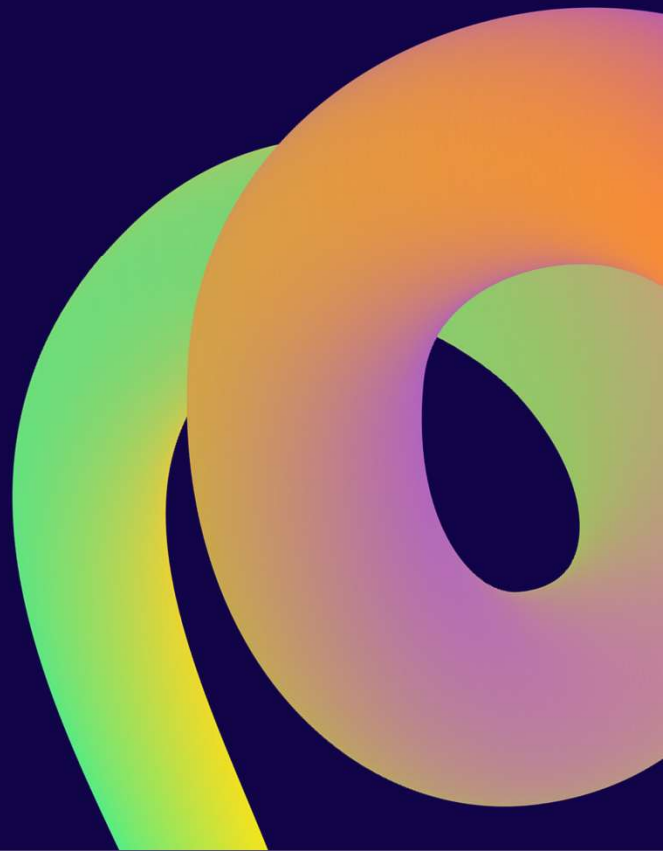
SQL Server Security from top to bottom

Ben Johnston

Principal Architect

I-Tech Solutions, Inc.



Ben Johnston

Principal Architect
I-Tech Solutions, Inc.

I've worked with SQL Server and database systems for over 25 years, building warehouses, transactional systems, reporting environments, and system integrations.

Consulting with medium and large enterprises, I specialize in database design, performance tuning, security, troubleshooting, and Azure upgrades.

www.red-gate.com/simple-talk/author/ben-johnston/

www.linkedin.com/in/bensql

github.com/BenSQL/SQLSecurity

Only assign security required for the user or task.

Principle of Least Privilege

PASS
Data Community Summit

Authentication vs Authorization

Authentication

Is this the actual user, have they proven it, are they allowed to access the system?

Authorization

We know this user. What are they allowed to access and what tasks can they perform on the system? This includes databases, objects, server level tasks (i.e., backups), even metadata about the system.

Logins / Users

LOGINS

- Authentication
- sys.server_principals
- SQL logins are specific to servers
  - Even with the same name, it is a different login
- It is easier to manage AD and Entra groups
- Many enterprises disable sa

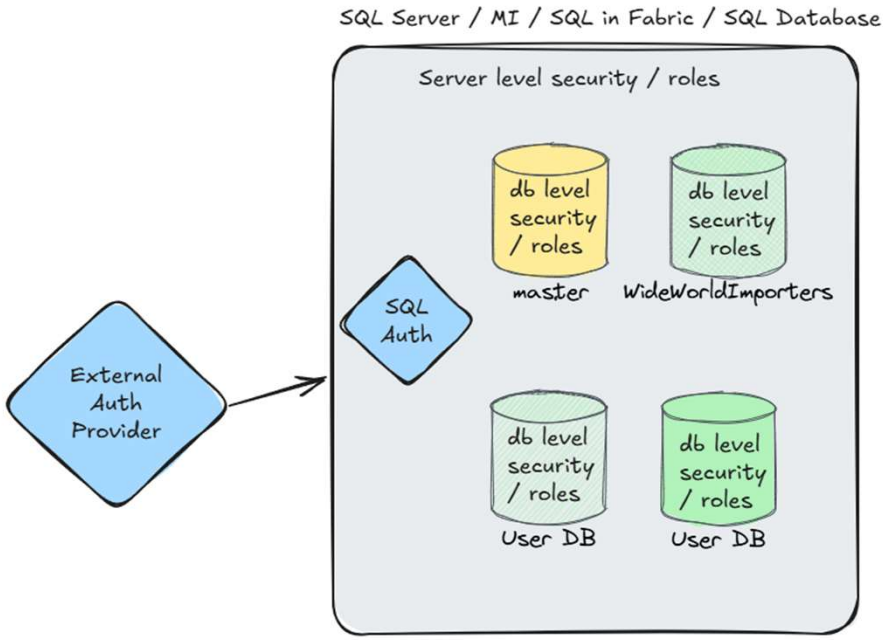
USERS

- Authorization
- Contained users are also be used for authentication
- sys.database_principals
- Specific to the current database
  - Cross database queries are possible depending on configuration
- User name can differ from login

Authentication in SQL

- Windows Authentication (integrated authentication)
- SQL Server Authentication
- Entra Authentication (Azure Active Directory / AAD)
  - Entra Integrated
  - Entra MFA
  - Entra Default
  - Managed Identities
  - Entra service principal name and application secret
- Certificate mapped
- Asymmetric key mapped

Authentication to Authorization



Authorization in SQL

GRANT

- Additive

REVOKE

- Removes existing access

DENY

- Overrides any GRANT statement
- Column level GRANT not impacted by DENY

Users / Database Principals

Users based on logins in master

- User based on login using SQL Authentication
- User based on login using Windows Active Directory
- User based on login using Windows AD group
- User based on Microsoft Entra login (AAD)

Contained users – no login in master

- SQL authentication
- Based on user in Windows Active Directory
- Based on user in Windows AD group
- Based on Microsoft Entra login
- Based on Microsoft Entra group

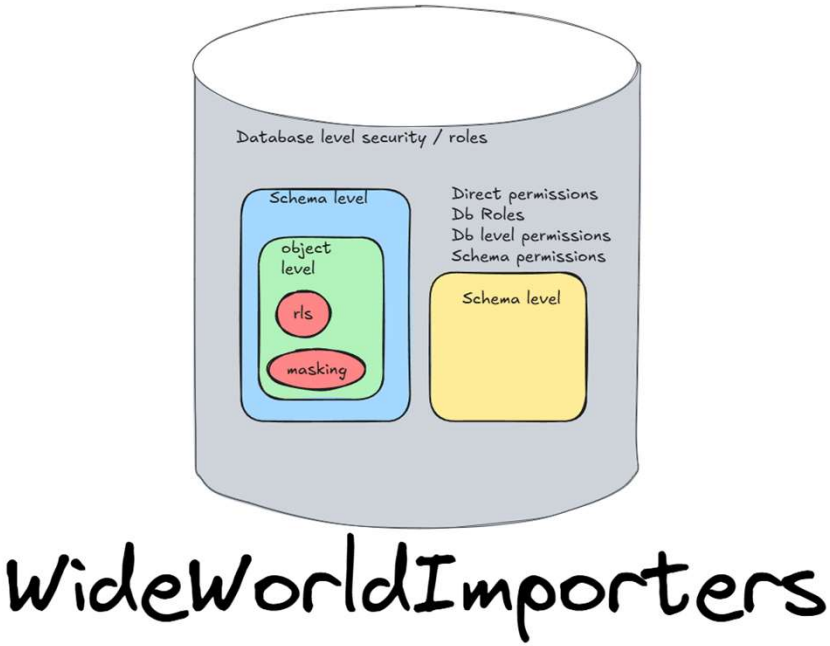
User without login

User from certificate

User from asymmetric key

Application Roles

Authorization Levels



Demo – Security in SSMS

Server Roles - Classic

Server level roles

- bulkadmin
- dbcreator
- diskadmin
- processadmin
- public
- securityadmin
- serveradmin
- setupadmin
- sysadmin

Server Roles – Least Privilege

Available in SQL 2022 +

- ##MS_DatabaseConnector##
- ##MS_DatabaseManager##
- ##MS_DefinitionReader##
- ##MS_LoginManager##
- ##MS_PerformanceDefinitionReader##
- ##MS_SecurityDefinitionReader##
- ##MS_ServerPerformanceStateReader##
- ##MS_ServerSecurityStateReader##
- ##MS_ServerStateManager##
- ##MS_ServerStateReader##

Database Roles

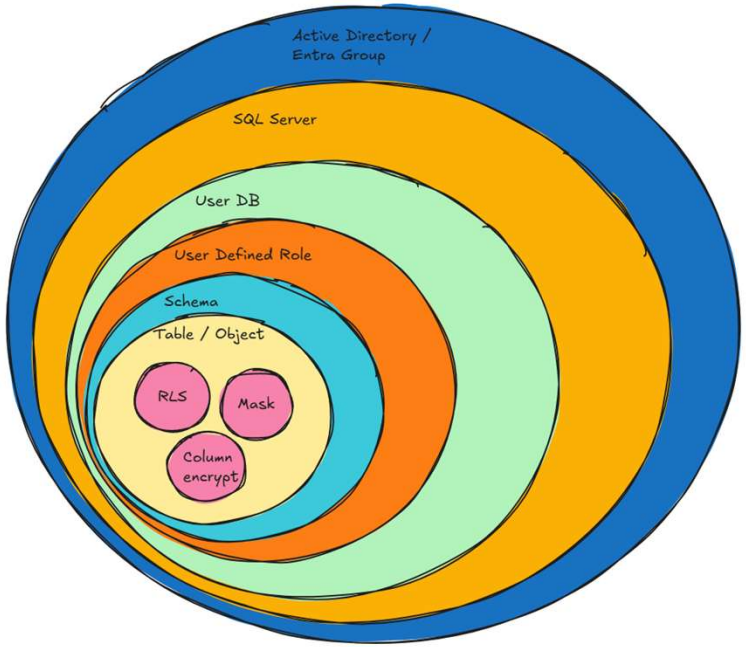
System database roles

- db_owner
- db_accessadmin
- db_securityadmin
- db_ddladmin
- db_backupoperator
- db_datareader
- db_datawriter
- db_denydatareader
- db_denydatawriter

Object Level Permissions

- ALTER
- CONTROL
- DELETE
- EXECUTE
- INSERT
- RECEIVE
- REFERENCES
- SELECT
- TAKE OWNERSHIP
- UPDATE
- VIEW CHANGE TRACKING
- VIEW DEFINITION
- UNMASK

Security Nesting



Validating Security

sys.fn_my_permissions (securable , 'securable_class')

- Table valued function

HAS_PERMS_BY_NAME (securable , securable_class , permission, [, sub-securable] [, sub-securable_class])

- Inline function

sys.server_permissions

sys.server_role_members

sys.database_permissions

sys.database_role_members

Demo – Security Validation

Additional Security Mechanisms

Stored procedures

Views

Functions

Row Level Security (RLS)

System Managed Dynamic Data Masking (masking)

Column Level Encryption

Transparent Data Encryption (TDE)

Object / Schema Owners

Objects should be owned by dbo unless there is a specific, articulable requirement

Stored procedures / views / functions can be used as a security mechanism

- Users accessing the stored procedure / object don’t need direct access to the objects referenced

Objects owned by the same principal (schema) don’t need explicit access granted inside of a stored procedure / view / function

Best Practices

Remember Principal of Least Privilege

- Balance this with developer needs – everything is a tradeoff

Use Active Directory /Entra groups when possible

- Managed identity in Azure for services
- Consider disabling sa / SQL Logins

Don’t alter security or owners on system / is_ms_shipped objects (column in sys.objects)

Automate deployments / Automate ETL

- Deployments from source control are auditable
- Limits the number of accounts with elevated permissions
- Reduces errors

Use SQL Audits to track sensitive systems

Simple is better than clever

Best Practices

Use schemas to segregate data

- ETL / load data / RLS / temporal
- Vendor data
- Sensitive data

Consistency

- Create security patterns and use them

Default database / server roles

- Ok for small teams
- Make your own roles for bigger projects

Security is always a trade-off

Security design is integral to database and application design

Sample Application Framework

Active Directory / Entra Groups for login / user assignment

Groups added as users to destination database

User defined roles for each common security use case

- Principal of least privilege
- Assign security at a schema level when possible (or database level)
  - Use schemas to segregate data by function and security
  - Expose data to external services via service specific schemas
    - Views and stored procedure wrap base tables
- Direct access to data always incurs a risk
  - Restrict access - even in warehouse environments
  - Be very cautious granting access in production – even SELECT access

Demo – Sample Setup

Mitigating Risk

System procedures / functions can often be wrapped by a user function

- Allows higher risk objects be assigned to users / roles without adding user to server roles and database roles such as db_owner
- Database must be set to trustworthy or the user object must be signed

Specific permissions can be granted to a user defined server role for power users rather than server roles and database roles

- GRANT KILL DATABASE CONNECTION TO <role / user>
- GRANT SHOWPLAN TO <login / group>

Demo – Mitigations

Your feedback is important to us

Evaluate this session at:

passdatacommunitysummit.com/evaluations

Thank you

Ben Johnston

www.red-gate.com/simple-talk/author/ben-johnston/

www.linkedin.com/in/bensql

github.com/BenSQL/SQLSecurity