Database Security

- Database security - only authorized users can perform authorized activities
- Developing database security
  - Determine users’ rights and responsibilities
  - Enforce security requirements using security features from both DBMS and application programs

DBMS Security

- DBMS products provide security facilities
- They limit certain actions on certain objects to certain users or groups (also called roles)
- Almost all DBMS products use some form of user name and password security
  - Examples?

Principle of Least Privilege

- Privileges
  - “A user (or process) should have the lowest level of privilege required to perform his assigned task”
GRANT and REVOKE

- GRANT – create users / grant them privileges
- REVOKE – remove privileges

- Privileges:
  - ALL
  - SELECT
  - INSERT, DELETE, UPDATE
  - CREATE, ALTER, DROP
  - USAGE //no privileges

GRANT Syntax

GRANT privileges [columns]
ON object
TO user [IDENTIFIED BY ‘password’]
[WITH GRANT OPTION]
Example:
GRANT ALL
ON dbmusic.*
TO dbuser IDENTIFIED BY ‘userpass’

REVOKE Syntax

REVOKE priv_type
ON object
FROM user [, user]

Example:
REVOKE INSERT
ON dbmusic.*
FROM dbuser

Lab Exercise: Connect to Local DB Server

- Use WAMP on local machine
  - C:\WAMP\www – web directory
  - C:\WAMP\mysql\data – MySQL data
- Start MySQL -> MySQL Query Browser
  - Server host: localhost
  - User name: root
  - Password: [leave blank]
  - Default schema: mysql
  - OK
Lab Exercise

- Create database *midstore*; set it as default for the rest of the session (*use midstore*);
- Create tables (you can use Lab 7 MidStore SQL)
- Grant select privileges on table midstore.Product to user mxxx with password mxxx
- Logout
- Connect to MySQL on localhost as mxxx with password mxxx, default schema midstore
- SELECT * FROM Product;
- INSERT into Product(...) VALUES(...) – What happens?
- Fix the problem

Changing the Password – Option 1

- *mysql* database, *user* table, *password* column

```
UPDATE user
SET Password = PASSWORD('newpass')
WHERE User = 'dbuser';
```

Example:
While logged in as dbuser
```
SET PASSWORD = PASSWORD('it420t')
```

Changing the Password – Option 2

- SET PASSWORD
  [FOR 'username'@'host'] = PASSWORD('newpass');

Example: While logged in as dbuser
```
SET PASSWORD = PASSWORD('it420t')
```

Lab Exercise

- Login as root
- Change the password of mxxx to be midnxxx
- Logout
- Login as mxxx – use the new password
- Change your own password to midn2cxxx
DBMS Security Model With Roles

Eleanore Wu
James Johnson
Richard Ent

USER

ACCOUNTING

PERMISSION

ROLE

OBJECT

ACCOUNTING can update CUSTOMER table.

Eleanore Wu can execute MonthEnd Stored Procedure. James Johnson can alter all tables.

DBMS Security Guidelines

- Run DBMS behind a firewall, but plan as though the firewall has been breached
- Apply the latest operating system and DBMS service packs and fixes
- Use the least functionality possible
- Protect the computer that runs the DBMS

DBMS Security Guidelines

- Manage accounts and passwords
  - Use a low privilege user account for the DBMS service
  - Protect database accounts with strong passwords
  - Monitor failed login attempts
  - Frequently check group and role memberships
  - Audit accounts with null passwords
  - Assign accounts the lowest privileges possible
  - Limit DBA account privileges

- Planning
  - Develop a security plan for preventing and detecting security problems
  - Create procedures for security emergencies and practice them

Application Security

- If DBMS security features are inadequate, additional security code could be written in application program
  - Example?
- Use the DBMS security features first
SQL Injection Attacks!

- **SQL injection attack** occurs when data from the user is used to modify a SQL statement.
- Example: users are asked to enter their alpha into a Web form textbox.
  - User input: 081234 OR TRUE
  ```sql
  SELECT * FROM STUDENT_GRADES
  WHERE Alpha = 081234 OR TRUE;
  ```
  - Result?

Making your MySQL Database Secure - Server

- Do not run MySQL (mysqld) as root!
- Set up a user just for running the server
- Make directories accessible just to this user
- Run MySQL server behind a firewall

Making your MySQL Database Secure - Passwords

- Make sure all users have strong passwords.
- Connecting from PHP:
  - Have the user an password stored in a file `my_db_connect.inc.php` and include this file when required.
  - Store `my_db_connect.inc.php` outside web tree (`$_SERVER['DOCUMENT_ROOT']`).
  - Store passwords only in `.php` files (not `.inc`, `.txt`, etc.).
  - Do not store application passwords in plain text. Use `sha1()`.

Making your MySQL Database Secure – User Privileges

- Use principle of least privilege:
  - Grant only the privileges actually needed to each user.
  - Grant access only from the host(s) that they will be connecting from.
Making your MySQL Database Secure – Web Issues

- Set up a special user just for web connections, **with minimum required privileges**
- **Check all data coming from user** (SQL Injection Attacks!!)
  - addslashes() / stripslashes()
  - doubleval()

Lab Exercise – Run Mids Store on Local Machine

- Copy your code from Lab 7 or Lab 8 into C:\WAMP\www
- Modify to connect to
  - Host: localhost
  - User: mxxxx
  - Password: midn2cxxxx
  - Database: midstore
- Open browser. Test your programs.

Lab Exercise – Secure the Password File

- Create a folder “Secret” in C:\WAMP
- Move the my_connect_db.inc.php file (or other file that you "require") in C:\WAMP:\Secret
- Include the file from new location instead of
  require("my_connect_db.inc.php");
  Have
  require($_SERVER['DOCUMENT_ROOT'] . '/../Secret/my_connect_db.inc.php");

Lab Exercise – SQL Queries for Mids Store Database

- Find barcode, product name and price for products bought by at least two customers
- List category name and the average price of products in that category, for each category
- List product name, price, and category for products with price higher than most expensive product in ‘food’ category