P1L7 Database Security

Importance of Database Security
Why is securing data so important?
-Databases store massive amounts of sensitive data
-Data has structure that influences how it is accessed.
-Accessed via queries or programs written in languages like SQL
-Transactional nature of queries (updates or reads) - it is done completely or not done at all.
-Derived data or database views.

Who are the biggest threat to databases? Insiders and unauthorized users.
Databases are attractive to hackers because:
-They store information that is easily monetized.
-The store information about a lot of users.
-Queries languages used to access data can be abused to gain unauthorized access.

Relational Databases (RDBS)
-relational model based database systems are widely used in real-world environment.
-a relational database consists of relations or tables
-a table is defined by a schema and consists of tuples
-Tuples store attribute values as defined by schema
-Keys used to access data in tuples

Primary key → a unique key for each tuple of the table
Foreign Key → a primary key from another table

Operations on relations:
- Create a table, select, insert, update, join, and delete
- Example: SELECT * FROM EMPLOYEE WHERE DID = ‘15’

Queries written in a query language use basic operations to access data in a database.
Two tuples in a relation can not have the same primary key value.

We can use a database view to enhance data security because it can exclude sensitive attributes that should not be accessible to certain users.

Database Access Control

Two commands: GRANT and REVOKE

Grant or Revoke either privileges to a table or a role (user).
If you don't specify a table … the command refers to all tables.
Identified by password is optional (this is only on the Grant command).

Privileges can be for operations: SELECT, INSERT, UPDATE, or DELETE

Example: Alice has SELECT access to a table and she can propagate this access to BOB when ..Alice was granted this access with the GRANT option.

Cascading Authorizations Quiz
Cascading authorizations occur when an access is propagated multiple times and possibly by several users. Assume that Alice grants access to Bob who grants it further to Charlie. When Alice revokes access to Bob, Charlie's access is also revoked.

MAC or DAC Quiz
In MAC database access control can be managed centrally by a few privileged users.

Attacks on Databases: SQL Injections
SQL Injections → Malicious SQL commands are sent to a database.
When successful: these attacks can impact confidentiality (extraction of data) and integrity of data (corruption of data).

In a web application environment, typically a script takes user input and builds an SQL query.

Web application vulnerability can be used to craft an SQL injection.

Example: return information about items shipped to a certain city, specified user in a web application that uses forms.
Script code:

```
var Shipcity;
Shipcity = Request.form("Shipcity")
var sql = "select * from OrdersTable where Shipcity = "" + shipcity + "".
```

- User enters REDMOND
- Script generates SELECT * FROM OrdersTable where Shipcity = 'Redmond'

What if the user enters …. REDMOND; DROP table OrdersTable; ?

In this case, SELECT * FROM OrdersTable WHERE Shipcity = ‘Redmond’; DROP OrdersTable is generated. The malicious user is able to inject code that will delete the table.

SQL Injection Defenses

- Input Checking -- follow the golden rule, all input is evil.
- See OWASP top 10 proactive controls at the link in the instructor notes.

Top 10 Defenses

1. Parameterize Queries
2. Encode data
3. Validate all inputs
4. Implement appropriate access controls
5. Establish identity and authentication controls
6. Protect Data and Privacy
7. Implement logging, error handling, and intrusion detection
8. Leverage security features of frameworks and security libraries
9. Include security-specific requirements
10. Design and architect security libraries

OWASP Proactive Controls

SQL Login Quiz

A web application script uses the following code to generate a query:
Query = “SELECT accounts FROM users WHERE login = ‘ “ + login + “ ‘ AND pass = ‘ “ + password + “ ‘ AND pin = “ + pin;

This query will be executed when user login, password, and pin are correctly provided.

Inference Attacks on Databases

Certain aggregate/statistical queries can be allowed by all users.
This attack works by…

1. making a query that the user is authorized to make
2. the using the data to make an inference about the results returned.

For example:
A database contains studentid, student_standing(junior or senior), exam 1_score, exam2_score, final_grade.

A malicious user can ask for the average score on an exam. Any student should be able to do this. The malicious user asks, for example, for the average score on an exam by juniors in the course. If there is only one junior in the course, this student’s score will be returned.

Inference attacks sometimes require additional outside information. So maybe the attacker might know when the target takes the exam late. So the attacker runs the average score before and after the late exam is taken, and the attacker can infer the student's score.

Defenses Against Inference Attacks

These kinds of defenses are hard to implement.

- Do not allow aggregate query results when the set of tuples selected is either too small or too large. An extreme case would be where there is one tuple returned.
- De-identification: Transform data by removing identifying information. For example drop the student id from the the return.
- Anonymization: replace exact values with a more general value.

Regardless of the defense, it must be done with care.

SQL Inference Attack Quiz #2

Assume the data in the database is de-identified by removing student id and other information, such as names. Furthermore, the field that has the state of the student is generalized by replacing it with the US region. The generalization ensures that there are at least two students from each region, *but inference attacks are still possible.*