Database Security

Anne Denton

Department of Computer Science
North Dakota State University

Outline

Table of Contents

Concerns

- Loss of integrity
  - Corruption of data
  - Could be through intentional fraudulent changes
  - Could be accidental
- Loss of availability
  - When a user has a right to access data but cannot
- Loss of confidentiality
  - Could violate privacy rights
  - Could disclose explicitly confidential data

Types of control

- Access control
  - Access to databases is granted to users based on database operations
  - Access can be specified for schemas, tables, and views
- Inference control
  - Allowing access to statistical information without disclosing personal data
  - Relevant especially in statistical databases
  - Research area of privacy preserving machine learning
- Encryption
  - Back-end storage can be vulnerable
  - Data transmission across networks is a particular concern

Practical Perspective

- Application security
  - SQL Injection allows code insertion through faulty web applications
  - Discussed in the Applications section
- Access control
  - Discretionary access control through privilege granting standard in all DBMSs
  - Mandatory access control available in some DBMSs enforce multiple security levels
- Systems side
  - Encryption protects against system- and network-level attacks
  - System configuration files provide additional fire-wall-like protections
  - DBMSs are typically deployed on dedicated servers with highly restrictive system-level firewalls

Table of Contents

Account creation

- Privileges are granted in two steps
  - First a user account is created with a password that allows authentication
  - Second, the account is granted privileges (authorization) e.g. `GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA schema_a TO user_a`
    https://www.postgresql.org/docs/13/sql-grant.html
  - (Note that you have entire databases to yourself)
- DBMSs typically have a special account that has all administrative rights ("`postgres`" user for PostgreSQL)
- Privileges are granted depending on types of SQL commands
  - "`SELECT`" for querying
  - "`INSERT`", "`UPDATE`", and "`DELETE`" for modifications
  - "`ALL PRIVILEGES`" if access is to be unrestricted
- Privileges can be revoked similarly

Question 1 (Multiple answers may be correct)

Creating a database user/role has two steps (beyond creating the database and possibly schema) that are part of discretionary access control. Among them are

1. Creating a user name and password combination that are used for authenticating the user
2. Specifying the security class of the user such as “Top Secret," “Secret," “Confidential," and “Unclassified"
3. Granting the user/role privileges that authorize reading and/or writing certain tables
4. Giving a user separate roles at lower security levels than the maximum because it is not possible to write from a high security level to a lower security level

Specific access control through views

- Privileges can be granted to views much like tables
- Allows specifying specific attributes or rows in a table
  - Create a view of the information that is to be shared
  - Only grant privileges to view
  - Updates to tables can happen unchanged
- By default this can only be used for "SELECT" and not for modifications
- Updatable views exists
  - Cannot contain aggregate functions, set-theoretic operations, "DISTINCT" and some other clauses for which the reverse is ambiguous

Question 2 (Multiple answers may be correct)

Views may help with adequate security for tables, because

1. A view may be created to only contain a subset of rows and/or columns that content non-confidential information
2. A user may be given read privileges to a view, even if they don't have such privileges for the underlying table
3. A user may be given write privileges to a view, even for values that are the result of aggregate functions
4. A user may be given write privileges to a view, but only to those values that are the result of set-theoretic operations

Privilege propagation

- Any privilege granting can be given in a such way that the user can grant the same privilege to others using `WITH GRANT OPTION`, e.g.

  `GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA schema_a TO user_a` **`WITH GRANT OPTION`**
- Problematic from a security perspective, since the DBA may then not know who has access to what
- It can then also be problematic to revoke privileges
  - If a person receives privileges from multiple sides they would have to be revoked from all sides
- Some DBAs rather choose to grant all privileges themselves
- Some SQL extensions have been developed to limit propagation, but are not standard
- Alternatives are mandatory and role-based access control

Question 3 (Multiple answers may be correct)

With SQL, a database user A can grant privileges to a database user B

1. If user B will need the access to do database administrator types of work
2. Any time A has the privileges themselves
3. If A was granted the privileges “WITH GRANT OPTION"
4. If A has has access to the administrator account (“postgres" in PostgreSQL)

Table of Contents

Mandatory access control

- Developed for military and government applications
- Users cannot override the policy
- Organization-wide
- Often used in addition to discretionary access control
- Originally tied to multi-level security with security classes
  - Top secret (TS)
  - Secret (S)
  - Confidential (C)
  - Unclassified (U)

Principles of MAC

- Mandatory access control applied to
  - Subjects, that have a clearance
  - Objects, that are classified
- Bell-LaPadula model
  - Simple security property says that a subject cannot read at a higher classification level (no "read up")
  - Security property says that a subject cannot write to a lower classification level (no "write down"), i.e. have to log in with lower classification to communicate with that classification level

Label Security

- More basic versions of mandatory access control use label-based security
- Based on a label security policy that is defined by an administrator
- Security labels for objects
  - Can be applied to any object, such as a schema, table, column, aggregate, or domain
- Row-level access control
  - Requires an extra label column

Example systems

- Oracle Label Security
  - Built on Virtual Private Database (VPD) technology
  - Query evaluation considers discretionary access control first and label-based security second
- PostgreSQL offers basic elements of label-based security
  - Object-level labels
    `https://www.postgresql.org/docs/13/sql-security-label.html`
  - Row-level security
    `https://www.postgresql.org/docs/13/ddl-rowsecurity.html`
- Specialized DBMSs such as Rubix include more advanced features
  `http://www.rubix.com/`

Role-based access control

- Role-based access control generalizes security privileges of groups of users
  - Structured around roles of users within organizations
- Includes functionality of both discretionary and mandatory access control
  - Granting and revoking of privileges resembles discretionary access control
  - Mandatory access control policy can also be specified in terms of roles

Table of Contents

SQL Injection

- Discussed in the application section of this course
- Often considered under web application security
  - Depends on web framework
  - Once injected code reaches database, little can be done to prevent damage
- Risks associated with SQL injection
  - Manipulation of existing SQL statements, e.g., expanding the set of records that are returned
  - Injecting additional SQL statements, while bypassing authentication
  - Function call injection may call privileged database functions or even system-level functions
  - Database fingerprinting, i.e. extracting information about the database backend

Table of Contents

Protecting database systems through firewalls

- Firewalls, as such, are not specific to databases
- However, system protections are typically used to give databases special protection
  - Databases are often placed on dedicated systems
  - System that is distinct from web server
  - Allows to only open ports that are needed for specific database interactions
  - Protects database backend storage
  - Network traffic has to be protected separately

Firewalls

- At the level of one system, a firewall is a set of rules
  - Which incoming and outgoing traffic is allowed
  - What protocols may be used for that traffic
  - Which ports may be used for that traffic
  - On Linux systems ufw (Uncomplicated FireWall) provides a very basic interface for changing rules
- Example ports and protocols
  - Port 22 and protocol TCP/IP used for login via ssh
  - Port 5432 and protocol TCP used for PostgreSQL
- Typically no other incoming traffic allowed on a database backend server
- Port numbers may be changed for extra security
- Limiting outgoing traffic can help avoid propagating damage

Question 4 (Multiple answers may be correct)

Firewalls for the system on which the database is hosted typically helps prevent

1. Cases of SQL injection
2. Attempts of accessing the database backend storage
3. Problems due to excessive propagation of privileges when using “WITH GRANT OPTION" while granting access
4. Someone breaking into an administrator account, such as “`postgres` in a PostgreSQL database

Database configuration files

- Databases also allow system level configuration of access
- Check detailed DBMS-specific information, e.g.
  `https://www.postgresql.org/docs/13/auth-pg-hba-conf.html`
- Allows for example host-based access control
  - Which client machines can access a database
  - How the users on those machines must authenticate themselves

Encryption

Encryption is important for two distinct purposes

- Protection of unauthorized access
    - Database content and/or traffic to and from the database is encrypted such that it cannot be read
- Establishing the identity of a person and/or service
    - Digital signatures identify an entity
    - Digital certificates tie the digital signature to a certificate owner

Asymmetric encryption or public key encryption

- Relies on public and private keys
- Among the first and most commonly used schemes is RSA encryption
  - Named after inventors Rivest, Shamir, and Adleman
- For protection against unauthorized access across a network
  - Data are encrypted with the public key of the recipient
  - Decrypted with the private key
- To provide a digital signature
  - A timestamp or other message-dependent piece of information is encrypted using the private key of the sender
  - Authenticity tested by decrypting with their public key

Example uses of encryption in databases

- PostgreSQL is set up to use encryption of
  - Password storage
  - Specific columns
  - Data partitions
  - Passwords transfer across a network
  - Data transfer across a network
  - SSL Host Authentication using SSL keys or certificates
  - Client-side encryption

`https://www.postgresql.org/docs/13/encryption-options.html`

Question 5 (Multiple answers may be correct)

Encryption is typically useful towards protecting against someone

1. Being able to read database files after gaining access to the system that hosts the database
2. Listing the contents of a table after gaining access to a database via SQL injection
3. Intercepting and reading passwords to and from a web application with database backend
4. Inappropriately granting others access to tables they shouldn't have access to