Lesson 9: Security Policy and Audit

Database Security Fundamentals

Database security is a critical aspect of managing and protecting data within a database management system (DBMS). Ensuring the security of a database involves implementing various measures to protect data integrity, confidentiality, and availability.

The foundation of database security is built on the principles of the CIA triad: Confidentiality, Integrity, and Availability.

> **Confidentiality:** Ensures that sensitive information is accessible only to authorized users and is protected from unauthorized access. This principle is implemented through encryption, access controls, and data masking techniques.
>
> **Integrity:** Ensures the accuracy and consistency of data over its lifecycle. Integrity is maintained by enforcing data validation rules, using checksums, and implementing audit trails to track changes and detect unauthorized modifications.
>
> **Availability:** Ensures that data and database services are available to authorized users when needed. Availability is maintained through redundancy, failover mechanisms, regular backups, and performance tuning to prevent system downtime and ensure reliable access to data.

Adhering to these principles helps protect the database from various threats and ensures that data remains secure and reliable.

User Authentication Methods

User authentication is the process of verifying the identity of users attempting to access the database. There are several methods of user authentication, each providing different levels of security.

> **Password-based Authentication:** The most common method, where users provide a username and password to gain access. Strong passwords and regular password changes are essential to enhance security.

Multi-factor Authentication (MFA): Adds an additional layer of security by requiring users to provide two or more verification factors, such as a password and a one-time code sent to their mobile device. MFA significantly reduces the risk of unauthorized access.

Biometric Authentication: Uses unique biological characteristics, such as fingerprints or facial recognition, to verify user identity. Biometric authentication provides a high level of security but requires specialized hardware.

Certificate-based Authentication: Uses digital certificates issued by a trusted certificate authority to verify user identity. This method is commonly used in secure environments where strong authentication is required.

Implementing robust user authentication methods helps ensure that only authorized users can access the database.

Role-based Access Control (RBAC)

Role-based access control (RBAC) is a method of restricting database access based on the roles assigned to users. In RBAC, permissions are associated with roles rather than individual users, simplifying the management of user privileges.

Define Roles: Identify and define roles that represent different job functions or responsibilities within the organization. Each role is granted a set of permissions required to perform its duties.

Assign Roles to Users: Assign one or more roles to each user based on their responsibilities. Users inherit the permissions associated with their roles.

Manage Role Permissions: Regularly review and update role permissions to ensure they align with current job functions and organizational policies.

For example, in an Oracle database, you can create a role and grant it permissions as follows:

```
CREATE ROLE data_entry;
GRANT INSERT, UPDATE ON employees TO data_entry;
GRANT data_entry TO user1;
```

In this example, the **data_entry** role is created with **INSERT** and **UPDATE** permissions on the **employees** table, and then the role is assigned to **user1**.

Security Policies and Standards

Security policies and standards provide a framework for managing and enforcing database security. These policies define the rules and procedures for protecting data and ensure compliance with legal and regulatory requirements.

Develop Security Policies: Create comprehensive security policies that cover all aspects of database security, including access controls, data protection, incident response, and auditing. Policies should be clear, enforceable, and regularly reviewed.

Implement Security Standards: Adopt industry-standard security practices and frameworks, such as ISO/IEC 27001, NIST Cybersecurity Framework, and CIS Benchmarks, to ensure a consistent and effective approach to database security.

Conduct Regular Audits: Perform regular security audits to assess compliance with security policies and standards. Audits help identify vulnerabilities and areas for improvement, ensuring that security measures are effective and up-to-date.

Provide Security Training: Educate employees and users about database security best practices and policies. Regular training helps raise awareness and reduces the risk of security breaches caused by human error.

Database security is essential for protecting sensitive data and ensuring the reliability of database systems. By adhering to the principles of the CIA triad, implementing robust user authentication methods, using role-based access control, and developing comprehensive security policies and standards, database administrators can effectively safeguard their databases against various threats. Ensuring database security requires

ongoing vigilance, regular updates, and a commitment to best practices to maintain the integrity, confidentiality, and availability of data.

User Management and Access Control

Effective user management and access control are essential components of database administration, ensuring that only authorized users can access and manipulate data according to their roles and responsibilities. This chapter covers creating and managing database users, assigning roles and privileges, implementing fine-grained access control, and auditing user activities and session management.

Creating and Managing Database Users

Creating and managing database users involves defining user accounts, setting authentication methods, and configuring user properties. A well-defined user management process helps maintain security and streamline database administration.

In Oracle, creating a user is straightforward. The CREATE USER statement is used to define a new user and set their authentication method. For example:

```
CREATE USER john_doe IDENTIFIED BY password123;
```

This command creates a user named **john_doe** with the password **password123**. Once a user is created, you can manage their properties and privileges, such as setting default tablespaces, quotas, and profile limits.

```
ALTER USER john doe DEFAULT TABLESPACE users QUOTA 100M ON users PROFILE default;
```

This command sets **users** as the default tablespace for **john_doe**, allocates 100 MB of space in that tablespace, and assigns the **default** profile to manage resource limits.

Assigning Roles and Privileges

Roles and privileges are crucial for controlling user access to database objects and operations. Privileges define specific permissions, such as the ability to select, insert,

update, or delete data in a table, while roles are collections of privileges that can be assigned to users.

Creating a role and assigning privileges can simplify user management. For example, to create a role and grant it privileges:

```
CREATE ROLE data_entry;
GRANT INSERT, UPDATE ON employees TO data_entry;
```

In this example, the **data_entry** role is created and granted **INSERT** and **UPDATE** permissions on the **employees** table. To assign this role to a user:

```
GRANT data_entry TO john_doe;
```

Now, **john_doe** inherits the privileges associated with the **data_entry** role. This approach simplifies the management of user privileges, especially when multiple users share similar access requirements.

Implementing Fine-grained Access Control

Fine-grained access control (FGAC) allows more precise control over what data users can access within a database. This technique goes beyond standard privileges by enabling row-level security, where users can only see or manipulate data that they are authorized to access.

Oracle Virtual Private Database (VPD) is a feature that provides FGAC. It uses policies to enforce security rules dynamically at the database level. For example, to create a VPD policy that restricts access to rows based on the department:

1. **Create a Policy Function:** Define a function that returns the security predicate.

```
CREATE OR REPLACE FUNCTION dept_sec (p_schema VARCHAR2, p_object VARCHAR2)
RETURN VARCHAR2 AS
BEGIN
   RETURN 'department_id = ' || SYS_CONTEXT('USERENV', 'SESSION_USER');
END;
```

2. **Apply the Policy:** Use the **DBMS_RLS** package to associate the policy function with a table.

```
BEGIN
   DBMS_RLS.ADD_POLICY(
      object_schema => 'hr',
      object_name   => 'employees',
      policy_name   => 'dept_policy',
      function_schema=> 'hr',
      policy_function=> 'dept_sec'
   );
END;
```

In this example, the **dept_policy** is applied to the **employees** table, restricting access based on the **department_id** of the session user.

Auditing User Activities and Session Management

Auditing user activities and session management are critical for maintaining database security and compliance. Auditing helps track user actions, detect suspicious activities, and ensure that users follow security policies.

Oracle provides comprehensive auditing capabilities through the use of audit trails. To enable auditing for specific actions, use the AUDIT statement. For example, to audit all logins and logouts:

```
AUDIT SESSION;
```

This command enables auditing of user sessions, recording each login and logout event. The audit records can be viewed in the **DBA_AUDIT_TRAIL** view.
For more detailed auditing, such as tracking changes to specific tables, use:

```
AUDIT INSERT, UPDATE, DELETE ON employees BY ACCESS;
```

This command audits **INSERT**, **UPDATE**, and **DELETE** operations on the **employees** table, recording each access to the table.

Session management involves monitoring and controlling user sessions to prevent unauthorized access and ensure efficient resource utilization. Oracle provides several tools and views, such as **V$SESSION**, to monitor active sessions. Administrators can terminate sessions using the **ALTER SYSTEM KILL SESSION** command if necessary:

```
ALTER SYSTEM KILL SESSION 'sid,serial#';
```

Replace **sid** and **serial#** with the session identifier values from the **V$SESSION** view.

User management and access control are vital for securing a database and ensuring that users can only access the data they are authorized to see and manipulate. By creating and managing users, assigning roles and privileges, implementing fine-grained access control, and auditing user activities, database administrators can maintain a secure and well-organized database environment. Effective user management and access control help prevent unauthorized access, ensure compliance with security policies, and provide a clear audit trail for monitoring and responding to security incidents.

Auditing Database Activities

Auditing database activities is a crucial aspect of database security and management. It involves monitoring and recording user actions and changes within the database to

ensure compliance with policies, detect potential security issues, and maintain the integrity of the data. The primary purpose of auditing is to ensure that all actions within the database are recorded and can be reviewed to maintain security, compliance, and operational integrity. Auditing serves several key purposes, including security monitoring to detect unauthorized access and potential security breaches, ensuring compliance with internal policies and external regulations such as GDPR, HIPAA, and SOX, establishing accountability by tracking user actions, helping in troubleshooting by providing a detailed history of actions and changes within the database, and analyzing user activity patterns to optimize database performance and resource utilization. By implementing a robust auditing system, organizations can protect sensitive data, ensure regulatory compliance, and improve overall database management.

Auditing can be broadly classified into standard auditing and fine-grained auditing (FGA). Standard auditing captures general database activities, such as user logins, schema modifications, and access to sensitive data. It is typically configured to audit specific actions or objects at a high level and is suitable for capturing broad activity patterns and ensuring compliance with basic security policies. For instance, to audit all **SELECT, INSERT, UPDATE**, and **DELETE** operations on the employees table, you can use the command **AUDIT SELECT, INSERT, UPDATE, DELETE ON employees BY ACCESS;.** Fine-grained auditing (FGA), on the other hand, provides more detailed auditing by capturing specific conditions and contexts under which database actions occur. FGA allows for auditing based on user-defined policies, which can include complex conditions involving column values and context variables. For example, to audit access to the **salary** column in the employees table when the salary value is greater than 10,000, you can use the **DBMS_FGA.ADD_POLICY** procedure.

Configuring and enabling auditing involves setting up audit policies and parameters to capture the desired activities. In Oracle, this process includes defining audit settings in the **AUDIT_TRAIL** parameter and using the **AUDIT** statement or **DBMS_FGA** package. To enable the audit trail, you can set the **AUDIT_TRAIL** parameter to determine where audit records are stored, such as **DB** for database auditing and **DB, EXTENDED** for additional details, with the command **ALTER SYSTEM SET AUDIT_TRAIL = DB, EXTENDED SCOPE = SPFILE;**. This requires a database restart to take effect. Next, create audit policies using the **AUDIT** statement for standard auditing and the **DBMS_FGA** package for fine-grained auditing. For example, to audit all actions by users in the **hr** schema, use **AUDIT ALL BY hr BY ACCESS;**, and for fine-grained auditing, use **DBMS_FGA.ADD_POLICY** to add specific policies.

Reviewing and managing audit logs is essential for maintaining database security and ensuring compliance. Audit logs provide a record of audited activities and can be

analyzed to detect unusual behavior, ensure compliance, and troubleshoot issues. Access audit logs using views such as **DBA_AUDIT_TRAIL**, **DBA_FGA_AUDIT_TRAIL**, and **V$XML_AUDIT_TRAIL**. For example, to retrieve records of all login actions from the audit trail, you can use the query **SELECT * FROM DBA_AUDIT_TRAIL WHERE ACTION_NAME = 'LOGIN';**. Regularly review audit logs to identify suspicious activities, unauthorized access, and compliance violations, and use automated tools and scripts to assist in the analysis. Implement a strategy for managing audit logs, including regular archiving, purging old records, and ensuring that logs are securely stored, to maintain database performance and ensure that audit logs are available for compliance and forensic analysis. Develop procedures for responding to findings from audit log analysis, including investigating incidents, reporting to relevant authorities, and taking corrective actions to prevent future occurrences.

In conclusion, auditing database activities is essential for ensuring security, compliance, and operational integrity. By understanding the purpose and importance of auditing, configuring standard and fine-grained audits, and effectively reviewing and managing audit logs, database administrators can protect sensitive data, detect potential threats, and maintain a secure database environment. Regular auditing and diligent management of audit logs help organizations meet regulatory requirements, improve accountability, and enhance overall database security.