Principals (Part-1) : Server Logins, Server Roles

SQL Server Security Components

- SQL Server provides three types of components for controlling. Which users can log onto SQL Server. What data they can access. Which operations they can carry out.
- Principals
- Securables
- Permissions
- Together, these three component types provide a structure for authenticating and authorizing SQL Server users.
- You must grant each principal the appropriate permissions it needs on specific securables to enable users to access SQL Server resources.
- If the sqluser01 database user needs to be able to query data in the Sales schema, you can grant the SELECT permission to that user on the schema. The user would then be able to query each table and view within the schema.

Server Logins

- SQL Server supports four types of logins: Windows, SQL Server, certificate-mapped, and asymmetric key-mapped.
- Windows and SQL Server logins.
- We use CREATE LOGIN statement to define several logins.
- Because logins exist at the server level, you must create them within the context of the master database.
- A Windows login is associated with a local Windows account or domain account. When you create the login, you must specify the Windows account, preceded by the computer name or domain name and a backslash.
    - USE master;

      CREATE LOGIN [win10b\winuser01] FROM WINDOWS

      WITH DEFAULT_DATABASE = master, DEFAULT_LANGUAGE = us_english;

Server Logins (contd..)

- SQL Server supports four types of logins: Windows, SQL Server, certificate-mapped, and asymmetric key-mapped.
- Windows and SQL Server logins.
- We use CREATE LOGIN statement to define several logins.
- Because logins exist at the server level, you must create them within the context of the master database.
- A Windows login is associated with a local Windows account or domain account. When you create the login, you must specify the Windows account, preceded by the computer name or domain name and a backslash.
    - USE master;

      CREATE LOGIN [win10b\winuser01] FROM WINDOWS

      WITH DEFAULT_DATABASE = master, DEFAULT_LANGUAGE = us_english;
- The statement must include the FROM WINDOWS clause to indicate that this is a Windows login. In this case, the statement also includes an optional WITH clause, which specifies a default database and language.
-

Server Logins (contd..)

- If you’re creating a login based on a domain account, replace the computer name with the domain name, following the same format: [<domain_name>\<windows_account>].
- Also use this format if creating a login based on a Windows group.
  - CREATE LOGIN [win10b\wingroup01] FROM WINDOWS

    WITH DEFAULT_DATABASE = master, DEFAULT_LANGUAGE = us_english;
- By creating a login based on a group, you can provide the same level of access to any user within that group, while letting Windows and SQL Server handle authenticating and authorizing the individual users.
- use the CREATE LOGIN statement to define a SQL Server login (one that is not associated with a Windows account), in which case, do not include the FROM WINDOWS clause. However, you must include a WITH clause that specifies a password.
  - CREATE LOGIN sqluser01

    WITH PASSWORD = 'tempPW@56789' MUST_CHANGE, CHECK_EXPIRATION = ON, DEFAULT_DATABASE = master, DEFAULT_LANGUAGE = us_english;

Server Logins (contd..)

- Use the GRANT statement to grant permissions to that login.
  - GRANT IMPERSONATE ANY LOGIN TO [win10b\winuser01], sqluser01;
- Use the sys.server_principals and sys.server_permissions catalog views to verify that the permissions have been configured correctly.
  - SELECT pr.principal_id, pr.name, pe.state_desc, pe.permission_name FROM sys.server_principals pr INNER JOIN sys.server_permissions pe

    ON pr.principal_id = pe.grantee_principal_id WHERE pr.principal_id = SUSER_ID('win10b\winuser01') OR pr.principal_id = USER_ID('sqluser01');

Server Roles

- SQL Server provides server-level roles to help you manage the permissions on a server.
- These roles are security principals that group other principals.
- Server-level roles are server-wide in their permissions scope.
- SQL Server supports fixed server roles and user-defined server roles.
- You can assign logins to a fixed server role, but you cannot change its permissions.
- You can do both with a user-defined server role.

Fixed Server Roles

Fixed Server - Level Role	Description
sysadmin	Members of the **sysadmin** fixed server role can perform any activity in the server.
serveradmin	Members of the **serveradmin** fixed server role can change server-wide configuration options and shut down the server.
securityadmin	Members of the **securityadmin** fixed server role manage logins and their properties. They can GRANT, DENY, and REVOKE server-level permissions. They can also GRANT, DENY, and REVOKE database-level permissions if they have access to a database. Additionally, they can reset passwords for SQL Server logins.
processadmin	Members of the **processadmin** fixed server role can end processes that are running in an instance of SQL Server.
setupadmin	Members of the **setupadmin** fixed server role can add and remove linked servers by using Transact-SQL statements. (**sysadmin** membership is needed when using Management Studio.)
bulkadmin	Members of the **bulkadmin** fixed server role can run the BULK INSERT statement.
diskadmin	The **diskadmin** fixed server role is used for managing disk files.
dbcreator	Members of the **dbcreator** fixed server role can create, alter, drop, and restore any database.
public	Every SQL Server login belongs to the **public** server role. When a server principal has not been granted or denied specific permissions on a securable object, the user inherits the permissions granted to public on that object. Only assign public permissions on any object when you want the object to be available to all users. You cannot change membership in public.

User Defined Server Roles

- Creating and configuring a user-defined server role is very straightforward.
- You create the role, grant permissions to the role, and then add logins—or you can add the logins and then grant the permissions.
- CREATE SERVER ROLE devops;
- GRANT ALTER ANY DATABASE TO devops;
- ALTER SERVER ROLE devops ADD MEMBER [win10b\winuser01];