# Database Security

**Database security safeguards sensitive information from unauthorized access and misuse. It ensures data confidentiality, integrity, and availability.**

Database Security involves a set of practices, tools, and processes to protect database systems

This lecture explores key concepts, threats, and protective measures in database security.

Watch the video lectures here: https://youtu.be/2Bi6n5QFHj8

**by Afjal H Sarower**

**Example**: Consider a bank’s database storing sensitive customer information. If unauthorized users gain access, it could lead to financial and reputational losses.

# Introduction to Database Security

1

## Confidentiality

Restrict access to authorized users only. Encryption and access controls protect sensitive data.

2

## Integrity

Prevent unauthorized modifications. Checksums and digital signatures ensure data authenticity.

3

## Availability

Ensure consistent access for authorized users. Redundancy and backups maintain data availability.

by Afjal H Sarower

# Types of Database Security Threats

## Internal Threats

Employees misusing privileges. Insider trading or data theft can occur.

## External Threats

Hackers exploiting vulnerabilities. SQL injection attacks are common external threats.

## Operational Threats

Misconfigurations exposing data. Unpatched systems can lead to vulnerabilities.

by Afjal H Sarower

# Access Control Mechanisms

1

## Authentication

Verify user identity. Passwords, biometrics, or multi-factor authentication are common methods.

2

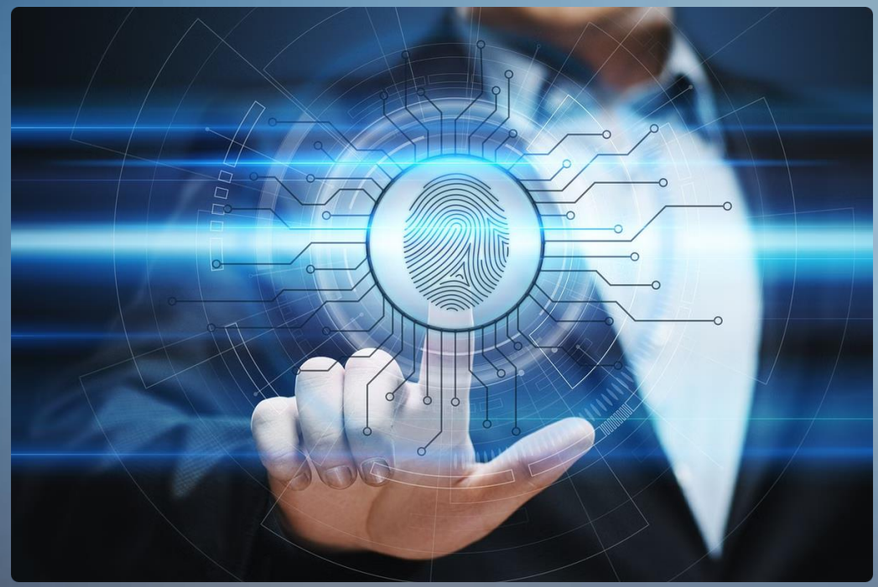
## Authorization

Determine access levels. Role-Based Access Control (RBAC) assigns permissions based on roles.

3

## Privileges and Permission

Privileges and permissions are specific access rights granted to users or roles,



by Afjal H Sarower

## Access Control Mechanisms

- Role-Based Access Control (RBAC)
- Discretionary Access Control (DAC)
- Mandatory Access Control (MAC)
- Attribute-Based Access Control (ABAC)

# Role-based Access Control

**Example**: For an e-commerce database, customers can only access their orders, while admin users have broader access.

SQL Code for User Privileges:

```sql
-- Grant SELECT privilege to user 'john'
GRANT SELECT ON Orders TO john;

-- Revoke privilege
REVOKE SELECT ON Orders FROM john;
```

by Afjal H Sarower

# Role-Based Access Control

```sql
-- Create roles for Admin and Customer
CREATE ROLE Admin;
CREATE ROLE Customer;

-- Grant privileges to roles
GRANT ALL PRIVILEGES ON Orders TO Admin;
GRANT SELECT ON Orders TO Customer;

-- Assign roles to users
GRANT Admin TO 'admin_user'@'localhost';
GRANT Customer TO 'customer_user'@'localhost';
```

by Afjal H Sarower

# Mandatory Access Control

**Example**: In a government database, access to classified data is restricted based on user clearance levels. For instance, only users with “Top Secret” clearance can access certain tables.

While MAC is less common in SQL databases (typically used in OS-level or secure government environments), it may be implemented by classifying data at the application layer and enforcing rules via stored procedures or application logic.

# Discretionary Access Control

**Example**: In a research database, each researcher has control over their data and can share it with other specific researchers.

```
-- Researcher 'alice' creates a table and grants access to 'bob'
CREATE TABLE alice_data (id INT, research_note TEXT);
GRANT SELECT, INSERT ON alice_data TO 'bob'@'localhost';
```

Privileges and permissions are specific access rights granted to users or roles, such as **SELECT**, **INSERT**, **UPDATE**, or **DELETE**.

```
-- Grant SELECT privilege to manager
GRANT SELECT ON Sales TO 'manager'@'localhost';

-- Grant all privileges to sales_rep
GRANT SELECT, INSERT, UPDATE, DELETE ON Sales TO 'sales_rep'@'localhost';
```

by Afjal H Sarower

# Database Encryption

| | | |
|---|---|---|
| **At-Rest Encryption** | Protects stored data | Full disk encryption, transparent data encryption |
| **In-Transit Encryption** | Secures data during transfer | SSL/TLS protocols, VPN tunnels |
| **Column-Level Encryption** | Encrypts specific fields | AES encryption for sensitive columns |

by Afjal H Sarower

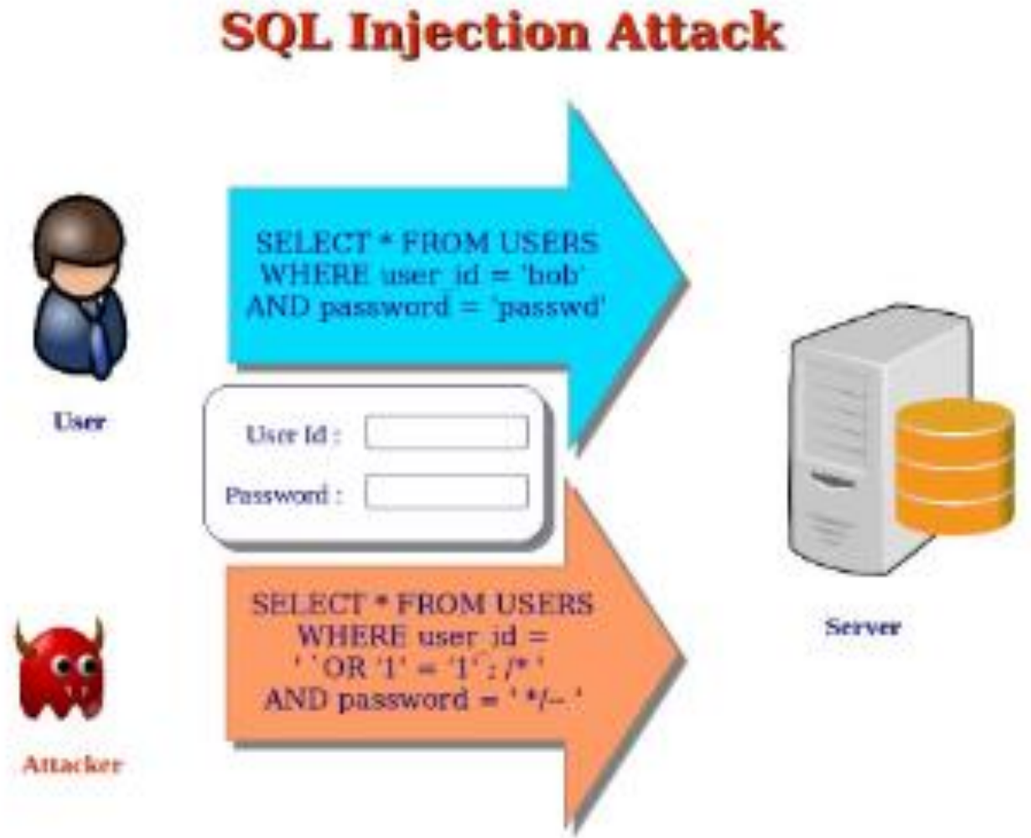
# SQL Code for Column-Level Encryption (Example in MySQL)

```sql
-- Encrypt column data
UPDATE Customers
SET CreditCard = AES_ENCRYPT('1234-5678-9012-3456', 'encryption_key');

-- Decrypt column data
SELECT AES_DECRYPT(CreditCard, 'encryption_key') FROM Customers;
```

by Afjal H Sarower

# SQL Injection Prevention




1

## Input Validation

Sanitize user inputs. Remove special characters and validate data types.

2

## Parameterized Queries

Use query placeholders. Separate SQL logic from user-supplied data.

```
cursor. execute("SELECT * FROM Users WHERE username = ? AND password = ?", (username, password))
```

3

## Least Privilege

Limit database account permissions. Restrict access to only necessary operations.

by Afjal H Sarower

# Database Auditing and Monitoring

## Log Analysis

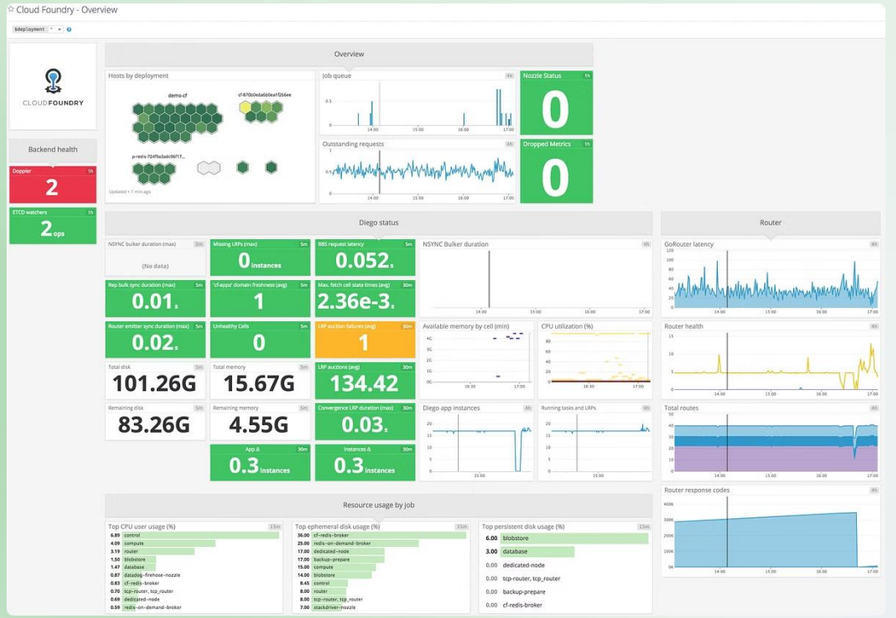
Review database logs regularly. Detect suspicious activities and access patterns.

## Real-time Alerts

Set up automated notifications. Respond quickly to potential security breaches.

## Compliance Reporting

Generate audit reports. Ensure adherence to regulatory requirements and internal policies.




by Afjal H Sarower


# Emerging Trends in Database Security

## Cloud Security

Secure multi-tenant databases. Implement strong encryption and access controls in cloud environments.

## Privacy-Preserving Techniques

Use homomorphic encryption. Perform computations on encrypted data without decryption.

## AI-Driven Security

Employ machine learning for anomaly detection. Identify unusual patterns in database access.

by Afjal H Sarower

# Thank You

**Afjal H Sarower**