SQL Authorization and Database Security

FCDB 10.1

Dr. Chris Mayfield

Department of Computer Science
James Madison University

Apr 09, 2012
Database security 101

- Access control, users/groups
- Views (for limiting access)
- Encryption (e.g., passwords)
- Denial of service attacks
- Fault tolerance (hot standby)
- Privacy of user’s information
- Audit trail (using triggers?)
Privileges

POSIX file system:
- {User, Group, Other} may {4=Read, 2=Write, 1=Execute}
- Example: `chmod 755 myfile.txt`

SQL database:
- SELECT, INSERT, UPDATE, DELETE
- TRUNCATE, REFERENCES, TRIGGER
- CREATE, CONNECT, TEMPORARY
- EXECUTE, USAGE, ALL PRIVILEGES

http://www.postgresql.org/docs/9.1/static/sql-grant.html
Granting privileges

GRANT <privilege list> ON <database element> TO <user list>

GRANT SELECT, INSERT ON Studio TO kirk, picard WITH GRANT OPTION;

-- PostgreSQL syntax is slightly different from the book
GRANT SELECT (title), UPDATE (title) ON Movies TO sisko;

-- PUBLIC means any user
GRANT INSERT ON films TO PUBLIC;

http://www.postgresql.org/docs/9.1/static/ddl-priv.html
Grant diagrams

Directed graph:
- Nodes = user and privilege
  - ** = owner of element
  - * = with grant option
- Edges = who granted privilege

Fundamental rule:
- User C has privilege Q as long as
  - path from X** to CQ, CQ*, or CQ**
  - and P is a superprivilege of Q
- Remember that P could be Q, and X could be C
Example grant diagram

- A owns the object for which \( P \) is a privilege
  - User A: `GRANT P TO B WITH GRANT OPTION;`
  - User B: `GRANT P TO C WITH GRANT OPTION;`
  - User A: `GRANT P TO C;`
Example revoke cascade

User A: \text{REVOKE} \ P \ \text{FROM} \ B \ \text{CASCADE};

- Both $B$ and $C$ lose $P^*$
- However, $C$ still has $P$
Revoking privileges

REVOKE <privilege list> ON <database element> FROM <user list>
[ CASCADE | RESTRICT ]

Note: RESTRICT by default

▶ Cannot revoke if has any dependent privileges

REVOKE SELECT, INSERT ON Studio
FROM picard CASCADE;

-- PostgreSQL has additional options
REVOKE ALL PRIVILEGES ON Studio FROM picard;

See practice problems on page 436
Creating initial privileges

How I created your databases:

```sql
CREATE DATABASE mayfiecs OWNER = mayfiecs;
REVOKE ALL ON DATABASE mayfiecs FROM public;
```

And made “postgres” DB read-only:

```sql
REVOKE CREATE ON DATABASE postgres FROM public;
REVOKE TEMP ON DATABASE postgres FROM public;
REVOKE CREATE ON SCHEMA public FROM public;
```
Privilege-checking process

1. Is the user the owner?
2. Is the object public?
3. Does the user have access?

Normal user:

```sql
CREATE ROLE mayfiecs LOGIN NOSUPERUSER INHERIT NOCREATEDB NOCREATEROLE NOREPLICATION;
-- Each user has a set of authorization IDs
GRANT students TO mayfiecs;
```

Group role:

```sql
CREATE ROLE students NOSUPERUSER INHERIT NOCREATEDB NOCREATEROLE NOREPLICATION;
```

Super user:

```sql
CREATE ROLE postgres LOGIN SUPERUSER INHERIT CREATEDB CREATEROLE REPLICATION;
```
SQL Injection

Why is this still a problem?
Examples of SQL injection

In Java:

```java
String sql = "SELECT * FROM users"
    + " WHERE name = '<'" + userName + "'";
```

Hello, my name is: ‘ OR ’1’=’1

```sql
SELECT * FROM users
WHERE name = '' OR '1'='1';
```

Better yet, don’t check my password: ‘ OR 1=1 --

```sql
SELECT * FROM users
WHERE name = '' OR 1=1 -- ';
```

Or try to get someone fired: ‘; DROP TABLE users; --

```sql
SELECT * FROM users
WHERE name = ''; DROP TABLE users; --';
```
SQL injection methods

- Adding or modifying data
  - Denial of service
  - Privilege escalation

- Bypassing authentication
  - Evading detection
  - Executing remote commands

- Extracting data
  - Identifying injectable parameters
  - Inferring sensitive information

http://www.unixwiz.net/techtips/sql-injection.html