# Topic 14:

DBMS Security



# A Few DBMS Security Issues

# Issue #1: Availability

Two goals that often conflict:

- Making authorized access easy
- Making unauthorized access hard

Two categories of access controls:



# DAC Features of SQL (1 / 4)

Views are a very basic form of DAC:

- Gives users access to necessary information
- Completely hides origins of values

- Is a form of ‘security by obscurity’

# DAC Features of SQL (2 / 4)

Options to the CREATE USER command:

Form: `CREATE USER <username> [ <option(s)> ];`

Typical options include:



# DAC Features of SQL (3 / 4)

Providing privileges with the GRANT command:

```
Form:  GRANT <privilege>
       [ ON <object> ]
       TO <user>
       [ WITH GRANT OPTION ];
```

**Example(s):**

# DAC Features of SQL (4 / 4)

What can be GRANTed may be REVOKEd:

```
Form:  REVOKE <privilege>
       [ ON <object> ]
       FROM <user>;
```

**Example(s):**



# Mandatory Access Controls (1 / 3)

Idea: The DBMS has default security procedures that must be followed.

# Mandatory Access Controls (2 / 3)

**Example:** The Bell–LaPadula Model (1974)

Security classes are applied to two groups:



# Mandatory Access Controls (3 / 3)

Bell-Lapadula enforces two restrictions on security classes (class) assigned to a subject (S) and an object (O):

# Issue #2: Confidentiality

To help maintain confidentiality, we can require:



# A Special Case: Statistical DBMS Security

Restriction: Users may ask aggregate queries only

**Example(s):**

**Example(s):**

# Issue #3: Integrity

Idea: Be able to recover DBs after accident or disaster



# Some Standard Oracle Security Features

These are available by default in recent versions of Oracle:

- User authentication
- User privileges and roles
- Virtual Private DBs (via query modification)
- Classification of fields
- Network data encryption (via PL/SQL's DBMS_CRYPTO)
- Digital certificate authentication
- Database auditing

# A Common DBMS Attack: SQL Injection (1 / 5)

A portion of the roster of teams registered for the 2009 ACM North Central Programming Contest at Lincoln, NE:

| | | | |
|---|---|---|---|
| Kansas State University | United States | Team K-State | ACCEPTED |
| Kansas State University | United States | Wildcat hijack | ACCEPTED |
| Mount Marty College | United States | Mount Marty College Lancers | ACCEPTED |
| Nebraska Wesleyan University | United States | Epik High | ACCEPTED |
| South Dakota State University | United States | 2+2 | ACCEPTED |
| South Dakota State University | United States | Never Gonna Let You Down | ACCEPTED |
| Southwest Minnesota State University | United States | Mustang 1 | ACCEPTED |
| Southwest Minnesota State University | United States | Mustang 2 | ACCEPTED |
| University of Nebraska - Lincoln | United States | '; DROP TABLE TEAMS; | ACCEPTED |
| University of Nebraska - Lincoln | United States | Audrey II | ACCEPTED |
| University of Nebraska - Lincoln | United States | Estrogen Attack | ACCEPTED |
| University of Nebraska - Lincoln | United States | Incendiary Pigs | ACCEPTED |
| University of Nebraska - Lincoln | United States | Phelpsian Φt | ACCEPTED |
| University of Nebraska - Lincoln | United States | Smiley Faces :) | ACCEPTED |
| University of Nebraska - Lincoln | United States | ThreadDeath | ACCEPTED |
| University of Nebraska - Omaha | United States | Team Damage | ACCEPTED |



# A Common DBMS Attack: SQL Injection (2 / 5)

The attack:

A user tries to add (inject) SQL into an incomplete query,

in hopes of getting the DBMS to reveal additional information.

# A Common DBMS Attack: SQL Injection (3 / 5)

**Example(s):**

Consider this dynamically–constructed SQL query:



# A Common DBMS Attack: SQL Injection (4 / 5)

**Example(s):** (continued)

But what if the user types this input?

# A Common DBMS Attack: SQL Injection (5 / 5)

Preventing Injection Attacks: