed to.
  - E.g., Only instructors can assign grades.
- Availability: Users should be able to see and modify things they are allowed to.

Access Controls

- A security policy specifies who is authorized to do what.
- A security mechanism allows us to enforce a chosen security policy.
- Two main mechanisms at the DBMS level:
  - Discretionary access control
  - Mandatory access control

Discretionary Access Control

- Based on the concept of access rights or privileges for objects (tables and views), and mechanisms for giving users privileges (and revoking privileges).
- Creator of a table or a view automatically gets all privileges on it.
  - DMBS keeps track of who subsequently gains and loses privileges, and ensures that only requests from users who have the necessary privileges (at the time the request is issued) are allowed.

GRANT Command

```
GRANT privileges ON object TO users [WITH GRANT OPTION]
```

- 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.
- If a user has a privilege with the GRANT OPTION, can pass privilege on to other users (with or without passing on the GRANT OPTION).
- Only owner can execute CREATE, ALTER, and DROP.

GRANT and REVOKE of Privileges

- 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 UPDATE (*rating*) ON Sailors TO Dustin
  - Dustin can update (only) the *rating* field of Sailors tuples.
- GRANT SELECT ON ActiveSailors TO Guppy, Yuppy
  - This does NOT allow the 'uppies to query Sailors directly!
- REVOKE: When a privilege is revoked from X, it is also revoked from all users who got it *solely* from X.

GRANT/REVOKE on Views

- If the creator of a view loses the SELECT privilege on an underlying table, the view is dropped!
- If the creator of a view loses a privilege held with the grant option on an underlying table, (s)he loses the privilege on the view as well; so do users who were granted that privilege on the view!

Views and Security

- Views can be used to present necessary information (or a summary), while hiding details in underlying relation(s).
  - Given ActiveSailors, but not Sailors or Reserves, we can find sailors who have a reservation, but not the *bid*'s of boats that have been reserved.
- Creator of view has a privilege on the view if (s)he has the privilege on all underlying tables.
- Together with GRANT/REVOKE commands, views are a very powerful access control tool.

Role-Based Authorization

- In SQL-92, privileges are actually assigned to authorization ids, which can denote a single user or a group of users.
- In SQL:1999 (and in many current systems), privileges are assigned to roles.
  - Roles can then be granted to users and to other roles.
  - Reflects how real organizations work.
  - Illustrates how standards often catch up with "de facto" standards embodied in popular systems.

Security to the Level of a Field!

- Can create a view that only returns one field of one tuple. (How?)
- Then grant access to that view accordingly.
- Allows for *arbitrary* granularity of control, *but*:
  - Clumsy to specify, though this can be hidden under a good UI
  - Performance is unacceptable if we need to define field-granularity access frequently. (Too many view creations and look-ups.)

Internet-Oriented Security

- Key Issues: User authentication and trust.
  - When DB must be accessed from a secure location, password-based schemes are usually adequate.
- For access over an external network, trust is hard to achieve.
  - If someone with Sam's credit card wants to buy from you, how can you be sure it is not someone who stole his card?
  - How can Sam be sure that the screen for entering his credit card information is indeed yours, and not some rogue site spoofing you (to steal such information)? How can he be sure that sensitive information is not "sniffed" while it is being sent over the network to you?
- *Encryption* is a technique used to address these issues.

Encryption

- "Masks" data for secure transmission or storage
  - Encrypt(data, encryption key) = encrypted data
  - Decrypt(encrypted data, decryption key) = original data
  - Without decryption key, the encrypted data is meaningless gibberish
- Symmetric Encryption:
  - Encryption key = decryption key; all authorized users know decryption key (a weakness).
  - DES, used since 1977, has 56-bit key; AES has 128-bit (optionally, 192-bit or 256-bit) key
- Public-Key Encryption: Each user has two keys:
  - User's public encryption key: Known to all
  - Decryption key: Known only to this user
  - Used in RSA scheme (Turing Award!)

RSA Public-Key Encryption

- Let the data be an integer I
- Choose a large (>> I) integer $L = p * q$
  - p, q are large, say 1024-bit, distinct prime numbers
- Encryption: Choose a random number $1 < e < L$ that is relatively prime to $(p-1) * (q-1)$
  - Encrypted data $S = I^{e} \bmod L$
- Decryption key d: Chosen so that
  - $d * e = 1 \bmod ((p-1) * (q-1))$
  - We can then show that $I = S^{d} \bmod L$
- It turns out that the roles of e and d can be reversed; so they are simply called the public and private keys

Certifying Servers: SSL, SET

- If Amazon distributes their public key, Sam's browser will encrypt his order using it.
  - So, only Amazon can decipher the order, since no one else has Amazon's private key.
- But how can Sam (or his browser) know that the public key for Amazon is genuine? The SSL protocol covers this.
  - Amazon contracts with, say, Verisign, to issue a certificate <Verisign, Amazon, amazon.com, public-key>
  - This certificate is stored in encrypted form, encrypted with Verisign's *private* key, known only to Verisign.
  - Verisign's public key is known to all browsers, which can therefore decrypt the certificate and obtain Amazon's public key, and be confident that it is genuine.
  - The browser then generates a temporary *session key,* encodes it using Amazon's public key, and sends it to Amazon.
  - All subsequent msgs between the browser and Amazon are encoded using symmetric encryption (e.g., DES), which is more efficient than public-key encryption.
- What if Sam doesn't trust Amazon with his credit card information?
  - Secure Electronic Transaction protocol: 3-way communication between Amazon, Sam, and a trusted server, e.g., Visa.

Authenticating Users

- Amazon can simply use password authentication, i.e., ask Sam to log into his Amazon account.
  - Done after SSL is used to establish a session key, so that the transmission of the password is secure!
  - Amazon is still at risk if Sam's card is stolen and his password is hacked. Business risk …
- Digital Signatures:
  - Sam encrypts the order using his *private* key, then encrypts the result using Amazon's public key.
  - Amazon decrypts the msg with their private key, and then decrypts the result using Sam's public key, which yields the original order!
  - Exploits interchangeability of public/private keys for encryption/decryption
  - Now, no one can forge Sam's order, and Sam cannot claim that someone else forged the order.

Mandatory Access Control

- Based on system-wide policies that cannot be changed by individual users.
  - 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.
- Most commercial systems do not support mandatory access control. Versions of some DBMSs do support it; used for specialized (e.g., military) applications.

Why Mandatory Control?

- Discretionary control has some flaws, e.g., the *Trojan horse* problem:
  - Dick creates Horsie and gives INSERT privileges to Justin (who doesn't know about this).
  - Dick modifes the code of an application program used by Justin to additionally write some secret data to table Horsie.
  - Now, Justin can see the secret info.
- The modification of the code is beyond the DBMSs control, but it can try and prevent the use of the database as a channel for secret information.

Bell-LaPadula Model

- Objects (e.g., tables, views, tuples)
- Subjects (e.g., users, user programs)
- Security classes:
  - Top secret (TS), secret (S), confidential (C), unclassified (U): TS > S> C > U
- Each object and subject is assigned a class.
  - Subject S can read object O only if class(S) >= class(O) (Simple Security Property)
  - Subject S can write object O only if class(S) <= class(O) (*-Property)

Intuition

- Idea is to ensure that information can never flow from a higher to a lower security level.
- E.g., If Dick has security class C, Justin has class S, and the secret table has class S:
  - Dick's table, Horsie, has Dick's clearance, C.
  - Justin's application has his clearance, S.
  - So, the program cannot write into table Horsie.
- The mandatory access control rules are applied in addition to any discretionary controls that are in effect.

Multilevel Relations

<u>bid</u>	bname	color	class
101	Salsa	Red	S
102	Pinto	Brown	C

- Users with S and TS clearance will see both rows; a user with C will only see the 2nd row; a user with U will see no rows.
- If user with C tries to insert <101,Pasta,Blue,C>:
  - Allowing insertion violates key constraint
  - Disallowing insertion tells user that there is another object with key 101 that has a class > C!
  - Problem resolved by treating class field as part of key.

Statistical DB Security

- Statistical DB: Contains information about individuals, but allows only aggregate queries (e.g., average age, rather than Joe's age).
- New problem: It may be possible to infer some secret information!
  - E.g., If I know Joe is the oldest sailor, I can ask "How many sailors are older than X?" for different values of X until I get the answer 1; this allows me to infer Joe's age.
- Idea: Insist that each query must involve at least N rows, for some N. Will this work? (No!)

Why Minimum N is Not Enough

- By asking "How many sailors older than X?" until the system rejects the query, can identify a set of N sailors, including Joe, that are older than X; let X=55 at this point.
- Next, ask "What is the sum of ages of sailors older than X?" Let result be S1.
- Next, ask "What is sum of ages of sailors other than Joe who are older than X, plus my age?" Let result be S2.
- S1-S2 is Joe's age!

Summary

- Three main security objectives: secrecy, integrity, availability.
- DB admin is responsible for overall security.
  - Designs security policy, maintains an audit trail, or history of users' accesses to DB.
- Two main approaches to DBMS security: discretionary and mandatory access control.
  - Discretionary control based on notion of privileges.
  - Mandatory control based on notion of security classes.
- Statistical DBs try to protect individual data by supporting only aggregate queries, but often, individual information can be inferred.