Exercise Session 5

8 December

1 Authorization, SQL Injection, Transactions (3 parts, 12p)

Consider an existing database with the following database definition in a PostgreSQL DBMS:

```
CREATE TABLE Users (
    id INTEGER PRIMARY KEY,
    name TEXT,
    password TEXT
);

CREATE TABLE UserStatus (
    id INTEGER PRIMARY KEY REFERENCES Users,
    loggedin BOOLEAN NOT NULL
);

CREATE TABLE Logbook (
    id INTEGER REFERENCES Users,
    timestamp INTEGER,
    name TEXT,
    PRIMARY KEY (id, timestamp)
);
```

6a. A database user "Alice" is granted the following permissions:

```
GRANT SELECT(id, name, password) ON Users TO Alice;
GRANT SELECT(id, loggedin) ON UserStatus TO Alice;
GRANT INSERT(id, loggedin) ON UserStatus TO Alice;
GRANT SELECT(id, timestamp, name) ON LogBook TO Alice;
```

Alice now executes the following SQL statement:

```
INSERT INTO LogBook
    SELECT u.id, 201706071400, u.name
        FROM (UserStatus us JOIN Users u ON us.id = u.id)
        WHERE us.loggedin = True;
```

We want Alice to only have exactly the privileges that are necessary to complete this SQL statement. Does Alice have the correct privileges? What minimal set of permissions should she be granted instead, if not the same as listed above? (4p)

6b. Users of a web application are allowed to query this database for a certain user id. This functionality is implemented in JDBC using the following code fragment:

```
...
String query = "SELECT * FROM UserStatus WHERE id = '" + userinput + "'";
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery(query);
...
```

The `userinput` variable is controlled directly by an attacker. What can an attacker input into `userinput` so that the SQL query returns all data in `UserStatus`? What is this specific security problem called? How should the above code be corrected to prevent this problem? (4p)

6c. The following transaction calculates the total number of entries in UserStatus as the sum of the number of logged-in and not logged-in users.

```
BEGIN TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT
   (SELECT COUNT(*) FROM UserStatus WHERE loggedin = True)
   +
   (SELECT COUNT(*) FROM UserStatus WHERE loggedin = False);
COMMIT;
```

The used transaction isolation level is not sufficient to ensure an accurate count of entries in UserStatus. Why not? Give all isolation levels that are sufficient so that the query works as expected. (4p)