Tutorial 4

Q1. What is the difference between authentication and authorization in database systems?

Answer: Authentication verifies who you are while authorization determined what you can do after authentication

Q2. Explain the principle of “Least Privilege” and how it is applied in RBAC.

Answer: users should have only the minimum permissions required for their task, create roles and assign users to the roles , never grant direct object permissions

Q3. What is a SQL Server Login vs. a Database User?

Answer: SQL server login allows you to authenticates connection to server created with create login , database user maps its login to a database.

Q4. Why is permission management critical for database security?

Answer: Pernission management prevents data breaches , data corruption and privilege escalation

Q5. Describe the role of the db\_datareader and db\_datawriter roles in SQL Server.

Answer: data reader is a built in role with select permissions on all tables/view in the database. Data writer built in role is with insert update delete on all tables but not select.

Q6. How does Row-Level Security (RLS) restrict access in SQL Server?

Answer: RLS uses predicates inline functions to dynamically filter rows based on user context

Q7. What are the steps to implement Role-Based Access Control (RBAC) in SQL Server?

Answer: Create Logins, Create Database, Use Sales DB to create user for login, create role, grant select on sales. Order for sales analyst.

Q8. What is an Authorization Matrix?

Answer: It is table mapping users/roles to permissions on objects

Q9. What is Dynamic Data Masking (DDM)? How does it differ from encryption?

Answer: DDM only hides data at query time and it remains unencrypted on isc, encryption is for storage protection

Q10. Why is using Views a good alternative to direct table access when applying permission controls?

Answer, in order to hide sensitive columns, it can also be simplifying the permissions

Q11. Create a SQL login and user for ‘audit\_officer’ with a secure password. Then, restrict this user to only view data from the AuditLog table using a custom role.

Answer: -- Create Login (server-level)

CREATE LOGIN audit\_officer WITH PASSWORD = 'Audit@Pass!2024',

CHECK\_POLICY = ON;

Create Database User

USE AuditDB;

CREATE USER audit\_officer\_user FOR LOGIN audit\_officer;

Create Custom Role

CREATE ROLE AuditViewers;

GRANT SELECT ON AuditLog TO AuditViewers; -- Only access to AuditLog

Assign User to Role

EXEC sp\_addrolemember 'AuditViewers', 'audit\_officer\_user';

Q12. Create a user-defined role called DataOperators. Grant it INSERT and UPDATE access to all tables under the Sales schema.

USE SalesDB;

CREATE ROLE DataOperators;

GRANT INSERT, UPDATE ON SCHEMA::Sales TO DataOperators;

Q13. Deny DELETE permission on the Sales.Orders table to the DataOperators role. Explain why you would do this.

Answer: DENY DELETE ON Sales.Order TO DataOperator

Q14. Create a view named HRView that shows only the Name, Email, and Department columns from the HR.Employees table. Grant access to HRClerks role.

Answer: CREATE VIEW HRView AS

SELECT Name, Email, Department FROM HR.Employees;

GRANT SELECT ON HRView TO HRClerks;

Q15. Create a stored procedure GetEmployeeInfo that returns data from the HR.Employees table and shows only data for the user executing it. Use USER\_NAME() to filter data.

Answer: CREATE PROCEDURE GetEmployeeInfo

AS

BEGIN

SELECT \* FROM HR.Employees

WHERE CreatedBy = USER\_NAME(); -- Filters to current user's data

END

Q16. Mask the Salary column of the Finance.Staff table using the default() masking function. Also, mask the ICNumber column using partial() masking.

-- Mask Salary with default() (e.g., 0 for numbers)

ALTER TABLE Finance.Staff

ALTER COLUMN Salary ADD MASKED WITH (FUNCTION = 'default()');

-- Mask ICNumber with partial() (e.g., XXXX-XX-1234 → XXXX-XX-XXXX)

ALTER TABLE Finance.Staff

ALTER COLUMN ICNumber ADD MASKED WITH (FUNCTION = 'partial(0,"XXXX",0)');

Q17. Create a security predicate function for RLS that filters data based on CreatedBy column. Then apply it to the Projects table.

-- 1. Create Predicate Function

CREATE FUNCTION dbo.fn\_ProjectAccess(@CreatedBy NVARCHAR(128))

RETURNS TABLE WITH SCHEMABINDING

AS RETURN SELECT 1 AS AccessAllowed

WHERE @CreatedBy = USER\_NAME() OR IS\_MEMBER('ProjectManagers') = 1;

-- 2. Apply to Projects table

CREATE SECURITY POLICY ProjectPolicy

ADD FILTER PREDICATE dbo.fn\_ProjectAccess(CreatedBy) ON dbo.Projects;

Q18. Grant a user john\_mgr SELECT access to the Sales.SensitiveDeals table but deny access to the DealValue column.

GRANT SELECT ON Sales.SensitiveDeals TO john\_mgr;

DENY SELECT (DealValue) ON Sales.SensitiveDeals TO john\_mgr; -- Column-level deny

Q19. Create an asymmetric key named HRKey and grant CONTROL access to the HRManagers role for decryption.

CREATE ASYMMETRIC KEY HRKey WITH ALGORITHM = RSA\_2048;

GRANT CONTROL ON ASYMMETRIC KEY::HRKey TO HRManagers; -- Allows decryption

Q20. Write a query to list all masked columns in the current database and the associated masking functions.

SELECT

t.name AS Table,

c.name AS Column,

c.is\_masked,

c.masking\_function

FROM sys.masked\_columns AS c

JOIN sys.tables AS t ON c.[object\_id] = t.[object\_id];