PURDUE
UNIVERSITY

CS34800 Information Retrieval

Database Security

Prof. Elisa Bertino

18 November 2016

Indiana Center for Database Systems

CERIAS Center for Education and Research in Information Assurance and Security

1

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.

Access Control in Commercial DBMSs

- Most commercial systems adopt DAC
- Current discretionary authorization models for relational DBMS are based on the System R authorization model [Griffiths and Wade76]
- It is based on ownership administration with administration delegation

The System R Authorization Model

- Objects to be protected are tables and views
- Privileges include: *select*, *update*, *insert*, *delete*, *drop*, *index* (only for tables), *alter* (only for tables)
- Groups are supported, whereas roles are not
- Privileges can be granted with the GRANT OPTION

The System R - Delegation

- Privilege delegation is supported through the *grant option*: if a privilege is granted with the grant option, the user receiving it can not only exercise the privilege, but can also grant it to other users
- A user can only grant a privilege on a given relation if he/she is the table owner or if he/she has received the privilege with grant option

Grant operation

GRANT *PrivilegeList*| ALL[PRIVILEGES]
ON *Relation* | *View*
TO *UserList* | PUBLIC
  [WITH GRANT OPTION]

- it is possible to grant privileges on both relations and views
- privileges apply to entire relations (or views)
- for the update privilege, one needs to specify the columns to which it applies

Grant operation

- The authorization catalogs keep track for each users of the privileges the user possesses and of the ones that the user can delegate
- Whenever a user *U* executes a Grant operation, the system intersects the privileges that *U* can delegate with the set of privileges specified in the command
- If the intersection is empty, the command is not executed

Grant operation

Bob:

- is the owner of Employee table
- He thus has: Select, Insert, Update, Delete all with the Grant privileges on table Employee

Grant operation - example

1. Bob: GRANT select, insert ON Employee TO Jim WITH GRANT OPTION;
2. Bob: GRANT select ON Employee TO Ann WITH GRANT OPTION;
3. Bob: GRANT insert ON Employee TO Ann;
4. Jim: GRANT update ON Bob.Employee TO Tim WITH GRANT OPTION;
5. Ann: GRANT select, insert ON Bob.Employee TO Tim;

Grant operation - example

- The first three GRANT commands are fully executed (Bob is the owner of the table)
- The fourth command is not executed, because Jim does not have the update privilege on the table
- The fifth command is partially executed; Ann has the select and insert but she does not have the grant option for the insert --> Tim only receives the select privilege

Revoke operation

REVOKE *PrivilegeList*| ALL[PRIVILEGES]
ON *Relation* | *View*
FROM *UserList* | PUBLIC

- a user can only revoke the privileges he/she has granted; <u>it is not possible to revoke the grant option only</u>
- upon execution of a revoke operation, the user from whom the privileges have been revoked loses these privileges, unless (s)he has them from another user *independent* from the one that has executed the revoke

Revoke operation - example

- Bob: GRANT select ON Employee TO Jim WITH GRANT OPTION;
- Bob: GRANT select ON Employee TO Ann WITH GRANT OPTION;
- Jim: GRANT select ON Bob.Employee TO Tim;
- Ann: GRANT select ON Bob.Employee TO Tim;
- Jim: REVOKE select ON Bob.Employee FROM Tim;
- Tim continues to hold the select privilege on table Employee after the revoke operation, since he has independently obtained such privilege from Ann.

Revoke operation - example

- Bob: GRANT *select ON Employee* TO Jim WITH GRANT OPTION;

The grantor

The permission

The grantee

Revoke operation - example

- Bob: GRANT *select ON Employee* TO Jim WITH GRANT OPTION;

The grantor

The permission

The grantee

Time	Grantor	Privilege	Grantee
10	**BOB**	**P1+G**	**JIM**

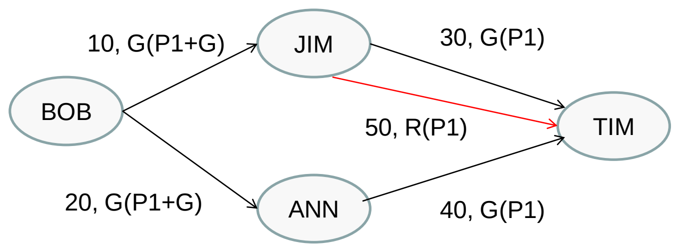
Revoke operation - example

- Bob: GRANT select ON Employee TO Jim WITH GRANT OPTION;
- Bob: GRANT select ON Employee TO Ann WITH GRANT OPTION;
- Jim: GRANT select ON Bob.Employee TO Tim;
- Ann: GRANT select ON Bob.Employee TO Tim;
- Jim: REVOKE select ON Bob.Employee FROM Tim;
- Tim continues to hold the select privilege on table Employee after the revoke operation, since he has independently obtained such privilege from Ann.

Revoke operation *tabular representation*

Time	Grantor/ Revoker	Privilege	Grantee/ Revokee
10	**BOB**	**G(P1+G)**	**JIM**
20	**BOB**	**G(P1+G)**	**ANN**
30	**JIM**	**G(P1)**	**TIM**
40	**ANN**	**G(P1)**	**TIM**
50	**JIM**	**R(P1)**	**TIM**

Revoke operation
graph representation



Revoke operations

- Recursive revocation: whenever a user revokes an authorization on a table from another user, all the authorizations that the revokee had granted because of the revoked authorization are removed
- The revocation is iteratively applied to all the subjects that received the access authorization from the revokee

Recursive revoke

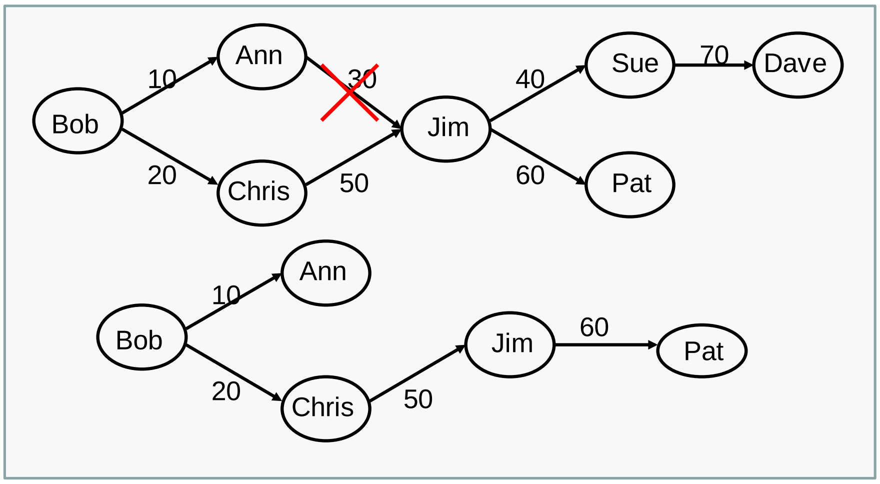
- Let G_1, …., G_n be a sequence of grant operations with a single privilege on the same relations, such that i,k = 1,…., n, if i<k, then G_i is executed before G_k. Let R_i be the revoke operation for the privilege granted with operation G_i.
- The semantics of the recursive revoke requires that the state of the authorization system after the execution of the sequence

  G_1, …., G_n , R_i

  be identical to the state that one would have after the execution of the sequence

  G_1, …., G_{i-1}, G_{i+1} , …., G_n

Recursive Revocation with timestamp



PURDUE
UNIVERSITY

CS34800 Information Retrieval

Database Security

Prof. Chris Clifton

21 November 2016

CERIAS
Center for Education and Research in Information Assurance and Security

Indiana Center for Database Systems

30

PURDUE
UNIVERSITY

Views and content-based authorization

- Views are a mechanism commonly used to support content-based access control in RDBMS
- Content-based access authorizations should be specified in terms of predicates
- Only the tuples of a relation verifying a given predicate are considered as the protected objects of the authorization

Views and content-based authorization

- The approach to support content-based access control in RDBMS can be summarized as follows:
  - Define a view containing the predicates to select the tuples to be returned to a given subject S
  - Grant S the select privilge on the view, and <u>not</u> on the underlying table

Views and content-based authorization

- Example: suppose we want authorize user Ann to access only the employees whose salary is lower than 20000. Steps:

```
- CREATE VIEW Vemp AS
  SELECT * FROM Employee
  WHERE Salary < 20000;

- GRANT Select ON Vemp TO Ann;
```

Views and content-based authorization

- Queries against views are transformed through the *view composition* in queries against base tables
- The view composition operation combines in AND the predicates specified in the query on the view with the predicates which are part of the view definition

Views and content-based authorization

```
Ann:   SELECT * FROM Vemp
        WHERE Job = ‘Programmer’;
```

Query after view composition:

```
SELECT * FROM Employee
        WHERE Salary < 20000 AND
         Job = ‘Programmer’;
```

Steps in Query Processing

- Parsing
- Catalog lookup
- Authorization checking
- View Composition
- Query optimization

- Note that authorization is performed before view composition; therefore, authorization checking is against the views used in the query and not against the base tables used in these views

Views and content-based authorization

- Views can also be useful to grant select privileges on specific columns: we only need to define a view as projection on the columns on which we want to give privileges
- Views can also be used to grant privileges on simple statistics calculated on data (such as AVG, SUM,..)

Authorizations on views

- The user creating a view is called the *view definer*
- The privileges that the view definer gets on the view depend from:
  - The view semantics, that is, its definition in terms of the base relation(s)
  - The authorizations that the definers has on the base table

Authorizations on views

- The view definer does not receive privileges corresponding to operations that cannot be executed on the view
- For example, `alter` and `index` do not apply to views

Authorizations on views

- Consider the following view:

```
Bob: CREATE VIEW V1 (Emp#,
    Total_Sal)
   AS SELECT Emp#, Salary + Bonus
   FROM Employee WHERE
   Job ='Programmer';
```

The update operation is not defined on column Total_Sal of the view; therefore, Bob will not receive the update authorization on such column

Authorizations on views

- Basically, to determine the privileges that the view definer has on the view, the system needs to intersect the set of privileges that the view definer has on the base tables with the set of privileges corresponding to the operations that can be performed on the view

Authorizations on views example

- Consider relation Employee and assume Bob is the creator of Employee
- Consider the following sequence of commands:
  - Bob: GRANT Select, Insert, Update ON Employee to Tim;
  - Tim: CREATE VIEW V1 AS SELECT Emp#, Salary FROM Employee;
  - Tim: CREATE VIEW V2 (Emp#, Annual_Salary) AS SELECT Emp#, Salary*12 FROM Employee;

Authorizations on views example

- Tim can exercise on V1 all privileges he has on relation Employee, that is, Select, Insert, Update
- By contrast, Tim can exercise on V2 only the privileges of Select and Update on column Emp#;

Authorizations on views

- It is possible to grant authorizations on a view: the privileges that a user can grant are those that he/she owns with grant option on the base tables
- Example: user Tim cannot grant any authorization on views V1 and V2 he has defined, because he does not have the authorizations with grant option on the base table

Authorizations on views example

- Consider the following sequence of commands:
  - Bob: GRANT Select ON Employee TO Tim WITH GRANT OPTION;
  - Bob: GRANT Update, Insert ON Employee TO Tim;
  - Tim: CREATE VIEW V4 AS SELECT Emp#, Salary FROM Employee;
  - Authorizations of Tim on V4:
    - Select with Grant Option;
    - Update, Insert without Grant Option;

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.

Future Planning Question

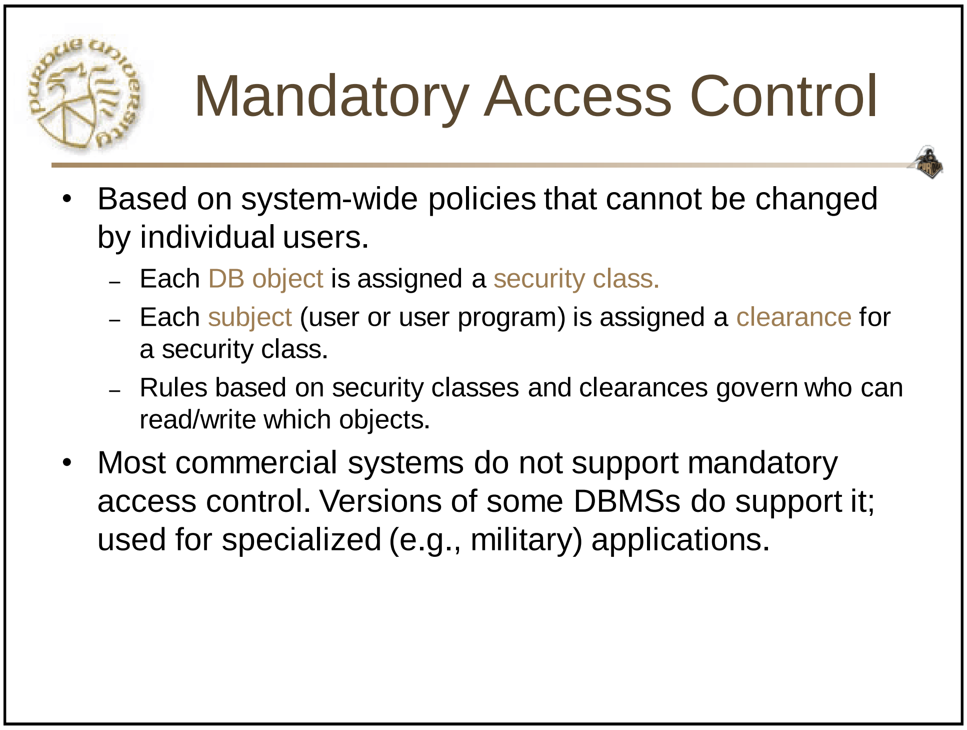
- How many of you
  - A. Have taken CS390-DM0
  - B. Are taking CS390-DM0
  - C. Plan to take CS390-DM0
  - D. Have taken or plan to take another data mining course?

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 <u>you</u> be sure it is not someone who stole his card?
  - How can <u>Sam</u> 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!)



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.