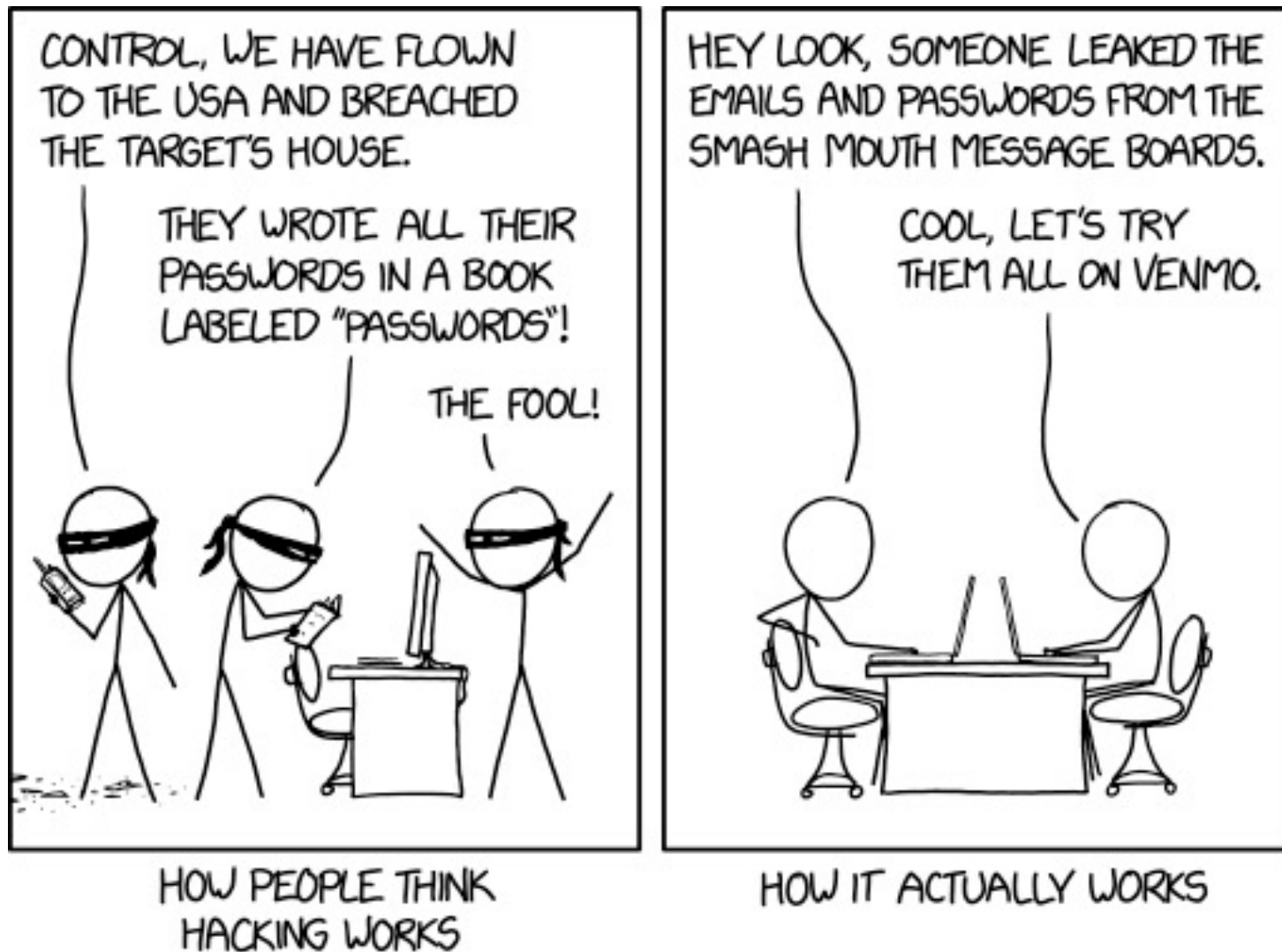


Database Security and Practical Data Management

CS 4750 Database Systems

[C.M. Ricardo, S.D. Urban, "Databases Illuminated", Ch.8]

DB Security and Privacy



[Image from <https://www.cnet.com/news/popular-web-comic-xkcd-shuts-down-forum-after-hack/>]

FERPA: Privacy and Disclosure

- FERPA (Family Educational Rights and Privacy Act)
- Information that institutions can disclose without consent
 - Name
 - Email
 - Photographs
 - Phone number
- Revealing sensitive information violates FERPA

Be careful with what information will be queried
and made available to users

[ref: <https://www.cdc.gov/phlp/publications/topic/ferpa.html>]

Sensitive Information

- Personal identifiers
 - Name
 - Student ID
 - Social security number
 - License number
- Passwords
- Context specifics
 - Grade, GPA, learning performance
 - Personal information
- Protected data (for legal and/or ethical reasons)
 - Academic records (FERPA)
 - Protected Health Information and medical records (HIPAA)
 - Customer records (GLBC)

[ref: <https://www.cdc.gov/phlp/publications/topic/ferpa.html>]

DB Security

- involves protecting the database from unauthorized access, modification, or destruction
- Must be included in the organization's overall information system security plan
- Must protect the privacy of individuals about whom data is kept
- **Privacy**: the right of individuals to have some control over information about themselves
 - Protected by laws in many countries; laws can vary significantly
 - Some laws require controls on access, disclosure, and modification of sensitive data
- Right to privacy can be protected by the DBMS

Security of Information

- **Confidentiality** – requires that only authorized users have access to information to preserve the privacy of individuals, business intellectual property, and national security effort
 - Use appropriate encryption procedures
- **Integrity** – requires that only authorized users be allowed to modify data consistency and trustworthiness
 - Incorrect data can be harmful to individual and organizations
- **Availability** – requires that information be accessible by authorized users when needed
 - Security attacks against an organization can cause business services to become unavailable, leading to violations of service level agreements that are critical to business operations

Security Threats

- Situations that could harm the system by compromising privacy or confidentiality, or by damaging the database itself
- A vulnerability is a weakness in a system (e.g., inappropriate access control or loopholes in firewall protection) that allows a threat to occur
- Security threats can occur either accidentally or deliberately

Example Accidental Security Threats

- User errors
 - User unintentionally requests object/operation for which he/she should not have authorized
- Communication system errors
 - Sending a message to the wrong user, resulting in unauthorized disclosure of database contents
 - Connecting a user to a session that belongs to another user with different access privileges
- OS or database server errors
 - Overwrite files or destroy part of database by accident
 - Fetches the wrong files or data and send them to the user
 - Fails to erase files or data that should be erased

Example Deliberate Security Threats

- Sources
 - User intentionally gains unauthorized access and/or perform unauthorized operations on the database for personal gain
 - Disgruntled employee who is familiar with the organization's computer system poses a threat to security
 - Industrial spies seeking information for competitors
- Methods
 - Wiretapping of communication lines
 - Electronic eavesdropping-picking up electronic signals
 - Reading display screens or printouts left unsupervised
 - Impersonating authorized users or users with greater access
 - Writing programs to bypass the DBMS and access the data directly
 - Writing programs to perform unauthorized operations
 - Deriving info about hidden data by clever querying (SQL injection)

Security Plan

- Define **physical security** and **information system access control** to restrict access to company resources, employee / client data
- Security plan
 - Begin with physical security measures for the building itself (e.g., require badges, barriers / locked door, sign-in)
 - Install the DBMS and configuring it securely
 - Create and secure user accounts and develop appropriate access control for users
 - Develop and enforce standards for apps that access the database
 - Encrypt sensitive data
 - Ensuring that network connections to the data are secure
 - Establish appropriate audit mechanisms for the database
 - Identify and guard against security threats, and apply security controls and security updates as needed

Information System Access Control

- **Authorization** – defines who has access to the system and the specific data, what operations they can perform on what data
- **Identification** – refers to the way in which users are identified (e.g., ID, biometrics)
- **Authentication** – verifies the identity of a user; checks against the user profile (kept secure, possibly in encrypted form)
- **Accountability** – refers to the need to capture and maintain log files that can be used for traceability when security incidents occur

6 Levels that Impact DB Security

- **Database level** – database users and authorization
- **Application level** – information management and processing
- **Operating system level** – data storage and protection
- **Network level** – data transmission
- **Physical level** – computer equipment protection
- **Human level** – social engineering protection

Security not only the database,
but the entire database application.
Breaches can happen at any of these levels.

Database Level

- By default, a DB system creator (or admin) is a **superuser**, having **global privileges on the DB system**
 - Select, insert, update, delete
 - File privileges – can import from a file
 - Create, alter, index, drop
 - View
 - Altering, creating, executing routines such as stored procedures
 - Assertion, check, trigger
 - Grant – give permission to another user; may grant your similar permissions or permissions on certain tables / columns / rows
- **General users**
 - Have no global privileges on the DB system; only have select, insert, and update privileges
 - Have **global privileges on their own DB**, including grant option

Security Mechanisms

- **View** = relation defined by a query
- Increase security – use views for access control, hide structures and data that the user should not see
- Increase logical data independence – create general presentations of data (see certain schema)
- Increase physical data independence – partitioning

```
CREATE VIEW CSMajor AS
  (SELECT sid, lname, fname, credits
   FROM student
   WHERE major = 'CS');
```

Value-dependent view

- Restrict data with specific WHERE clause used to create a view

```
CREATE VIEW MajorView AS
  (SELECT sid, lname, fname, major
   FROM student);
```

Value-independent view

- Restrict data with specific columns of the base tables

Security Mechanisms (2)

- **Access control management** – limits access per DB object (table, view, attributes), per user, per operation
- **Security log** – keeps a record of all attempted security violations
- **Audit trail** – records all access to the DB, including requestor, operation performed, workstation used, time, data items and value involved
- **Triggers** – sets up audit trail for a table, recording all changes, the time they were made, and the identity of the user who made them
- **Encryption** – should be used whenever data is communicated to other sties
 - MySQL has built-in hashing methods. However, encryption results in overhead – thus, only encrypt the passwords
 - A more efficient approach is to hash at the application level

Example: Trigger for Audit Log

```
CREATE TABLE purchase (purchase_date date NOT NULL,  
                          pid int(11) NOT NULL,  
                          custId int(11) NOT NULL,  
                          quantity int(11) DEFAULT NULL,  
                          price float DEFAULT NULL,  
                          PRIMARY KEY(purchase_date, pid, custId));
```

```
CREATE TABLE purchase_audit (log_date date NOT NULL,  
                              who_update varchar(30) NOT NULL,  
                              purchase_date date DEFAULT NULL,  
                              pid int(11) DEFAULT NULL,  
                              custid int(11) NOT NULL,  
                              old_price float NOT NULL,  
                              new_price float NOT NULL);
```

```
DELIMITER $$  
CREATE TRIGGER purchase_auditTrail  
BEFORE UPDATE ON purchase  
FOR EACH ROW  
    INSERT INTO purchase_audit  
    VALUES (CURRENT_DATE, CURRENT_USER, old.purchase_date, old.pid,  
            old.custid, old.price, new.price)  
$$  
DELIMITER ;
```


Access Control Policy

Access control – identify permissions individuals can have/do.

Block people who should not have access

Three variations of access control policy

- **Role-Based Access Control (RBAC)**
 - Group-level permission – “what can users of this role do”
 - Permissions per role; users are only granted role
- **Mandatory Access Control (MAC)**
 - Classification or privacy level
 - Permissions per classification
- **Discretionary Access Control (DAC)**
 - Personal permission – “who has access, what he/she can do”
 - Permissions per resource; change often
 - Least restrictive

Role-Based Access Control (RBAC)

- The limitations defined by job responsibilities
- Neutral access around roles; **users with the same role have the same privileges**
- Not assign permission to users directly
- Permissions per role are **normally static**
- Typically have a very few roles, **centrally administered**, and thus **easy to manage**
- Commonly used by large organizations
- **Must grant each user the correct role**

[https://en.wikipedia.org/wiki/Role-based_access_control]

Mandatory Access Control (MAC)

- Typically viewed as a **classification or privacy level**
- Most often used in military systems – emphasis on the confidentiality and classification of data, **centralized control access**
- Classify all end users and provide them with labels which permit them to gain access through security with established security guidelines
- Users **do not have the ability to override the policy** (either accidentally or intentionally) – cannot grant access to restricted table to another user
- Policy administrators implement **organization-wide security policies** – guaranteed (in principle) to be enforced for all users
- Not used much in database system nowadays

[https://en.wikipedia.org/wiki/Mandatory_access_control]

Discretionary Access Control (DAC)

- Business owner is responsible for deciding who are allowed to do what on which part of the database
- Data owner can **manage the content they own** – decide who has access, add or remove people from the list, pass the permission to other users
- Since an individual has **complete control over any objects he/she owns**, DAC is the **least restrictive** compared to the other access control policy
- Permissions given to an individual are **inherited** into other programs they use, potentially leading to malware being executed without the end user being aware of it
- Permissions per resource are **often changed**

[https://en.wikipedia.org/wiki/Discretionary_access_control]

Choosing Access Control

- If you have highly confidential or sensitive information on your business platform, use MAC or RBAC
- If you need to allow certain people to enter, DAC is simplest and most popular.
- However, if you need a lot of high security, DAC is not a good option since it is the least restrictive and privileges can inherit and transfer.

SQL Data Control Language

- Authorization sublanguage to **grant** privileges to and **revoke** privileges from users
- **Privilege** = action (such as creating, executing, reading, updating, deleting) that a user is permitted to perform on database object

```
GRANT { ALL PRIVILEGES | privilege-list }  
ON   { object-name }  
TO   { PUBLIC | user-list | role-list }  
[WITH GRANT OPTION];
```

Give
authorization

```
REVOKE { ALL PRIVILEGES | privilege-list }  
ON     object-list  
FROM   { PUBLIC | user-list | role-list }  
[CASCADE | RESTRICT];
```

Retract
authorization

Example: GRANT Statement

Grant all privileges abc1x user has to a user abc1x_a

```
GRANT ALL ON abc1x.* TO 'abc1x_a'@'%';
```

```
GRANT ALL PRIVILEGES ON abc1x.* TO 'abc1x_a'@'%';
```

Grant specific permissions to a user abc1x_a

```
GRANT SELECT, INSERT, UPDATE, DELETE ON abc1x.* TO 'abc1x_a'@'%';
```

Grant specific permissions to a user abc1x_a on a specific table

```
GRANT SELECT, INSERT, UPDATE ON abc1x.customer TO 'abc1x_a'@'%';
```

[Per the CS department policy, we do not have permission to create users or roles on CS server. The cshelpdesk created 4 subaccounts for us (in format: computingID_a, _b, _c, and _d). For example, if a computingID is abc1x, the subaccounts that come with this computingID would be abc1x_a, abc1x_b, abc1x_c, abc1x_d. The passwords for these subaccounts are the same as the main account's initial password]

* indicates all tables

'%' indicates the host (note the use of the single quote)

Example: **SHOW GRANTS** Statement

Review the privileges given to a certain user

```
SHOW GRANTS FOR 'abc1x_a'@'%' ;
```

[Per the CS department policy, we do not have permission to view other users' privileges (although we granted them the privileges)]

Review the privileges given to a certain user

```
SHOW GRANTS ;  
SHOW GRANTS FOR CURRENT_USER ;  
SHOW GRANTS FOR CURRENT_USER ( ) ;
```

* indicates all tables

'%' indicates the host (note the use of the single quote)

Example: GRANT Statement (2)

Create and use a role

```
CREATE ROLE AdvisorRole;
```

Grant privileges to the role

```
GRANT SELECT ON Student_table TO AdvisorRole;
```

Assign a role to a user

```
GRANT AdvisorRole TO 'abc1x_a'@'%';
```

```
GRANT AdvisorRole TO 'someuser'@'localhost';
```

Assign a role to another role

```
GRANT FacultyRole TO AdvisorRole;
```

Allows inheritance of role privileges

[Per the CS department policy, we do not have permission to create users or roles on CS server]

Example: **REVOKE** Statement

Revoke privileges on a certain table from a user

```
REVOKE INSERT ON Student_table FROM 'abc1x_a'@'%' ;
```

Revoke grant option without revoking the underlying privilege

```
REVOKE GRANT OPTION FOR INSERT ON Student_table FROM 'abc1x_a'@'%' ;
```

By default, if the user has passed on the privileges that are revoked, revocations cascade or trigger other revocations

If **RESTRICT** is specified, any revocation that would cascade to others will not be performed

More Example

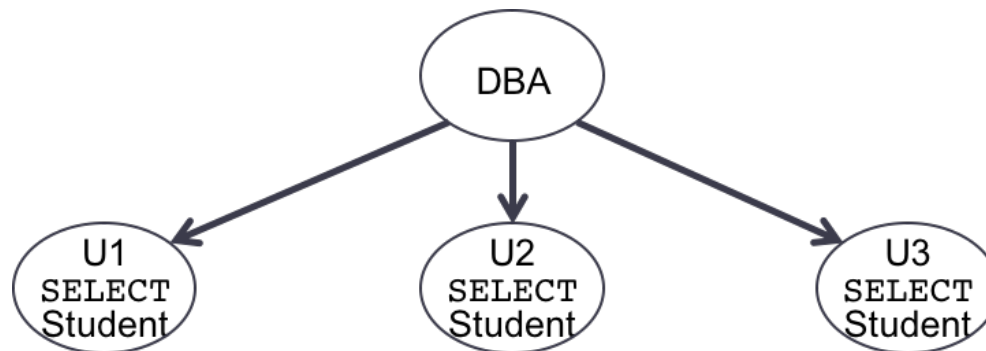
```
CREATE ROLE friendRole;  
GRANT ALL ON mytable TO friendRole;  
  
REVOKE INSERT ON mytable FROM friendRole;  
  
// assume there is a user named 'demo' who can connect to the  
database through localhost  
  
GRANT friendRole TO 'demo'@'localhost';  
  
REVOKE friendRole FROM 'demo'@'localhost';
```

Example Authorization Graph

Supposed DBA grants users U1, U2, U3 permissions on Student table

```
GRANT SELECT, INSERT, UPDATE ON Student TO U1, U2, U3 WITH GRANT OPTION;
```

Users U1, U2, and U3 would then be permitted to write SQL SELECT, INSERT, and UPDATE for the Student table, and to pass that permission on to other users



DBA = Database Admin

The figure shows only SELECT due to spaces.
[Example adapted from "Databases Illuminated", Figure 8.6]

Example Authorization Graph (2)

Supposed user U1 passes along the privilege to users U21 and U22, without the grant option.

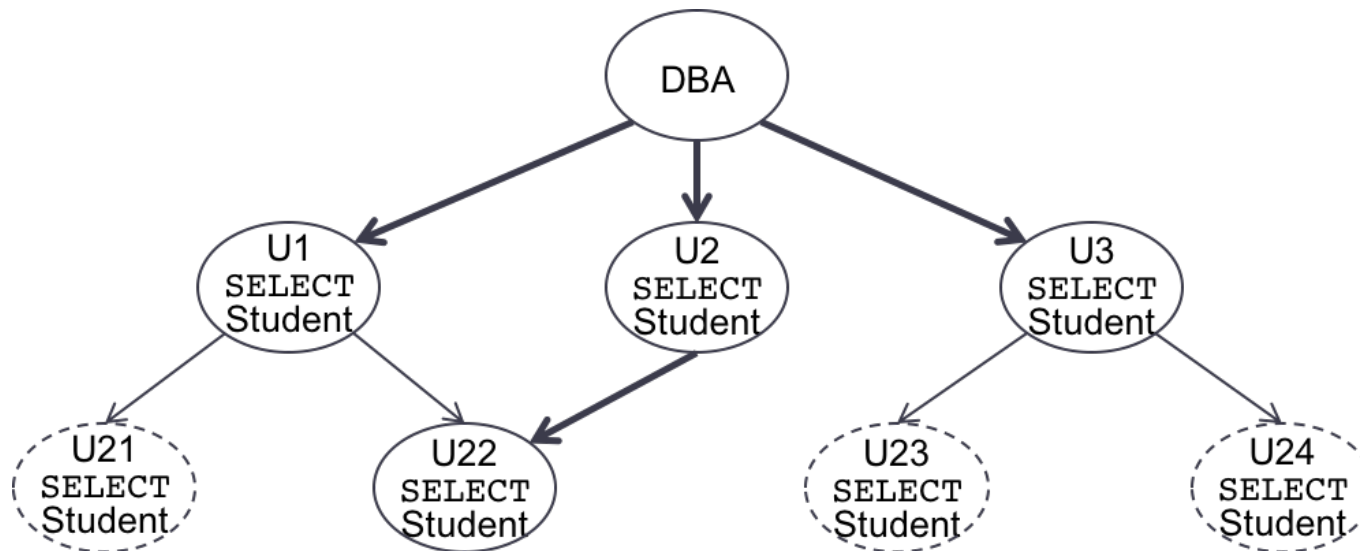
```
GRANT SELECT, INSERT, UPDATE ON Student TO U21, U22;
```

U2 also passes along the privilege to U22, with the grant option

```
GRANT SELECT, INSERT, UPDATE ON Student TO U22 WITH GRANT OPTION;
```

U3 passes along the privilege to U23 and U24, without the grant option

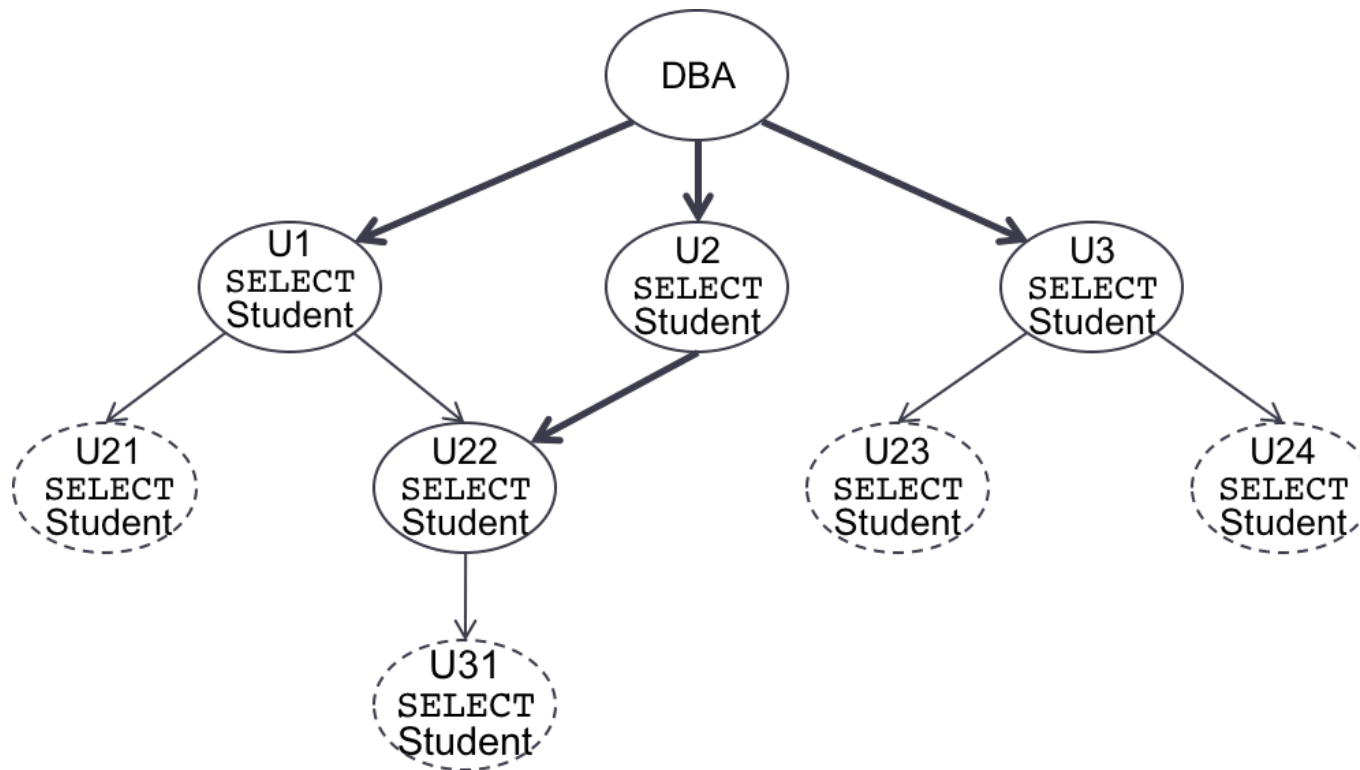
```
GRANT SELECT, INSERT, UPDATE ON Student TO U23, U24;
```



Example Authorization Graph (3)

U22 then passes the privilege to users U31, without the grant option.

```
GRANT SELECT, INSERT, UPDATE ON Student TO U31;
```



Note the privileges that are passed to other users. Improperly granting privileges may lead to security vulnerability.

6 Levels that Impact DB Security

- Database level – database users and authorization
- **Application level** – information management and processing
- **Operating system level** – data storage and protection
- **Network level** – data transmission
- **Physical level** – computer equipment protection
- **Human level** – social engineering protection

Security not only the database,
but the entire database application.
Breaches can happen at any of these levels.

Application Level

- Incorporate security aspect in the applications/programs
- Guard against SQL injection attacks – common approach is to use prepared statements
- **Prepare statements** happen in 2 phases
 - **Prepare** – sends a template to the server; the server analyzes the syntax and initialize the internal structure of the SQL statement
 - **Bind value** (if applicable) and execute – the incoming (user) inputs are treated as strings, stop them from running as scripts; replace the strings in the template; and then execute
- Implement **thorough input validations**
- **Strong typing** of applications can help prevent type errors
- Catch and **handle all errors properly**
- Use **secured channel** such as SSH or VPN

Application Level

- **Encrypt data** when possible
- Example hashing at the application level (PHP)

`htmlspecialchars(incoming_password)`

- stops script tags from being able to be executed and renders them as plaintext

`password_hash(incoming_password, algorithm_to_hash)`

- creates a password hash

`password_verify(incoming_password, existing_hashed_password)`

- returns true (1) if the *incoming_password* and the *existing_hashed_password* match; false otherwise

Operating System Level

- Set up virus protection and firewalls
- Do not install adware or spyware
- Do not use “Wizards” when installing software
 - SQLServer – clicking through the Wizards installation automatically creates a user with no password (free superuser)
- Hide stuff. Lock down your machine itself
- Run a minimal set of programs – the more program you runs, the more chance of being attacked. Disable all extra programs
- Close all ports. Lock down the interfaces

Network Level

- Hide the server. Do not make it world visible
- Separate your database server from your web server
- Limit connections to database server only from trusted sources (e.g., trusted web server) – can be done by specifying IP's or MAC's
- Only allow the world to connect to your application, which is hosted on the trusted application server (web server). Then, only the application can connect to your database. **Do not let the world (Internet) connects to your database server.**
- Do not use a default port
- Separate server for authentication
- Set up firewalls

Physical Level

- Always **lock** everything that can impact your database
- Lock the door, lock the box, lock the backups
- Implement security measures for the building itself (e.g., require badges, barriers / locked door, sign-in)
- Have a good back-up plan
 - Offsite and Online copies
 - Frequency: daily, weekly, monthly, yearly

Human Level

- Only give enough permissions to the right users
- Never give out your password
- Education and training – training about password scams, virus, how to handle your computer
- Log off your computer when not in use
- Never bring software from home to put on at work
 - Some companies / organizations have software to monitor what you put on or use at work
- Explain policies for not following rules and enforce them

Other Things to Consider

- A uniform approach to security across computer systems and databases
- Identification or authorization process that is required to initiate the creation of an account
- Who will create user accounts
- How accounts will be created
- Standard convention for users and passwords
- Password expiration
- How users will be tracked – accountability
- Levels of security breaches and penalties

Other Things to Consider (2)

- Digital signatures – use double form of public key encryption to create secure two-way communications that cannot be repudiated
- Certification authorities with SSL (Secure Sockets Layer)
 - Verisign – a method of verifying that a site is genuine
- Secure Electronic Transaction (SET) protocol – provides additional security for credit card information

Wrap-Up

- Properly handle and grant privileges can help prevent against SQL injection attacks – for example, a drop all tables command will fail
- If someone break in using a superuser, the attacker has the ability to do anything – **should not give global permissions to any users**
- For your own databases, grant permissions to subuser accounts. Set privileges so that only certain commands can be performed on certain tables

Only grant enough privileges to a user
to allow them to do their job