

# Computer Security

CS 426

Lecture 29

Review and DBMS Security

# Review of Web Application Security Issues

- SQL Injection
  - caused by using user inputs to form SQL queries
  - should use “prepare statement” (e.g., PreparedStatement in Java)
    - using input validation and/or quotation helps, but less bulletproof
  - for legacy code, can use, e.g., query structure checking
- Cross Site Scripting
  - caused by using user inputs to generate HTML pages
    - inputs may contain malicious scripts
  - should process user input before using it
    - use input checking helps, but less bulletproof

# Review of Web Application Security Issues

- Cross Site Request Forgery
  - caused by using cookie as the only authentication mechanism
  - use additional authentication mechanism (hidden fields, etc.)

# Final Exam Topics

- Topics in Lecture 10 or later
  - Discretionary Access Control, Mandatory Access Control, Multi-level Security, Assurance, Integrity Protection, RBAC
  - Cryptography
  - Network Security
  - Web security
- Be familiar with quizzes and homeworks

# DBMS Security Issues

- Users and authentication
- Secure communication between client and server
- Vulnerabilities of DBMS implementation
  - e.g., SQL Slammer worm
  - limit who can connect to DBMS server

# Basics

- Data are modeled as tables (relations)
- Each database may contain many tables and their definitions
- Multiple users may use the same DBMS, and have different privileges

# Access Control Mechanisms

- Key features for access controls in DBMS
  1. Privileges
  2. Views
  3. Stored Procedures
  4. Roles
  5. Row-level access control

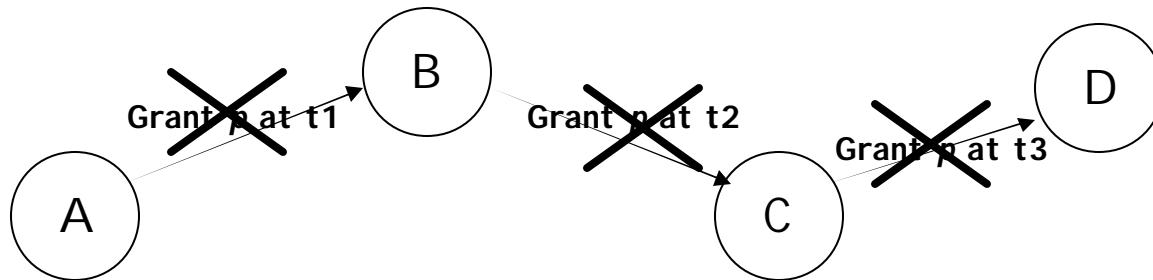
# Privileges

- System privilege
  - A right to perform a particular action or to perform an action on any schema objects of a particular type
  - E.g., ALTER DATABASE and SELECT ANY TABLE
- Object privilege
  - A right to perform a particular action on a specific schema object such as tables, views, procedures and types
  - E.g., SELECT, INSERT, UPDATE, DELETE

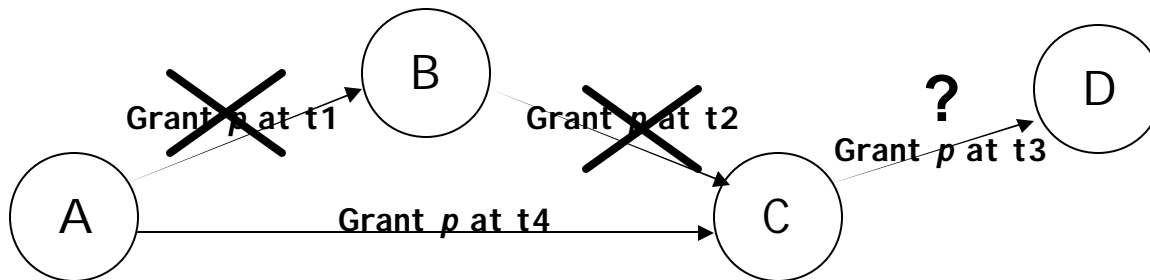
# Grant/Revoke Privileges

- System privileges
  - GRANT create table TO Bob [WITH ADMIN OPTION]
  - REVOKE create table FROM Bob
  - Users with ADMIN OPTION can not only grant the privilege to other users, but also revoke the privilege from any user.
- Object privileges
  - GRANT select ON table1 TO Bob [WITH GRANT OPTION]
  - REVOKE select ON table1 FROM Bob
  - Users who revokes a particular object privileges must be the direct grantor of the privilege.
  - There is *always* a cascading effect when an object privilege is revoked.

# Cascading Effect



- There is no timestamp for privileges.
  - Revocation (i.e., cascading effect) is coarse.



# Views

- Access control based on column and content
- Employee(Emp\_ID, name, dept\_ID, salary)
  - Want to allow employees to see only (dept\_ID, salary)
  - CREATE VIEW view\_name AS select dept\_ID, salary from Employee;
  - Grant select privileges to employees

# Views

- To create a view
  - The creator must have been explicitly (i.e., not through roles) granted one of SELECT, INSERT, UPDATE or DELETE object privileges on all base object underlying the view or corresponding system privileges.
- To grant access to the view
  - The creator must have been granted the privileges to the base tables with Grant Option.
- To access the view
  - The creator must have the proper privilege to the underlying base tables.

# Stored Procedures

- Two types of procedures in terms of access control
  - Definer's right procedures
  - Invoker's right procedures
- Definer's right procedures
  - A user of a definer's right procedure requires only the privilege to execute the procedure and no privileges on the underlying objects that the procedure access.
  - Fewer privileges have to be granted to users, resulting in tighter control of database access.
  - At runtime, the privileges of the owner are always checked.

# Definer's Right Procedure

- A user with Create Procedure privilege can effectively share any privilege he/she owns with other users without grant option.
  - Just create a definer's right procedure that uses a privilege.
  - Then grant Execute privilege to others.
  - Create Procedure privilege is very powerful.
  - When one grants Execute privilege, the system does not check if all the necessary privileges are in fact grantable.

# Invoker's Right Procedure

- Invoker's right procedures
  - A user of an invoker's right procedure needs privileges on the objects that the procedure accesses.
  - Invoker's right procedures can prevent illegal privilege sharing.
  - More like function calls in operating systems.

# Invoker's Right Procedure

- Invoker's right procedures can be embedded with Trojan Horse.
  - Users of invoker's right procedures can blindly run malicious procedures.
  - E.g.,

```
create procedure niceProcedure
  Authid Current_User As
Begin
  Do something useful;
  grant some-privileges to me;
  Do something useful;
End;
```

# Why use Roles?

- Two main purposes
  1. To manage the privileges for a user group (User roles)
    - DBA creates a role for a group of users with common privilege requirements. DBA grants all the required privileges to a role and then grants the role to appropriate users.
  2. To manage the privileges for an application (Application roles)
    - DBA creates a role (or a set of roles) for an application and grants it all necessary privileges to run the application. Then DBA grants the application role to appropriate users.

# Application Roles

- How can we secure application roles? That is, we want application roles to be used only through the associated applications.
  - Use a password for the application role and embed the password in the application. Then the role can be enabled only by the application.
  - Associate the application role with the application (i.e., a package). Then the role can be enabled only by a module in the application.

# User Assignments

- To grant a role to a user, one needs to have the “Grant Any Role” system privilege or have been granted the role with “Admin Option”.
  - GRANT ROLE clerk TO Alice;
- To revoke a role from a user, one needs to have the “Grant Any Role” system privilege or have been granted the role with “Admin Option”.
  - REVOKE ROLE clerk FROM Alice;
- Users cannot revoke a role from themselves.

# Permission Assignments

- To grant a privilege to a role, one just needs to be able to grant the privilege.
  - `GRANT insert ON table1 TO clerk;`
- To revoke a privilege from a role, one just needs to be able to revoke the privilege.
  - `REVOKE insert ON table1 FROM clerk;`
- No special admin privilege is required.
  - It can be a problem since one can make a role unusable by granting many roles to the role to exceed `MAX_ENABLED_ROLES`.
- “Grant Option” is not valid when granting an object privilege to a role.
  - To prevent the propagation of object privileges through roles.

# VPD

- How does it work?

When a user accesses a table (or view or synonym) which is protected by a VPD policy (function),

1. The Oracle server invokes the policy function.
2. The policy function returns a predicate, based on session attributes or database contents.
3. The server dynamically rewrites the submitted query by appending the returned predicate to the WHERE clause.
4. The modified SQL query is executed.

# Example

- Suppose Alice has the following table.

`Employees(e_id number(2), name varchar2(10), salary number(3));`

e_id	Name	Salary
1	Alice	80
2	Bob	60
3	Carl	99

- Users can access e\_id's and names without any restriction. But users can access only their own salary information.

# Example

## 1. Create a policy function

```
Create function sec_function(p_schema varchar2, p_obj varchar2)
Return varchar2
As
    user VARCHAR2(100);
Begin
    user := SYS_CONTEXT('userenv', 'SESSION_USER');
    return 'name = ' || user;
end if;
End;
```

# Example

## 2. Attach the policy function to Employees

```
execute dbms_ols.add_policy (object_schema => 'Alice',  
                             object_name => 'employees',  
                             policy_name => 'my_policy',  
                             function_schema => 'Alice',  
                             policy_function => 'sec_function',  
                             sec_relevant_cols=>'salary');
```

### 3. Bob accesses table Employees

select e\_id, name from Employee;

e_id	Name
1	Alice
2	Bob
3	Carl

select e\_id, name, salary from Employee;

→ select e\_id, name, salary from Employee  
where name = 'Bob';

e_id	Name	Salary
2	Bob	60

# Coming Attractions ...

- December 12:
  - Final Exam

