

Data Security and Privacy

Chapter 6:

Database Security

Lesson Introduction

- Understand the importance of securing data stored in databases
 - Learn how the structured nature of data in databases impacts security mechanisms
 - Understand attacks and defenses that specifically target databases
 - Database Encryption
-

Importance of Database Security

Why securing data stored in databases so important and different?

- 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 (Structured Query Language)**
- Transactional nature of queries (updates or reads)
- Derived data or database views

Database Threats Quiz

Choose the best answer.

Oracle, a major database vendor, sponsored a database security study which identified key security threats. In your view, which of the following is the biggest threat...

☐

External hackers

☐

Insiders and unauthorized users

Database Hacking Quiz

Mark all applicable answers.

Databases are attractive targets for hackers because...

☐

They store information such as SS#, DOB etc. that can be easily monetized

☐

They store information about lots of users

☐

Queries languages used to access data can be abused to gain unauthorized access.

Relational Database Systems (RDBS)



- Relational model based database systems are **widely used in real-world environments**
- 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**

Relational Database Systems (RDBS)

Employee Table

Ename	Did	Salarycode	Eid	Ephone
Robin	15	23	2345	6127092485
Neil	13	12	5088	6127092246
Jasmine	4	26	7712	6127099348
Cody	15	22	9664	6127093148
Holly	8	23	3054	6127092729
Robin	8	24	2976	6127091945
Smith	9	21	4490	6127099380


Foreign key


Primary key

Relational Database Systems (RDBS)

Operations on relations:

- Create, select, insert, update, join and delete

- Example:

```
SELECT *  
FROM EMPLOYEE  
WHERE DID = '15'
```

- It returns tuples for Robin and Cody

Queries written in a query language (e.g., SQL) use such basic operations to access data in a database as needed.



Key Value Quiz

Choose the best answer:

Two tuples (rows) in a relation can have the same primary key value.

☐

Yes

☐

No



Database Views Quiz

Choose the best answer.

We can use a database view to enhance data security because...

☐

It can exclude sensitive attributes that should not be accessible to certain users

☐

A view can only be accessed by a single user

Database Access Control

Two commands: GRANT and REVOKE

```
GRANT                {privileges | role}
[ON                  table]
TO                   {user | role | public}
[IDENTIFIED BY password]
[WITH                GRANT OPTION]
```

Example: GRANT SELECT ON ANY TABLE TO Alice

Database Access Control

Privileges can be for operations such as **SELECT, INSERT, UPDATE OR DELETE.**

REVOKE	{privileges role}
[ON	table]
FROM	{ user role PUBLIC}

Example: REVOKE SELECT ON ANY TABLE FROM Alice

Database Access Control Quiz

Choose the best answer.

Alice has SELECT access to a table and she can propagate this access to Bob when...

☐

Alice was granted this access with GRANT option

☐

She can always propagate an access she has

Cascading Authorizations Quiz

Choose the best answer.

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, should Charlie's access be also revoked?

☐

Yes

☐

No

DAC or MAC Quiz

Choose the best answer.

Database access control can be managed centrally by a few privileged users. This is an example of...

☐

DAC

☐

MAC

Attacks on Databases: SQL Injections



- Malicious SQL commands are sent to a database
- Can impact **both confidentiality (extraction of data) and integrity (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**

Attacks on Databases: SQL Injections

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 + "'";
```

SQL Injection Example

- User enters REDMOND,
- Script generates `SELECT * FROM OrdersTable Where Shipcity = 'Redmond'`.
- What if user enters Redmond' ; `DROP table OrdersTable;` ?
- In this case, `SELECT * FROM OrdersTable WHERE Shipcity = 'Redmond'` ; `DROP OrdersTable` is generated
 - Malicious user is able to inject code to delete the table
- Many other code injection examples exist

SQL Injection Defenses



- Input checking
(golden rule – all input is evil)
- See **OWASP top 10 proactive controls** at https://www.owasp.org/index.php/OWASP_Proactive_Controls



SQL Login Quiz

Mark all applicable answers.

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; The various arguments are read from a form to generate Query.

This query is executed to get a user's account information when the following is provided correctly...

☐

Login name

☐

Password

☐

PIN



SQL Login Quiz #2

Choose the best answer.

Query = "SELECT accounts FROM users WHERE login = ' " + login + " ' AND pass = ' " + password + " ' AND pin = " + pin; The various arguments are read from a form to generate Query.

If a user types "or 1 = 1 --" for login in the above query...

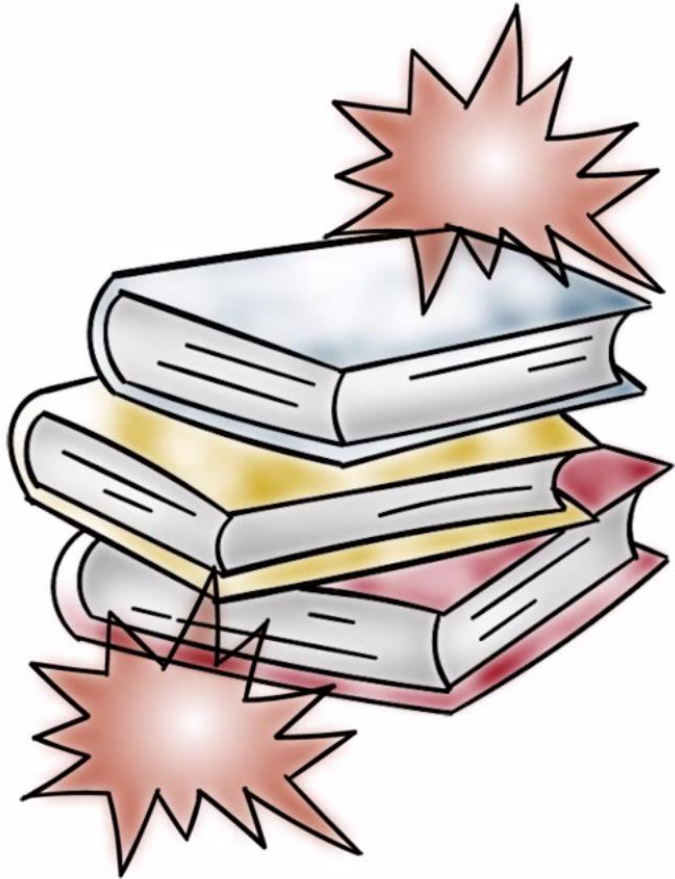
☐

Query will fail because the provided login is not a correct user

☐

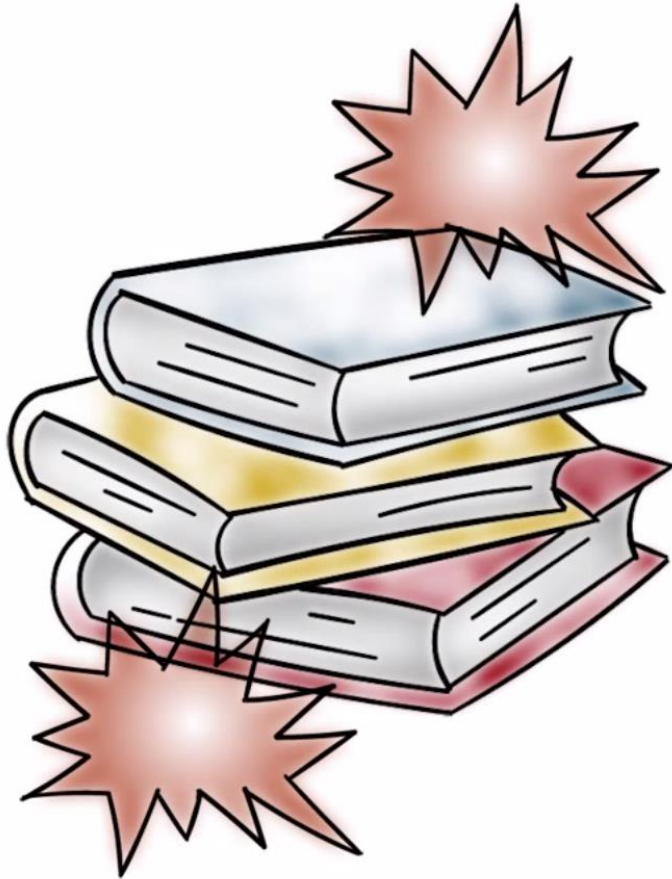
An injection attack will result in all users' account data being returned

Inference Attacks on Databases



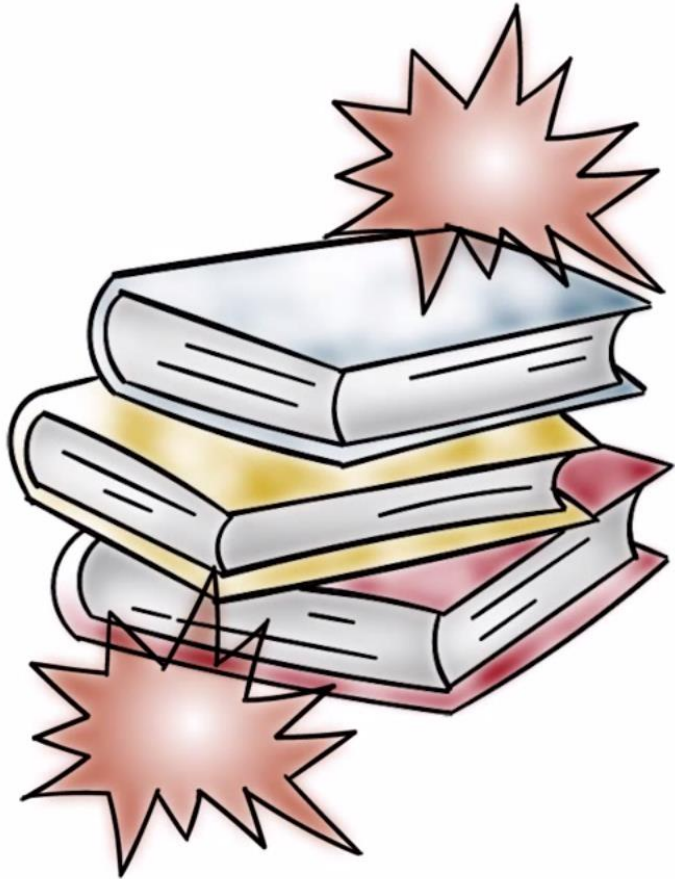
- **Certain aggregate/statistical queries can be allowed by all users.**
- Consider a student grade database with schema `studentid`, `student_standing` (junior or senior), `exam1_score`, `exam2_score`, `final_grade`.

Inference Attacks on Databases



- Average score on an exam is a query that any student should be able to run.
- **Attacker wants to find exact score of some student.**
- **Inference attack when target takes the exam late**
 - Average score before target takes the exam
 - Average score after target takes the exam
 - **Target score can be easily found**

Inference Attacks on Databases



Another example: only one student has junior standing in a senior class

- Get average score of students who have junior standing
- **This query discloses score of a single student**

Defenses Against Inference Attacks

- Do not allow aggregate query results when the set of tuples selected is either too small or too large
- Transform data by removing identifying information
 - Deidentification
 - Anonymization
 - This has to be done with care

SQL Inference Attack Quiz

Choose the best answer.

The database that stores student exam scores allows queries that return average score for students coming from various states. Can this lead to an inference attack in this system?

☐

Yes, depending on how many students come from each state

☐

No, it is not possible

SQL Inference Attack Quiz #2

Choose the best answer.

Assume in (1), 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 (e.g., Midwest). The generalization ensures that there are at least two students from each region. Are inference attacks still possible?

☐

Yes

☐

No

Database Encryption

- The database is typically the most valuable information resource for any organization
 - Protected by multiple layers of security
 - Firewalls, authentication, general access control systems, DB access control systems, database encryption
 - Encryption becomes the last line of defense in database security
 - Can be applied to the entire database, at the record level, the attribute level, or level of the individual field
- Disadvantages to encryption:
 - Key management
 - Authorized users must have access to the decryption key for the data for which they have access
 - Inflexibility
 - When part or all of the database is encrypted it becomes more difficult to perform record searching

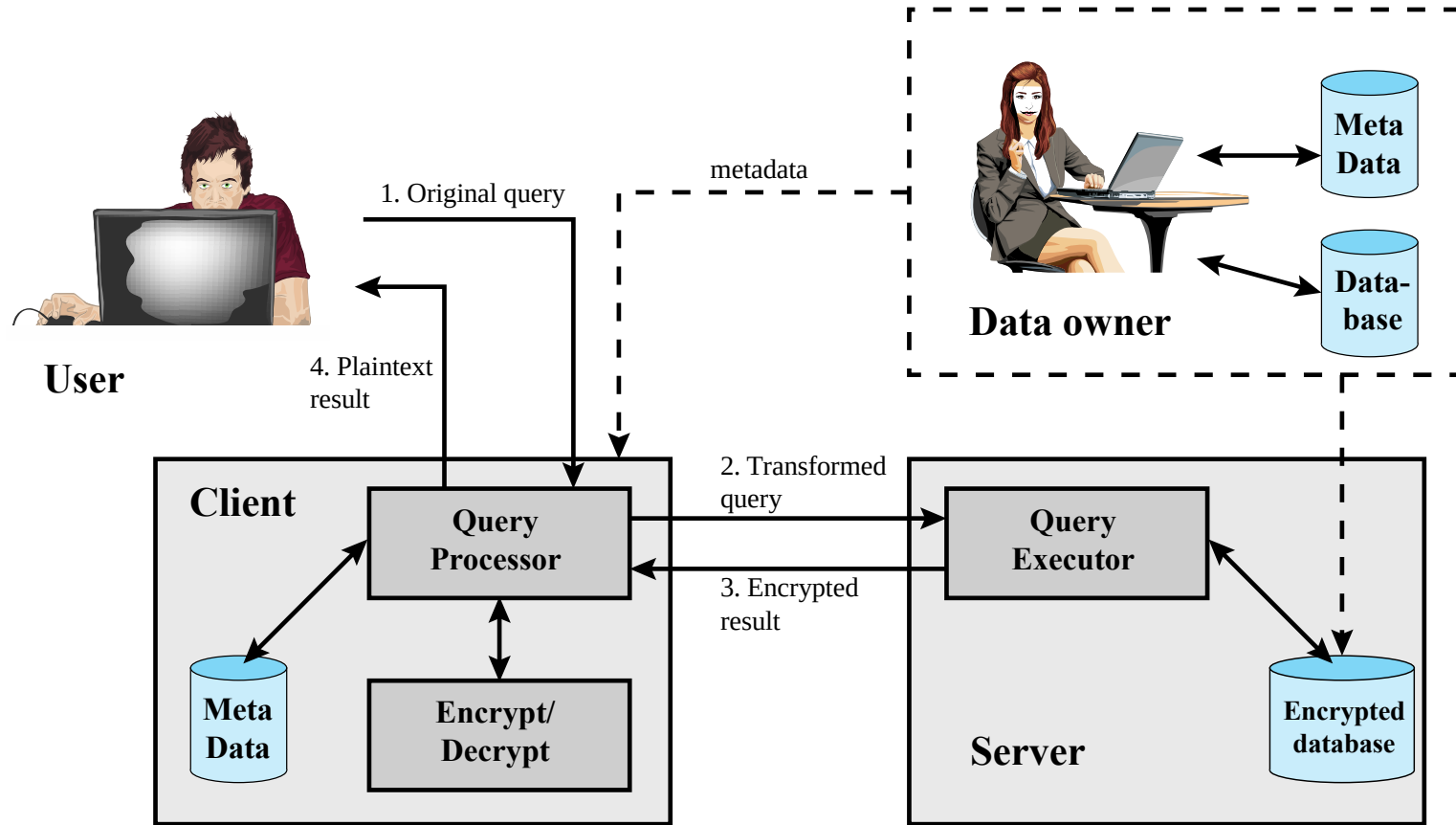


Figure 5.9 A Database Encryption Scheme

$E(k, B_I)$	I_{I1}	$\bullet \bullet \bullet$	I_{Ij}	$\bullet \bullet \bullet$	I_{IM}
$\bullet$	$\bullet$		$\bullet$		$\bullet$
$\bullet$	$\bullet$		$\bullet$		$\bullet$
$\bullet$	$\bullet$		$\bullet$		$\bullet$
$E(k, B_i)$	I_{i1}	$\bullet \bullet \bullet$	I_{ij}	$\bullet \bullet \bullet$	I_{iM}
$\bullet$	$\bullet$		$\bullet$		$\bullet$
$\bullet$	$\bullet$		$\bullet$		$\bullet$
$\bullet$	$\bullet$		$\bullet$		$\bullet$
$E(k, B_N)$	I_{N1}	$\bullet \bullet \bullet$	I_{Nj}	$\bullet \bullet \bullet$	I_{NM}

$$B_i = (x_{i1} \parallel x_{i2} \parallel \dots \parallel x_{iM})$$

Figure 5.10 Encryption Scheme for Database of Figure 5.3

Table 5.3 Encrypted Database Example

(a) Employee Table

eid	ename	salary	addr	did
23	Tom	70K	Maple	45
860	Mary	60K	Main	83
320	John	50K	River	50
875	Jerry	55K	Hopewell	92

(b) Encrypted Employee Table with Indexes

$E(k, B)$	I(eid)	I(ename)	I(salary)	I(addr)	I(did)
1100110011001011...	1	10	3	7	4
011110001111001010...	5	7	2	7	8
1100010010001101...	2	5	1	9	5
0011010011111101...	5	5	2	4	9

Database Security

Lesson Summary

- Used to **store lots of sensitive data** that can be accessed via programs (queries)
 - Access control must be **based on operations allowed by databases**
 - New attacks on databases arise due to their unique characteristics
 - Defenses **must address such attacks**
-