MS SQL Server Security Model Review - Questions on Configurations - and Vulnerabilities

Many thanks to Microsoft, VoiceOfTheDBA, Charlie Osborne (writing for ZeroDay) and Kevin Beaver (writing for Tech Target) for making this summary possible.

MS SQL Server Security Model can be viewed in two sections:

- **Principals**
  - SQL Server entities (login/users, etc) which are able to request SQL Server resources.
- **Securables**
  - SQL Server entities (or objects) (SQL Server, DB, Tables, logins/users, etc) which have permissions particular to their nature.
    - For instance,
      - a SQL Server can be configured, rebooted, etc.
      - a DB can be altered, created, dropped, etc.

Please also note:

- *A permission can be granted, revoked, or denied.*

- *Principals* are arranged in hierarchy and their placement in the security model exposes different permissions.

  At the Windows-level, we have the following principals:
  - Windows Domain Login
  - Windows Local Login

  At the SQL Server-level, we have the following principals:
  - SQL Server Login
  - Server Role

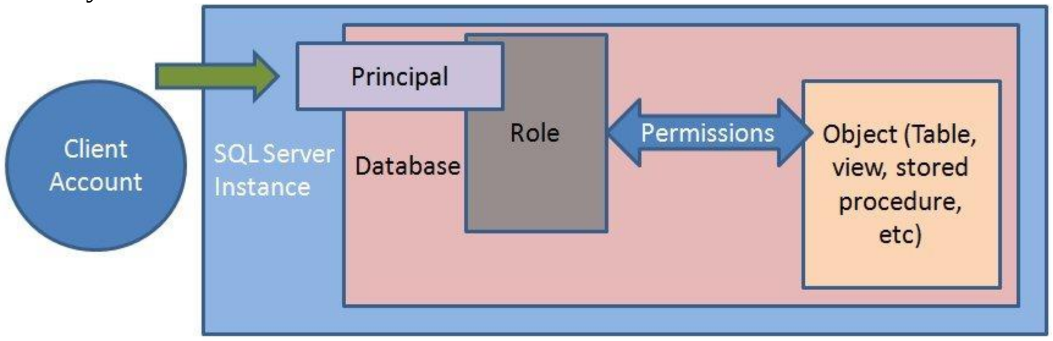
  At the Database-level, we have the following principals:
  - Database User
  - Database Role
  - Application Role

- Every principal has a security identifier (SID).

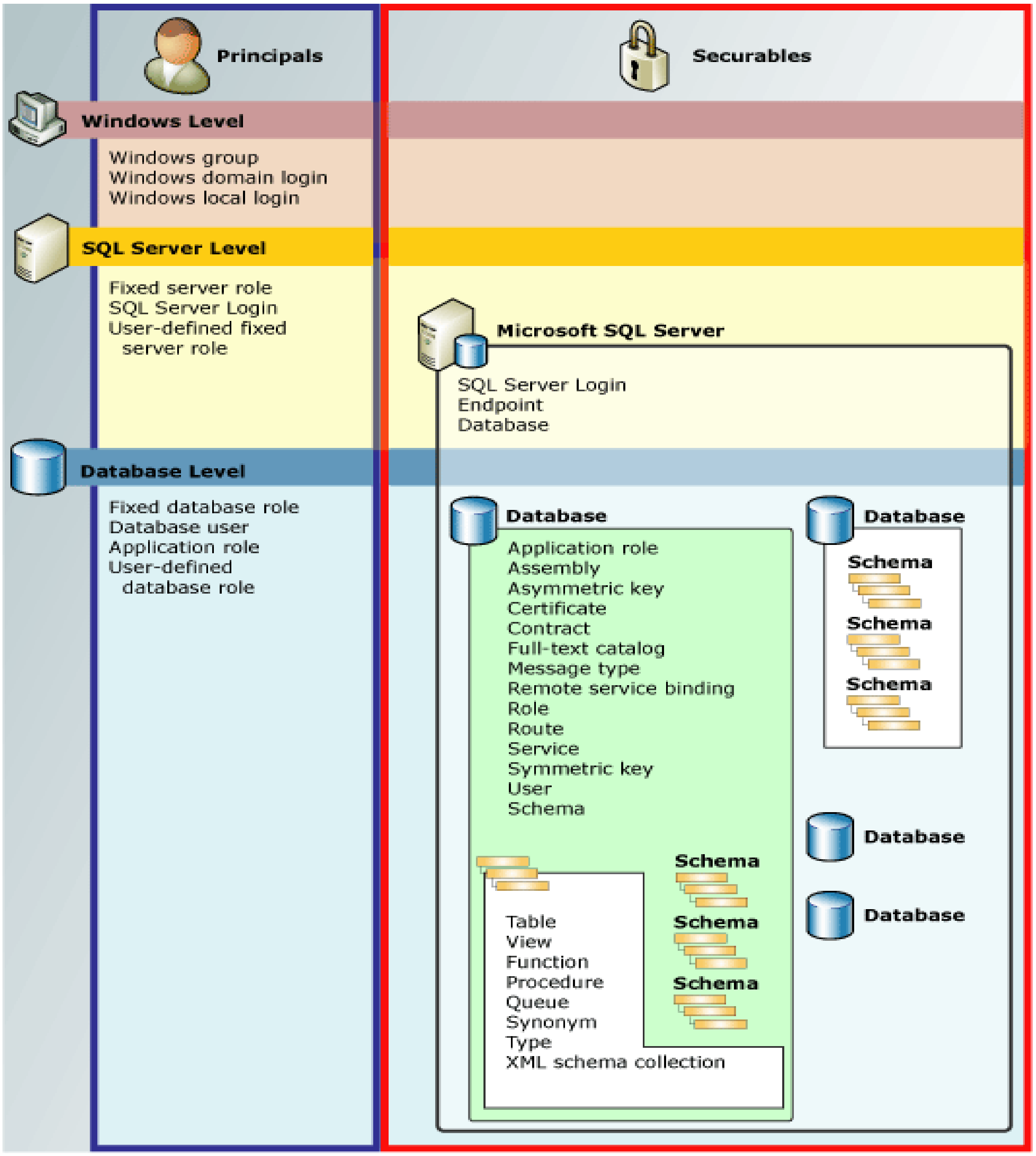
Resource:
- https://msdn.microsoft.com/en-us/library/ms181127.aspx
- https://msdn.microsoft.com/en-us/library/ms191465.aspx

"VoiceOfTheDBA" provides an easy to read chart on the Client's login process:

- A Client's name logins using the principal created by the DBA into a SQL Server Instance.
- The principle exist at the Windows and/or at the DB Level.
- The principle has been granted/revoked/denied permissions on securables (DB Objects).



Resource: https://voiceofthedba.wordpress.com/2011/08/29/the-basic-security-model-in-sql-server-skill-3/



At the SQL Server level, Microsoft has created default roles – Fixed Server-Level Roles

- **So what's a ROLE?**
    - **Do you have a Role Model?** Somebody you look up to…you should really – if you don't.
    - **If you were going to be part of a movie, what would your role be?**
    - **If you were going to work on a SQL Server, what would your role be?** The biggest most powerful guy in the server? Because if so, your role would be "Sysadmin."

Please see the different default/fixed roles we find at the SQL Server level:

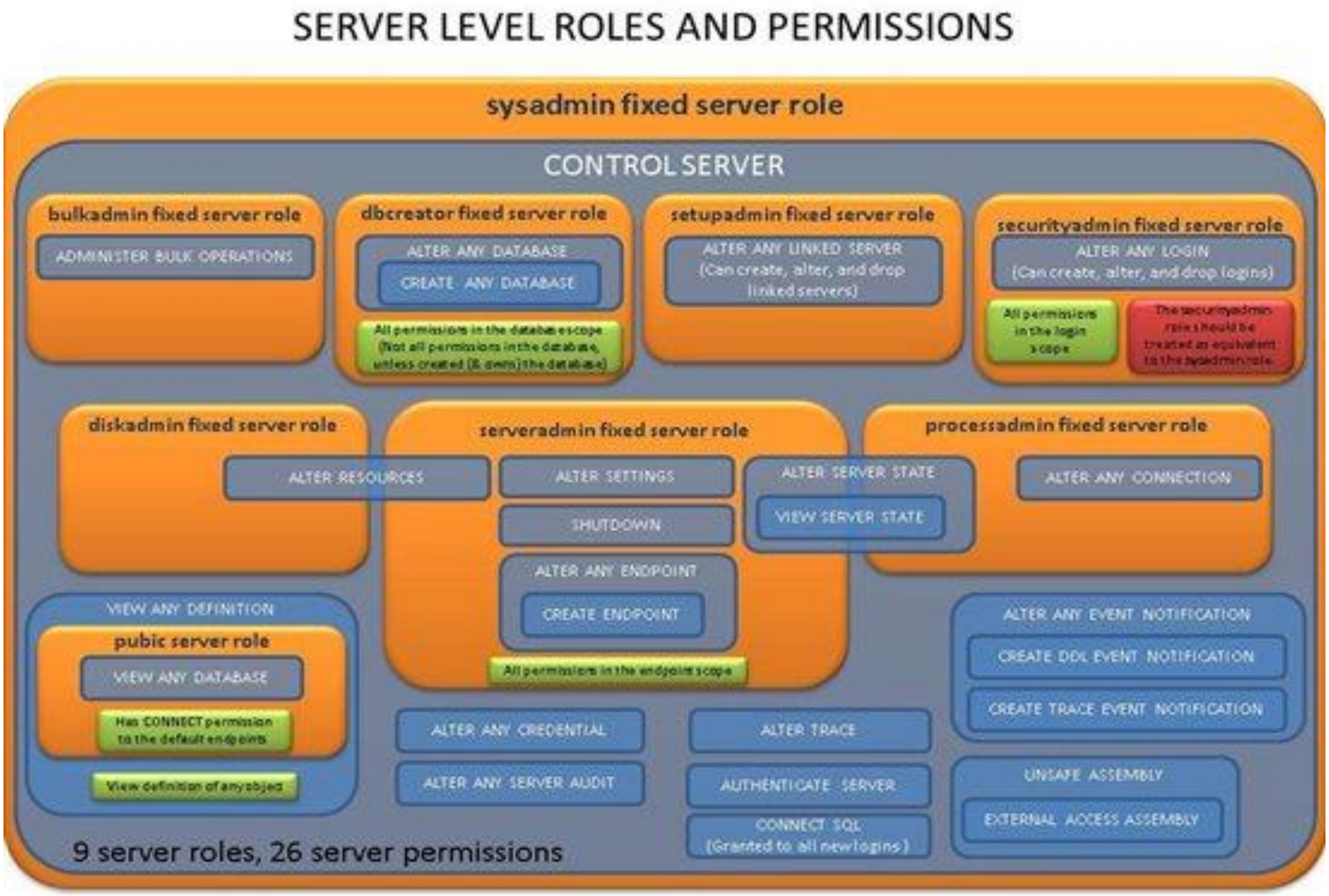
- These are straight from Microsoft; They can't be changed.

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. **** Security Note **** The ability to grant access to the Database Engine and to configure user permissions allows the security admin to assign most server permissions. The **securityadmin** role should be treated as equivalent to the **sysadmin** role.
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.)

Fixed server-level role	Description
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. Note: public is implemented differently than other roles. However, permissions can be granted, denied, or revoked from public.

Now, guess what? These fixed Server Roles are so wide in their scope that they can broke down into “Permissions” or “things that they can do.”

For a chart of the permissions assigned to the server roles, see Database Engine Fixed Server and Fixed Database Roles.


SERVER LEVEL ROLES AND PERMISSIONS
sysadmin fixed server role
CONTROL SERVER
bulkadmin fixed server role
ADMINISTER BULK OPERATIONS
dbcreator fixed server role
ALTER ANY DATABASE
CREATE ANY DATABASE
All permissions in the database scope (Not all permissions in the database, unless created (& owns) the database)
setupadmin fixed server role
ALTER ANY LINKED SERVER (Can create, alter, and drop linked servers)
securityadmin fixed server role
ALTER ANY LOGIN (Can create, alter, and drop logins)
All permissions in the login scope
The securityadmin role should be treated as equivalent to the sysadmin role.
diskadmin fixed server role
ALTER RESOURCES
serveradmin fixed server role
ALTER SETTINGS
SHUTDOWN
ALTER ANY ENDPOINT
CREATE ENDPOINT
All permissions in the endpoint scope
ALTER SERVER STATE
VIEW SERVER STATE
processadmin fixed server role
ALTER ANY CONNECTION
VIEW ANY DEFINITION
pubic server role
VIEW ANY DATABASE
Has CONNECT permission to the default endpoints
View definition of any object
ALTER ANY CREDENTIAL
ALTER ANY SERVER AUDIT
ALTER TRACE
AUTHENTICATE SERVER
CONNECT SQL (Granted to all new logins)
ALTER ANY EVENT NOTIFICATION
CREATE DDL EVENT NOTIFICATION
CREATE TRACE EVENT NOTIFICATION
UNSAFE ASSEMBLY
EXTERNAL ACCESS ASSEMBLY
9 server roles, 26 server permissions

Resource:

- https://msdn.microsoft.com/en-us/library/ms188659.aspx
- http://social.technet.microsoft.com/wiki/contents/articles/2024.database-engine-fixed-server-and-fixed-database-roles.aspx

◆ **Important**
The **CONTROL SERVER** permission is similar but not identical to the **sysadmin** fixed server role. Permissions do not imply role memberships and role memberships do not grant permissions. (E.g. **CONTROL SERVER** does not imply membership in the **sysadmin** fixed server role.) However, it is sometimes possible to impersonate between roles and equivalent permissions. Most **DBCC** commands and many system procedures require membership in the **sysadmin** fixed server role. For a list of 171 system stored procedures that require **sysadmin** membership, see the following blog post by Andreas Wolter CONTROL SERVER vs. sysadmin/sa: permissions, system procedures, DBCC, automatic schema creation and privilege escalation - caveats.

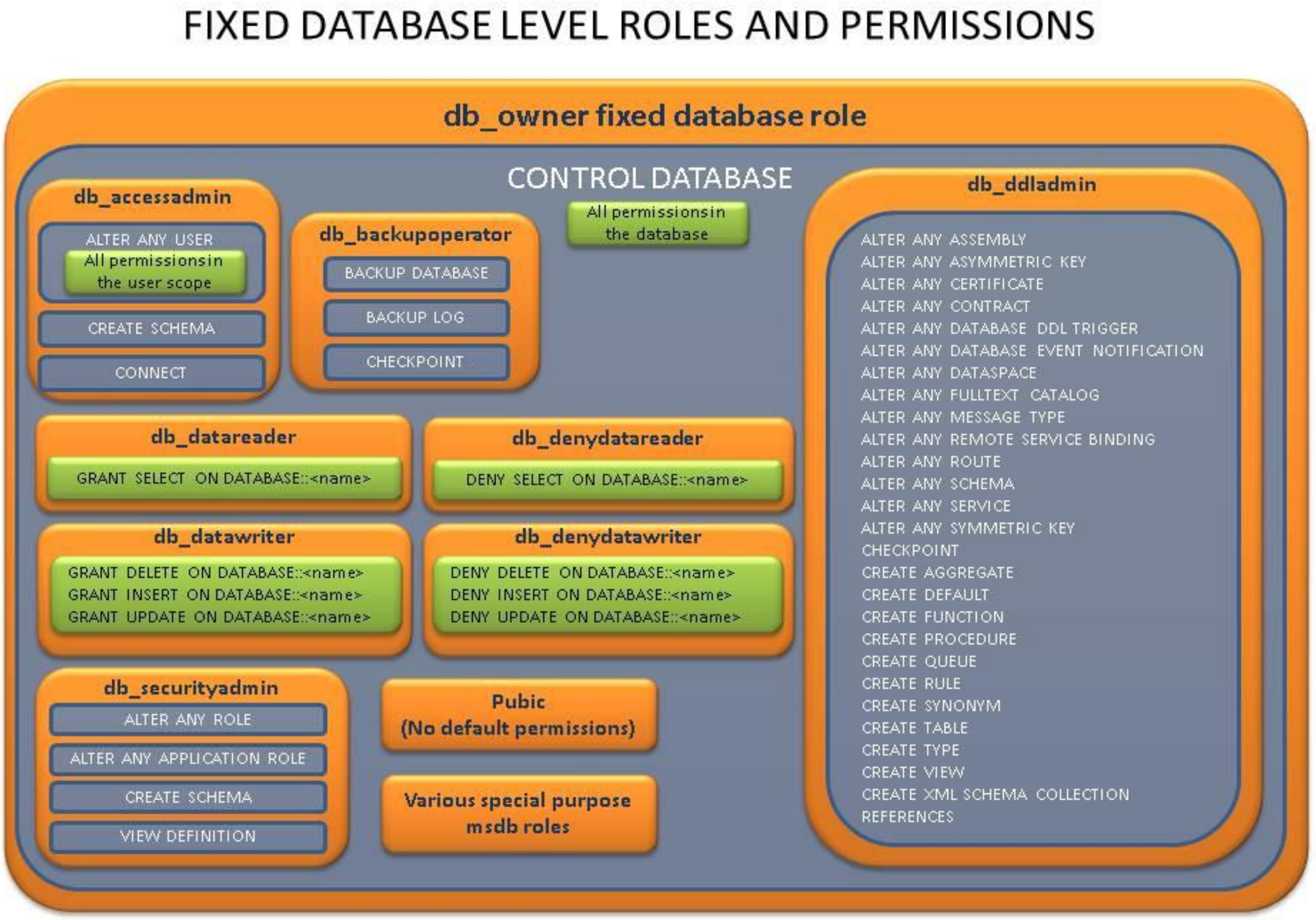
So a "ROLE" can be understood as a "PERSON" and this "PERSON" has different "permissions" to do this or that thing. ***If you were going to be placed at the DB level in a SQL Server, what role would you like to have? What permission would you like to have?***

Microsoft provides the following table showing the fixed-database roles and their capabilities. These roles exist in all databases. The permissions assigned to the fixed-database roles cannot be changed.

Fixed-Database role name	**Description**
db_owner	Members of the **db_owner** fixed database role can perform all configuration and maintenance activities on the database, and can also drop the database in SQL Server. (In SQL Database and SQL Data Warehouse, some maintenance activities require server-level permissions and cannot be performed by **db_owners**.)
db_securityadmin	Members of the **db_securityadmin** fixed database role can modify role membership and manage permissions. Adding principals to this role could enable unintended privilege escalation.
db_accessadmin	Members of the **db_accessadmin** fixed database role can add or remove access to the database for Windows logins, Windows groups, and SQL Server logins.
db_backupoperator	Members of the **db_backupoperator** fixed database role can back up the database.
db_ddladmin	Members of the **db_ddladmin** fixed database role can run any Data Definition Language (DDL) command in a database.

Fixed-Database role name	**Description**
db_datawriter	Members of the **db_datawriter** fixed database role can add, delete, or change data in all user tables.
db_datareader	Members of the **db_datareader** fixed database role can read all data from all user tables.
db_denydatawriter	Members of the **db_denydatawriter** fixed database role cannot add, modify, or delete any data in the user tables within a database.
db_denydatareader	Members of the **db_denydatareader** fixed database role cannot read any data in the user tables within a database.

The permissions assigned to the fixed-database roles cannot be changed. The following figure shows the permissions assigned to the fixed-database roles:



Resource: https://msdn.microsoft.com/en-us/library/ms189121.aspx?tduid=(1af15db34b7ae4bcc89e1e37ca0cd235)(256380)(2459594)(TnL5HPStwNw-2Hbx290tprrVnPVN6H10XA)()

How do we configure our SQL Server's security model?

- Would you rather have DBAs and others log into your SQL Server via Windows Authentication or SQL Authentication?
- What would be your password policy?
- Are all DBAs going to have the same roles/permissions? Do you have trainnes who might require less access?
- Are there any logins/passwords that you need to keep others from changing?
- Are there any securables (DB,tables, etc) on your SQL Server which are simply static? They don't change…and if they did, the "change" will likely take customers down. If so, do you want to protect it by denying everyone (except for the admin person) the ability to change these securables?
- Do you have multi-subnets? Do you need principals able to read not just on the local server but beyond?
- Do you have too many principles and need to organize them into groups and roles?
- Would an application role work out better for your needs?

Microsoft offers a great deal of information on the security model that could help us make the right decision:

https://msdn.microsoft.com/en-us/library/ms161956.aspx CHOOSE AN AUTHENTICATION MODE METADATA VISIBILITY CONFIGURATION SURFACE AREA CONFIGURATION SQL INJECTION TRUSTWORTHY DATABASE PROPERTY PERMISSIONS PERMISSIONS HIERARCHY PERMISSIONS OR SECURABLES PAGE PASSWORD POLICY STRONG PASSWORDS ROW-LEVEL SECURITY SQL SERVER ENCRYPTION DYNAMIC DATA MASKING SQL SERVER CERTIFICATES AND ASYMMETRIC KEYS	**Chart of all Database Engine permissions in pdf format** file:///C:/Users/pab/Downloads/Permissions_Poster_2016_and_SQLDB%20(1).pdf

Creating a well-designed Security Model is common-sense. There is obviously a balance between security and practicality, and one should choose what's best for the company. However, do not fall into tragic error of making it too complicated. I believe that a well-designed security model is Simple, effective, and easy to maintain. Microsoft has given us the tools to make this happen! Go do it!! ☺

Finally, Charlie Osborne (in 2013) writing for Zero Day and Kevin Beaver (in 2016) writing for TechTarget give us a top 10 list of SQL vulnerabilities.

Top 10 from Charlie Osborne	Top 10 from Kevin Beaver
1. Deployment Failures The most common cause of database vulnerabilities is a lack of due care at the moment they are deployed. Although any given database is tested for functionality and to make sure it is doing what the databases is designed to do, very few checks are made to check the database is not doing things it should not be doing. **2. Broken databases** The SQL Slammer worm of 2003 was able to infect more than 90 percent of vulnerable computers within 10 minutes of deployment, taking down thousands of databases in minutes. This worm took advantage of a bug that was discovered in	1. Users or groups are granted permission to execute xp_cmdshell, which allows the execution of OS commands they likely shouldn't have access to. 2. Remote databases are granted the ability to login and run remote stored procedures on the local database.

Microsoft's SQL Server database software the previous year, but few system administrators installed a fix, leaving computers vulnerable.

By exploiting a buffer-overflow vulnerability, the worm's success demonstrates how critical installing security patches and fixes are. However, whether lacking time or resources, not enough businesses keep their systems regularly patched, leaving databases vulnerable.

3. Data leaks

Databases may be considered a "back end" part of the office and secure from Internet-based threats (and so data doesn't have to be encrypted), but this is not the case. Databases also contain a networking interface, and so hackers are able to capture this type of traffic to exploit it. To avoid such a pitfall, administrators should use SSL- or TLS-encrypted communication platforms.

4. Stolen database backups

External attackers who infiltrate systems to steal data are one threat, but what about those inside the corporation? The report suggests that insiders are also likely to steal archives — including database backups — whether for money, profit or revenge. This is a common problem for the modern enterprise, and businesses should consider encrypting archives to mitigate the insider-risk.

5. The abuse of database features

The research team says that over the past three years, every database exploit they've seen has been based on the misuse of a standard database feature. For example, a hacker can gain access through legitimate credentials before forcing the service to run arbitrary code. Although complex, in many cases, this access was gained through simple flaws that allow such systems to be taken advantage of or bypassed completely. Future abuse can be limited by removing unnecessary tools — not by destroying the possibility of zero-day exploits, but by at least

3. Registry extended stored procedures are enabled, which permit unauthorized reads/writes of the Windows registry, including everything accessible via the Local System account.

4.Standard users are granted permission to execute CmdExec and ActiveX scripts, which allows full access to the OS (2000 only).

5.Standard users are granted permission to escalate privileges through the SQL Agent. That can facilitate a user running a malicious job via the extended stored procedures to gain full administrator privileges to the database (2000 only).

shrinking the surface area hackers can study to launch an attack.	
6. A lack of segregation The separation of administrator and user powers, as well as the segregation of duties, can make it more difficult for fraud or theft undertaken by internal staff. In addition, limiting the power of user accounts may give a hacker a harder time in taking complete control of a database.	6. PUBLIC group is granted excessive permissions into the database leading to unnecessary database and OS file access such as: • SELECT and EXECUTE to the data transformation services • SELECT to the system tables and metadata • The msdb.dbo.mswebtasks table (2000 only)
7. Hopscotch Rather than taking advantage of buffer overflow and gaining complete access to a database in the first stage, cybercriminals often play a game of Hopscotch: finding a weakness within the infrastructure that can be used as leverage for more serious attacks until they reach the back-end database system. For example, a hacker may worm their way through your accounts department before hitting the credit card processing arena. Unless every department has the same standard of control, creating separate administrator accounts and segregating systems can help mitigate the risk.	7. Guest user is enabled allowing unauthorized (or unaudited) access into the database.
8. SQL injections A popular method for hackers to take, SQL injections remain a critical problem in the protection of enterprise databases. Applications are attacked by injections, and the database administrator is left to clean up the mess caused by unclean variables and malicious code which is inserted into strings, later passed to an instance of SQL server for parsing and execution. The best ways to protect against these threats are to protect web-facing databases with firewalls and to test input variables for SQL injection during development.	8. Error log count is set too low, creating a situation where critical logs can be overwritten, which limits database audit trails.

9. Sub-standard key management Key management systems are meant to keep keys safe, but the research team often found encryption keys stored on company disk drives. Database administrators sometimes falsely believe these keys have to be left on the disk because of database failures, but this isn't true — and placing such keys in an unprotected state can leave systems vulnerable to attack. **10. Database inconsistencies** Finally, the researchers found that the common thread which brings all of these vulnerabilities together is a lack of consistency, which is an administrative rather than database technology problem. System administrators and database developers need to develop a consistent practice in looking after their databases, staying aware of threats and making sure that vulnerabilities are taken care of. This isn't an easy task, but documentation and automation to track and make changes can ensure that the information contained in enterprise networks is kept secure. http://www.zdnet.com/article/the-top-ten-most-common-database-security-vulnerabilities/	9. Logging of success and failure logins is not enabled, limiting database audit trails. 10. Missing Windows and SQL Server patches allow full remote command prompt access to the database server. http://searchsqlserver.techtarget.com/tip/Ten-common-SQL-Server-security-vulnerabilities-you-may-be-overlooking

Really, again, I reiterate: The Security Model should follow common-sense practices and