	P.E.S. COLLEGE OF ENGINEERING **(AN AUTONOMOUS INSTITUTE)** **CHH. SAMBHAJINAGAR-431002** **Regular Summer Examination – 2026** **Course: F.Y. M. Tech. Branch : CSE-(Cyber Security) Semester : II** **Subject Code & Name:** MTPESCS203T Database Security **Max Marks: 60 Date:03/07/2026 Duration: 3 Hr.**		
	Instructions to the Students: *1. All the questions are compulsory.* *2. The level of question/expected answer as per OBE or the Course Outcome (CO) on which the question is based is mentioned in ( ) in front of the question.* *3. Use of non-programmable scientific calculators is allowed.* *4. Assume suitable data wherever necessary and mention it clearly.*		
		(Level/CO)	Marks
Q. 1	**Solve Any one of the following.**		
A)	**Define threat modeling and explain the CIA Triad with suitable examples in database security.**	**CO1**	**12**

Database security is an important part of cyber security that protects sensitive organizational data from unauthorized access, modification, and destruction. Two important concepts in database security are **Threat Modeling** and the **CIA Triad**. Threat modeling helps identify possible threats and vulnerabilities in a database system, while the CIA Triad provides the fundamental principles required to secure information.

Threat Modeling

Threat Modeling is a systematic process used to identify, analyze, and reduce security threats and vulnerabilities in a database system or application. It helps organizations understand possible attacks before they occur and apply suitable security controls to minimize risks. The main objective of threat modeling is to protect valuable database assets such as customer information, banking records, passwords, and confidential organizational data.

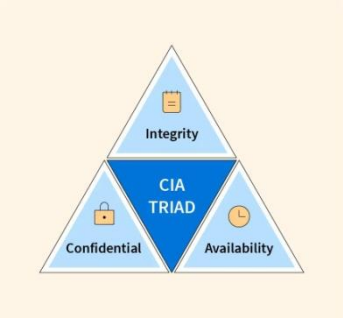
The process of threat modeling includes the following steps:

- **Asset Identification** – Identifying important data and resources that need protection.
- **Threat Identification** – Detecting possible attacks such as SQL Injection, malware attacks, insider threats, and unauthorized access.
- **Vulnerability Identification** – Finding weaknesses like weak passwords, outdated software, and lack of encryption.
- **Risk Analysis** – Evaluating the impact and probability of threats.
- **Mitigation Techniques** – Implementing security measures such as encryption, authentication, firewalls, and access control.

CIA Triad

The CIA Triad is the basic model of information security and consists of three important principles:

1. **Confidentiality**
2. **Integrity**

	3. **Availability** These principles ensure secure storage, processing, and transmission of data in database systems.  **1. Confidentiality** Confidentiality ensures that sensitive information is accessible only to authorized users and prevents unauthorized disclosure of data. It is achieved using methods such as authentication, encryption, strong passwords, and access control mechanisms. **2. Integrity** Integrity ensures that data remains accurate, consistent, and unaltered by unauthorized users. It protects databases from illegal modification, deletion, or corruption of data. Integrity is maintained using hashing, digital signatures, transaction management, and database constraints. **3. Availability** Availability ensures that database services and information are accessible to authorized users whenever required. It protects systems from hardware failures, server crashes, and cyber attacks such as Denial of Service (DoS) attacks. Availability is maintained using backups, disaster recovery systems, redundant servers, and load balancing.		
B)	**Explain various types of database vulnerabilities such as unpatched systems and misconfiguration.** Database vulnerabilities are weaknesses or security flaws present in a database system that can be exploited by attackers to gain unauthorized access, steal sensitive information, modify data, or disrupt services. These vulnerabilities may arise due to improper security practices, outdated software, weak authentication mechanisms, or configuration errors. Identifying and removing vulnerabilities is essential for maintaining database security and protecting organizational data. Some important types of database vulnerabilities are explained below: **1. Unpatched Systems** Unpatched systems are database systems or software applications that have	**CO1**	**12**

not been updated with the latest security patches released by vendors. Attackers often exploit known vulnerabilities present in outdated database software to gain unauthorized access or execute malicious activities.

Causes:

- Failure to install software updates
- Use of outdated DBMS versions
- Poor patch management policies

Risks:

- SQL Injection attacks
- Malware infections
- Data theft
- Remote code execution

Prevention:

- Regular software updates
- Automated patch management
- Vendor security monitoring
- Vulnerability scanning

2. Misconfiguration

Misconfiguration occurs when database security settings are improperly configured, leaving the system exposed to attacks. It is one of the most common causes of database breaches.

Causes:

- Default usernames and passwords
- Improper access permissions
- Open database ports
- Disabled security features

Risks:

- Unauthorized access
- Data leakage
- Privilege escalation
- Exposure of sensitive information

Prevention:

- Change default credentials
- Configure proper access controls
- Disable unnecessary services
- Regular security audits

3. Weak Authentication and Passwords

Weak authentication mechanisms and poor password policies make it easier

for attackers to gain unauthorized access to database systems.

Causes:

- Simple passwords
- Shared user accounts
- Lack of multi-factor authentication

Risks:

- Unauthorized login
- Account compromise
- Insider attacks

Prevention:

- Strong password policies
- Multi-factor authentication
- Regular password changes

4. SQL Injection Vulnerability

SQL Injection occurs when attackers insert malicious SQL queries into input fields to manipulate the database.

Causes:

- Improper input validation
- Dynamic SQL queries
- Insecure coding practices

Risks:

- Data theft
- Unauthorized data modification
- Complete database compromise

Prevention:

- Prepared statements
- Parameterized queries
- Input validation

5. Excessive User Privileges

Excessive privileges occur when users are given more access rights than required for their work.

Risks:

- Unauthorized data modification
- Data deletion
- Insider threats

	Prevention: • Principle of Least Privilege (PoLP) • Role-Based Access Control (RBAC) • Regular privilege review **6. Lack of Data Encryption** If sensitive data is stored or transmitted without encryption, attackers can easily read and misuse the information. **Risks:** • Data leakage • Financial fraud • Privacy violations **Prevention:** • Encrypt sensitive data • Use SSL/TLS protocols • Secure backup encryption		
Q.2	**Solve Any one of the following.**		
A)	**Explain Discretionary Access Control (DAC) with SQL GRANT and REVOKE examples** Discretionary Access Control (DAC) is a security mechanism used in database systems to control access to data and database objects based on user identity and permissions. In DAC, the owner of a database object such as a table, view, or procedure has the authority to decide which users can access the object and what type of operations they can perform. The owner can grant or revoke permissions according to organizational requirements. DAC provides flexibility in managing database security and is commonly used in relational database management systems. In Discretionary Access Control, permissions are assigned directly to users or groups. The access rights may include operations such as SELECT, INSERT, UPDATE, DELETE, EXECUTE, and ALTER. Since the owner has discretionary control over permissions, users with granted privileges can sometimes pass those permissions to other users if allowed. The main features of DAC are: • Access is based on user identity. • Object owners control permissions. • Permissions can be granted or revoked. • Flexible and easy to implement. • Commonly supported in SQL-based database systems.	**CO2**	**12**

The SQL commands used in DAC are:

1. **GRANT Command** – Used to provide privileges to users.
2. **REVOKE Command** – Used to remove privileges from users.

GRANT Command

The GRANT command is used to give access privileges to users on database objects.

Syntax:

```
GRANT privilege_name
ON table_name
TO user_name;
```

Example 1: Grant SELECT Permission

Suppose there is a table named `Employee`.

```
GRANT SELECT
ON Employee
TO Rahul;
```

This command allows user Rahul to view data from the Employee table.

Example 2: Grant Multiple Permissions

```
GRANT SELECT, INSERT, UPDATE
ON Employee
TO Rahul;
```

This command allows Rahul to:

- View records
- Insert new records
- Update existing records

Example 3: Grant Permission with GRANT OPTION

```
GRANT SELECT
ON Employee
TO Rahul
WITH GRANT OPTION;
```

Here, Rahul can also grant SELECT permission to other users.

REVOKE Command

The REVOKE command is used to remove previously granted permissions from users.

Syntax:

```
REVOKE privilege_name
ON table_name
FROM user_name;
```

Example 1: Revoke SELECT Permission

```
REVOKE SELECT
ON Employee
FROM Rahul;
```

This command removes Rahul's permission to view data from the Employee table.

Example 2: Revoke Multiple Permissions

```
REVOKE INSERT, UPDATE
ON Employee
FROM Rahul;
```

This command removes INSERT and UPDATE privileges from Rahul.

B)	**What are the limitations of DAC? Explain the Trojan Horse problem.**	**CO2**	**12**

Discretionary Access Control (DAC) is a commonly used access control mechanism in database systems where the owner of a database object has the authority to grant or revoke access permissions to other users. Although DAC is flexible and easy to implement, it has several security limitations that may expose the database system to threats and unauthorized access.

The major limitations of DAC are explained below:

Limitations of DAC

1. Weak Control over Information Flow

In DAC, users who receive access permissions can further share those permissions with other users if grant options are enabled. This makes it difficult to control the flow of sensitive information within the system.

2. Vulnerability to Trojan Horse Attacks

DAC is highly vulnerable to Trojan Horse attacks because programs executed by authorized users inherit their access privileges. Malicious software can misuse these privileges to steal or modify sensitive data.

3. Lack of Centralized Security Control

Since object owners manage permissions individually, maintaining consistent security policies across large organizations becomes difficult. Different users may apply different access rules, leading to security gaps.

4. Excessive Privilege Distribution

Permissions can spread from one user to another, resulting in excessive access rights. This increases the risk of unauthorized access and insider attacks.

	5. Difficult to Manage in Large Systems In large organizations with many users and database objects, managing permissions individually becomes complex and time-consuming. **6. No Strong Enforcement of Security Policies** DAC relies mainly on user discretion rather than strict organizational policies. Users may unintentionally violate security rules while sharing data. **7. Risk of Insider Threats** Authorized users with legitimate access may intentionally misuse database privileges for personal or malicious purposes. **Trojan Horse Problem in DAC** A Trojan Horse is a malicious program that appears to be useful or legitimate but secretly performs unauthorized activities such as stealing data, modifying files, or bypassing security controls. In Discretionary Access Control systems, Trojan Horse attacks are a major security concern because programs executed by users inherit the user's access permissions. In DAC, if an authorized user runs a malicious program, the program gains the same access rights as the user. As a result, the malicious software can access confidential database information and transfer it to unauthorized users without the knowledge of the legitimate user. **Working of Trojan Horse Attack** 1. An attacker creates a malicious program disguised as useful software. 2. An authorized user executes the program. 3. The program inherits the user's database permissions. 4. The malicious program secretly accesses sensitive information. 5. Data is copied, modified, or transmitted to the attacker.		
Q. 3	**Solve Any one of the following.**		
A)	**What is the inference problem in statistical databases?** The **Inference Problem** in a statistical database occurs when a user is able to deduce or infer confidential information about an individual indirectly from statistical query results, even though direct access to personal records is restricted. In other words, sensitive information is disclosed through analysis of statistical outputs rather than through direct retrieval of data. The inference problem is a major security issue because attackers can combine the results of multiple statistical queries and derive private information about individuals. Although the database only provides aggregate information, carefully designed queries may reveal confidential details.	**CO3**	**12**

Causes of Inference Problem

The inference problem may occur due to the following reasons:

- Small query result sets
- Repeated statistical queries
- Lack of proper query restrictions
- Combination of multiple query results
- Availability of background knowledge to attackers

Types of Inference Attacks

1. Direct Inference

Sensitive information is derived directly from statistical query results.

2. Indirect Inference

Attackers combine multiple query outputs and external knowledge to infer confidential data.

3. Tracker Attacks

Attackers use carefully designed overlapping queries to isolate information about specific individuals.

Risks of Inference Problem

- Violation of individual privacy
- Disclosure of confidential information
- Financial and reputational damage
- Legal and ethical issues
- Loss of trust in database systems

Methods to Prevent Inference Problem

Several techniques are used to protect statistical databases from inference attacks:

1. Query Set Size Control

Restrict queries that involve very small groups of records.

2. Data Suppression

Hide sensitive values or restrict access to certain statistical results.

3. Data Perturbation

Modify statistical outputs slightly by adding noise to the data.

4. Query Restriction

	Limit the number and type of queries users can perform. **5. Access Control Mechanisms** Allow only authorized users to access statistical information. **6. Auditing and Monitoring** Track user queries to detect suspicious activities.		
B)	**What is data perturbation? Explain noise addition techniques.** Data security and privacy are major concerns in statistical databases and data mining systems. Organizations often share statistical or research data for analysis, but sensitive personal information must be protected from unauthorized disclosure. One important technique used for privacy preservation is **Data Perturbation**. Data Perturbation is a privacy-preserving technique in which original data is modified slightly before releasing it for analysis. The purpose of data perturbation is to protect confidential information while still allowing useful statistical analysis. In this technique, the original values are altered in such a way that attackers cannot easily identify sensitive information about individuals. The main objectives of data perturbation are: • Protect sensitive and confidential data • Prevent inference attacks • Preserve user privacy • Allow statistical analysis without revealing exact information Data perturbation is widely used in statistical databases, healthcare systems, research organizations, and data mining applications. **Noise Addition Technique** Noise Addition is one of the most common data perturbation techniques. In this method, random values called **noise** are added to original data values to hide sensitive information. The modified data appears different from the original data, making it difficult for attackers to identify exact values. Mathematically, noise addition can be represented as: $X' = X + N$ Where: • X = Original data value • N = Random noise value • X' = Perturbed data value The added noise is usually small enough so that overall statistical properties such as mean and variance remain approximately unchanged.	**CO3**	**12**

Working of Noise Addition

The process of noise addition includes the following steps:

1. Original sensitive data is selected.
2. Random noise values are generated.
3. Noise is added to the original data.
4. Modified data is released for analysis.

This technique hides exact information while preserving the usefulness of the dataset.

Types of Noise Addition Techniques

1. Additive Noise

In additive noise, random values are directly added to the original data.

Formula:

$X' = X + N$

This method is simple and commonly used in statistical databases.

2. Multiplicative Noise

In multiplicative noise, original data values are multiplied by random noise values.

Formula:

$X' = X \times N$

This technique is useful when proportional modification of data is required.

3. Gaussian Noise Addition

In this method, noise values are generated using Gaussian (normal) distribution. It preserves statistical characteristics more effectively.

Advantages of Noise Addition

- Protects confidential information
- Reduces inference attacks
- Preserves privacy of individuals
- Maintains statistical usefulness of data
- Easy to implement

Disadvantages of Noise Addition

- Excessive noise may reduce data accuracy
- Difficult to balance privacy and data utility
- Statistical results may become slightly inaccurate

Q.4	**Solve Any one of the following.**		

A)	**What is SQL Injection (SQLi)? Explain different variants of SQL injection attacks.**
	
	SQL Injection (SQLi) is one of the most common and dangerous cyber attacks used against database-driven applications. It occurs when an attacker inserts malicious SQL statements into input fields such as login forms, search boxes, or URLs to manipulate the backend database. If the application does not properly validate user input, the attacker can execute unauthorized SQL commands and gain access to sensitive information stored in the database.
	
	SQL Injection attacks mainly target web applications connected to relational databases such as MySQL, Oracle, SQL Server, and PostgreSQL. Through SQL Injection, attackers can bypass authentication, steal confidential data, modify records, delete information, or even gain administrative control over the database system.
	
	The basic cause of SQL Injection is improper input validation and insecure coding practices.
	
	For example, consider the following SQL query used in a login system:
	
	`SELECT * FROM Users`
	`WHERE username = 'admin'`
	`AND password = '1234';`
	
	If an attacker enters the following input in the password field:
	
	`' OR '1'='1`
	
	The query becomes:
	
	`SELECT * FROM Users`
	`WHERE username = 'admin'`
	`AND password = '' OR '1'='1';`
	
	Since the condition `'1'='1'` is always true, the attacker may bypass authentication and gain unauthorized access.
	
	## Variants of SQL Injection Attacks
	
	There are different types of SQL Injection attacks depending on the attack method and database response.
	
	## 1. Tautology-Based SQL Injection
	
	In this attack, the attacker injects code that always evaluates to TRUE. It is commonly used to bypass authentication mechanisms.
	
	## Example:
	
	`' OR '1'='1`
		CO4	**12**

Impact:

- Login bypass
- Unauthorized access to the database

Prevention:

- Parameterized queries
- Input validation

2. Union-Based SQL Injection

In this attack, the attacker uses the SQL `UNION` operator to combine the results of two queries and retrieve data from other database tables.

Example:

```
' UNION SELECT username, password FROM Users --
```

Impact:

- Extraction of confidential data
- Access to sensitive tables

Prevention:

- Restrict database permissions
- Use prepared statements

3. Error-Based SQL Injection

In this variant, attackers intentionally generate database errors to gather information about the database structure, table names, or server details.

Example:

Attackers may enter invalid SQL statements to force the database to reveal error messages.

Impact:

- Disclosure of database schema information
- Easier exploitation of vulnerabilities

Prevention:

- Disable detailed error messages
- Use secure exception handling

4. Blind SQL Injection

Blind SQL Injection occurs when the application does not display database errors directly. Attackers send queries and analyze application responses to

	infer information.
	
	Types:
	
	• Boolean-Based Blind SQLi
	• Time-Based Blind SQLi
	
	Example:
	
	`' AND 1=1 --`
	
	and
	
	`' AND 1=2 --`
	
	Attackers observe changes in webpage responses to identify valid conditions.
	
	Impact:
	
	• Extraction of sensitive data slowly over time
	
	Prevention:
	
	• Input validation
	• Web Application Firewalls (WAF)
	
	5. Time-Based SQL Injection
	
	In this attack, attackers use database delay functions to determine whether injected conditions are true or false.
	
	Example:
	
	`'; WAITFOR DELAY '00:00:05' --`
	
	If the server response is delayed, the attacker confirms vulnerability.
	
	Impact:
	
	• Blind extraction of database information
	
	Prevention:
	
	• Parameterized queries
	• Query timeout restrictions		

B)	**Explain Transparent Data Encryption (TDE). Differentiate between Storage Encryption and Column-level Encryption.** **Transparent Data Encryption (TDE)** Transparent Data Encryption (TDE) is a database security technology that	**CO4**	**12**

automatically encrypts data stored in database files, backup files, and transaction logs without requiring changes to the application. It is called "transparent" because the encryption and decryption processes occur automatically in the background and are invisible to users and applications.

TDE protects data at rest, meaning data stored on disks or backup media remains encrypted. Even if attackers steal database files or storage devices, they cannot read the encrypted information without the encryption keys.

The working of TDE involves the following steps:

1. Data is stored in the database.
2. Before writing data to disk, the database engine encrypts it automatically.
3. When authorized users access the data, the database decrypts it automatically in memory.
4. Users and applications can access data normally without knowing the encryption process.

Difference Between Storage Encryption and Column-Level Encryption

Storage Encryption and Column-Level Encryption are two important methods used to secure database data. However, they differ in scope, functionality, and level of protection.

Basis	Storage Encryption	Column-Level Encryption
Definition	Encrypts the entire database storage or files	Encrypts specific columns containing sensitive data
Level of Encryption	Database or disk level	Individual column level
Protection Scope	Entire database files, logs, backups	Selected sensitive fields only
Transparency	Usually transparent to applications	May require application changes
Performance Impact	Lower performance impact	Higher processing overhead
Flexibility	Less flexible	More flexible and selective
Data Visibility	Authorized DB users can view data after login	Even DB administrators may not view encrypted columns
Example	TDE encrypting full database files	Encrypting credit card or Aadhaar number columns
Security Level	Protects against physical theft	Provides finer security for sensitive data
Usage	General database protection	Highly confidential information protection

Q. 5	**Solve Any one of the following.**		

A)	**Explain the Shared Responsibility Model in cloud database security and compare security responsibilities in IaaS, PaaS, and SaaS cloud models.**
	
	The Shared Responsibility Model is a cloud security framework that divides security responsibilities between the cloud service provider and the customer. The cloud provider is responsible for securing the cloud infrastructure, while the customer is responsible for securing their data, user access, applications, and configurations depending on the cloud service model being used.
	
	In cloud database security, both the provider and customer work together to protect databases from threats such as unauthorized access, data breaches, malware attacks, and misconfigurations.
	
	## Shared Responsibility Model
	
	In the Shared Responsibility Model:
	
	• The **Cloud Service Provider (CSP)** secures:
	  ○ Physical data centers
	  ○ Hardware
	  ○ Networking
	  ○ Virtualization infrastructure
	  ○ Cloud platform services
	• The **Customer** secures:
	  ○ User accounts
	  ○ Database access permissions
	  ○ Data encryption
	  ○ Application security
	  ○ Operating system configurations (depending on service type)
	
	The level of customer responsibility changes based on whether the cloud service is IaaS, PaaS, or SaaS.
	
	## 1. Infrastructure as a Service (IaaS)
	
	Infrastructure as a Service (IaaS) provides virtualized computing resources such as servers, storage, and networking over the internet. The cloud provider manages the physical infrastructure, while the customer manages most other components.
	
	### Provider Responsibilities:
	
	• Physical security
	• Servers and storage
	• Network infrastructure
	• Virtualization layer
	
	### Customer Responsibilities:
	
	• Operating system security
	• Database installation and configuration
	• Access control
		CO5	**12**

	• Data encryption • Application security • Security updates and patches **2. Platform as a Service (PaaS)** Platform as a Service (PaaS) provides a platform for developing and deploying applications without managing underlying hardware or operating systems. **Provider Responsibilities:** • Infrastructure security • Operating system maintenance • Runtime environment • Middleware and platform security **Customer Responsibilities:** • Application security • Database access management • User authentication • Data protection • Secure coding practices **3. Software as a Service (SaaS)** Software as a Service (SaaS) provides complete software applications over the internet. The cloud provider manages almost the entire environment. **Provider Responsibilities:** • Infrastructure security • Application security • Database management • Updates and patching • Backup and recovery **Customer Responsibilities:** • User management • Password security • Data sharing permissions • Account configuration		
B)	**Describe security models used in NoSQL databases. What are authentication and access control challenges in distributed NoSQL architectures?** **Security Models Used in NoSQL Databases**	**CO5**	**12**

1. Role-Based Access Control (RBAC)

Role-Based Access Control is one of the most commonly used security models in NoSQL databases. In RBAC, permissions are assigned to roles rather than directly to users. Users receive permissions according to their assigned roles.

Example:

- Administrator → Full database access
- Developer → Read and write access
- Analyst → Read-only access

Advantages:

- Easy permission management
- Reduces excessive privileges
- Improves security administration

MongoDB and Cassandra commonly use RBAC mechanisms.

2. Attribute-Based Access Control (ABAC)

In Attribute-Based Access Control, access decisions are based on attributes such as:

- User role
- Location
- Device type
- Time of access
- Data sensitivity level

Example:

A user may access customer records only during office hours from the company network.

Advantages:

- Fine-grained access control
- Flexible security policies
- Better control in distributed systems

3. Mandatory Access Control (MAC)

Mandatory Access Control is a highly secure model where access permissions are controlled centrally by administrators according to security classifications.

Example:

- Top Secret
- Secret

- Confidential
- Public

Users cannot modify permissions themselves.

Advantages:

- Strong security enforcement
- Suitable for military and government systems

4. Discretionary Access Control (DAC)

In DAC, database object owners decide who can access specific data objects and what operations can be performed.

Example:

A user may grant read access on a document collection to another user.

Advantages:

- Flexible permission management
- Easy implementation

Limitation:

- Vulnerable to Trojan Horse attacks

5. Encryption-Based Security Model

Many NoSQL databases use encryption to secure:

- Data at rest
- Data in transit
- Backup files

Techniques Used:

- SSL/TLS communication
- AES encryption
- Transparent Data Encryption (TDE)

Authentication Challenges in Distributed NoSQL Databases

Authentication verifies the identity of users before granting access.

Challenges:

- Large number of distributed nodes
- Weak default security settings
- Scalability issues
- Lack of standard authentication standards

	• Complex cloud and API integration **Access Control Challenges in Distributed NoSQL Databases** Access control ensures that only authorized users can access data. **Challenges:** • Difficulty in fine-grained access control • Data replication across multiple servers • Dynamic addition of nodes • Multi-tenant cloud environments • Insider threats		
	***** End *****		