SQL Saturday
San Diego, CA
2025

Microsoft | AMD

databricks

O'REILLY

COZYROC

SQL skills
immerse yourself in sql server

University of San Diego | ONLINE

Tessell

Community Support

Name	Type	Frequency	Join
San Diego SQL Server User Group	Hybrid	Every 3rd Thursday monthly @ 6.30PM PDT	
LA Data Platform User Group	Virtual	Every 3rd Wednesday monthly @ 7PM PDT	
San Francisco Microsoft Fabric Meetup	Virtual	Every 2rd Wednesday monthly @ 7PM PDT	
Sacramento SQL Server User Group	In-Person	Every 1st Wednesday monthly @ 6PM	

#SQLSatSD

Find more Azure Data Tech Groups on Meetup

Thomas AJ Hull

SQL Server Security, Encryption, & Compliance

PASS SQLSATURDAY

About Me

- 20+ Years of SQL Server experience, starting with v6.5
- Worked as consultant in financial and biotech industries in Boston and San Diego
- MCSE: Data Platform & MCSE: Data Management and Analytics
- DBA Team lead at MUFG Union Bank
- Presenter and co-organizer of SQL User Groups and SQL Saturdays
- Proud Dad of 2 very cute kids

What we’ll cover

- Regulatory Standards, Best Practices
- Overview of Security issues in SQL Server
    - Levels of security
    - Principals, Securables, Permissions
- Encryption
    - Certificates, Keys
    - Backup, TDE, Column, AlwaysEncrypted
- New features
- Common attacks
- Dealing with Audits

First step – Understand what are we protecting

- PII – “Personally Identifiable Information”
  - Information that can potentially be used to uniquely identify a person.
  - Includes: SSN, street address, Driver’s License number, phone number, IP addresses, Email address, vehicle registration, and age
- NPI – “Non Public Information”
  - Covers all information appearing on applications for obtaining financial services
  - Generally a broader category than PII
  - Includes PII info plus: PINs, passwords, account numbers, salaries, account balances
- PHI – “Protected Health Information”
  - Individually identifiable health information held or transmitted in any medium
- Proprietary business information that could damage the business
- Uptime (DOS attacks, etc.)

Regulatory / Industry Standards

What does it mean to be "in compliance"?

- Depends on:
- Type of data
  - PII, NPI, PHI
- Industry
  - Medical, Financial
- Jurisdiction
  - Privacy laws

"Hi, I'm Hugo — I'll be your compliance officer."

Financial Services - Regulatory Standards

- SOX – Sarbanes-Oxley Act of 2002
  - Enacted after major corporate frauds (Enron, Worldcom, Tyco)
  - Section 404 mandates internal controls for protecting financial information.
  - Confidentiality and integrity of data must be protected by controls, security, and documentation.
- GLBA - Gramm-Leach-Billey Data Protection Act of 1999
  - Section 501(b) requires firms to protect the confidentiality and integrity of NPI and PII through administrative, technical, and physical safeguards.
  - Note this act specifically mentions encryption of PII in transit. (more on this later)

Financial Services - Industry Standards

- Industry standards are not enacted by law, but violations can have huge costs in dollars and reputation.
- PCI - Payment Card Industry (PCI) Security Standards
  - Applies to financial institutions that accept credit card transactions
  - Regulates the security, confidentiality, and integrity of NPI
  - Require encryption of cardholder data, and logging & monitoring of electronic communications.
- NASD - National Association of Securities Dealers
  - Impacts electronic communications for transmitting NPI and PII

Medical – Regulatory Standards

- HIPAA - Health Insurance Portability and Accountability Act
  - Title 2.1 – Privacy Rule
    - Calls for protection of “PHI” – Protected health Information
  - Title 2.3 – Security Rule
  - Violations can bring fines and also criminal charges.

Other Jurisdictions

- GDPR - General Data Protection Regulation
  - Implemented in Europe on May 25, 2018
  - Requires "Data protection by design and by default“
  - Right of Erasure (formerly “right to be forgotten”)
  - Very stringent auditing requirements
  - Max penalty – Greater of €20M or 4% of annual worldwide revenue.
- CCPA - California Consumer Privacy Act of 2018
  - Slated to go into effect on January 1, 2020
- This stuff is only getting more prevalent
- Pays to stay ahead of it

Security Basics – Foundational Principle

Principle of Least Privilege

- Grant a user account “only those privileges which are essential to perform its intended function”
- Applies to each of the security layers to be discussed
- You will always get pushback on this from developers and customers, who like having sysadmin everywhere, and don’t see all the dangers.
    - “I would never do anything wrong!”
    - (Until they accidentally truncate a production table or compromise production customer info)
- Also, lazy or novice DBA’s ignore this rule because it seems “easier” to grant Admin everywhere and move on

Security Basics – 2nd Foundational Principle

Principle of Simplicity (or – Rule of “Least Effort”)**!**

- My second foundation principle (after PoLP) is Simplicity.
- Make use of:
  - Database roles
  - Schemas
  - Leverage integration with AD (password policies, AD groups)
- If you have the chance to architect a database from the ground up, use that opportunity to design schemas corresponding to required security roles
- You probably didn’t choose this career to spend all day creating logins/users and granting individual permissions
- The more complex it is, the more likely something slips through the cracks

Security Basics

- **Authentication vs Authorization**
- Authentication
  - Confirming or verifying one’s identity
- Authorization
  - Verifying what one has access to
- In order to be authorized, one must first be authenticated
- But it’s possible to be authenticated, but not authorized
  - (i.e. I know who you are, but you’re not allowed to do that)

Security Basics – Layers of security

- Platform and Network Security
  - Physical Security
  - Operating System Security
  - SQL Server Files Security
- Principals and Database Object Security
  - Server and database users, roles, and processes
  - Server and database objects security
  - Encryption and Certificates
- Application Security
  - Best coding practices
  - Client network configuration

Platform Security

- Physical security
  - Typically SQL Servers should be in a locked secured data center
    - I've seen PROD servers under somebody's desk or in a broom closet
  - Very limited access, nobody except server techs should be able to touch it, i.e. no developers, executives, etc.
  - Badge access with logging
- Operating System Security
  - What users have elevated (or any) rights on the server?
    - Do they need it? (go back to PoLP)
  - Do your SQL service accounts have admin permissions on the server?
    - Do they need it? (hint, NO)
    - **MAKE SURE** to edit service accounts through SQL Config Mgr, **NOT** "Services" Mgr
    - Issues with GPO (Group Policy Objects) settings

Network Security

- Direct internet access for SQL Server?
  - NO!
  - Put web server on DMZ
  - Use Network firewalls
- Server level firewalls
  - Inbound – limit ports to:
    - 1433 (default port); can and probably should should change this
    - Custom port for named instance (Avoid dynamic)
    - 1434 - UDP for SQL Browser (perhaps turn this off as well)
    - 4022 - Service Broker
    - More Details: https://docs.microsoft.com/en-us/sql/sql-server/install/configure-the-windows-firewall-to-allow-sql-server-access?view=sql-server-2017

Network Security

- If you have the opportunity to install/design from the ground up, consider using non-default instance name and port number
- Hackers all know port 1433 for default SQL Server instance
- Also, consider turning off the SQL Browser
  - Advertises the instances and ports living on the server
  - Without Browser, clients must specify port number for non-default (1433) instances
- Or consider making certain instances “hidden” in network config
- Enable only required protocols
  - We typically shut off Named Pipes & Shared Memory and use only TCP/IP
- “Security by obscurity” – not going to stop capable/determined attackers, but let’s not lay out a red carpet for them
- Demo

Reducing Surface Area

DEMO

```
]Q

?SYNTAX ERROR
]
]
]
]C
```

Principals and Database Object Security

A Principal is Any Entity that can request a SQL Server resource

- Instance-level Principals
  - SQL Server authentication Login
  - Windows authentication login (Integrated Security)
    - Active Directory User or Group
    - Local machine User or Group (avoid this)
  - Server Role
- Database-level Principals
  - Database User
    - Many types
  - Database Role
  - Application Role

Principals – Database Users

Types of Database User

- Users based on logins in master (most common)
  - User based on a login based on a Windows Domain or Local account.
  - User based on a login based on a Windows Domain or Local group.
  - User based on a login using SQL Server authentication.;

- Users that authenticate at the database level
  - User based on a Windows user (or group) that has no login in SQL.
  - Contained database user with password.

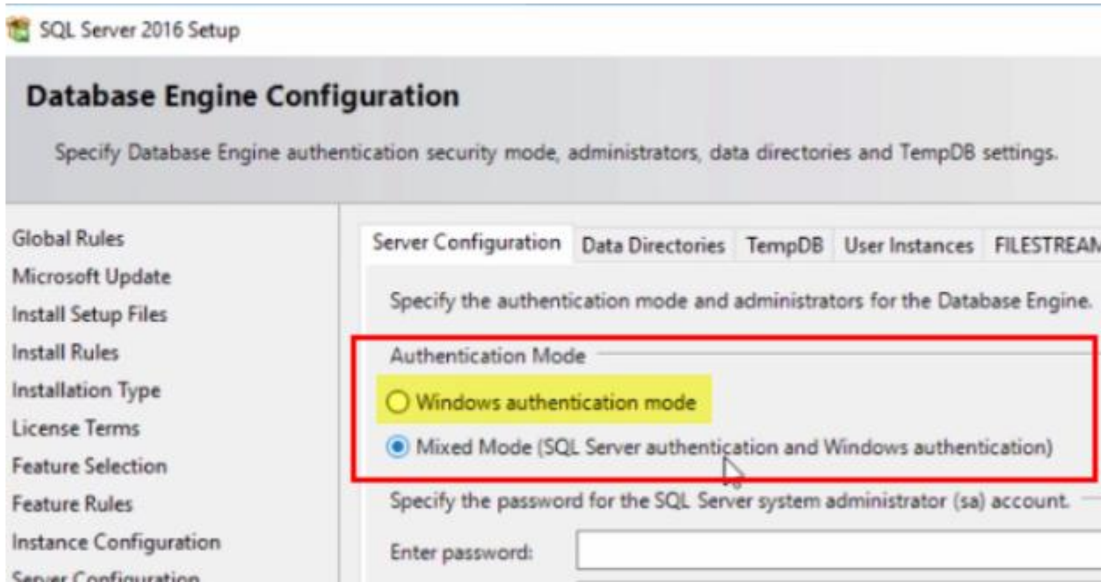
- Users that cannot authenticate – Cannot log in to instance
  - User without a login. Can be granted permissions.
  - User based on a certificate. Can an be granted permissions and sign modules.
  - User based on asymmetric key. Can be granted permissions and sign modules.

Principals – Logins vs Users

- This topic causes endless confusion, people always mix them up
- **Logins**
  - Authenticates and connects at the **Server level**
  - Authentication information is stored only in the master database
  - Can be associated with multiple users (one per database), and can have server-level roles/permissions
  - Two types
    - Windows Authenticated (“Integrated Security”)
    - SQL Server Authenticated
- **Users**
  - At the **Database level**
  - Most often, typically associated with a Login
    - But could be “WITH NO LOGIN”, or could be authenticated at database level for Contained databases
  - Joins on SID
    - Doesn’t need to have the same name as the login
    - Even if a user’s name matches a login, it won’t link properly if SID is different (SQL Authentication)

Principals – Logins

- When installing SQL Server, you are given the choice of "**Authentication Mode**"
- Default is "**Windows authentication Mode**" – This is **recommended** if at all possible.
- "Mixed Mode" allows SQL Authentication
  - Sometimes needed due to:
    - legacy apps
    - Application-specific login
    - Web-based app w/no Windows authentication
  - Requires non-blank sa password
    - (old versions didn't!)



SQL Server 2016 Setup

Database Engine Configuration

Specify Database Engine authentication security mode, administrators, data directories and TempDB settings.

Global Rules
Microsoft Update
Install Setup Files
Install Rules
Installation Type
License Terms
Feature Selection
Feature Rules
Instance Configuration
Server Configuration

Server Configuration | Data Directories | TempDB | User Instances | FILESTREA

Specify the authentication mode and administrators for the Database Engine.

Authentication Mode

Windows authentication mode [✓ Mixed Mode (SQL Server authentication and Windows authentication)]

Specify the password for the SQL Server system administrator (sa) account.

Enter password:

Principals – Logins

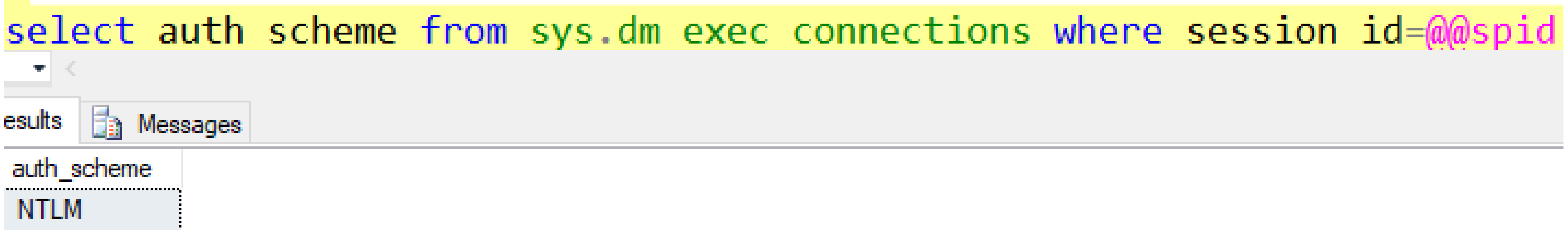
Why is “Windows authentication Mode” preferred over “Mixed”?

- Much more secure than SQL Server Authentication.
- SQL Server validates the account name and password using the Windows principal token in the operating system
- Provides password policy enforcement (complexity, expiration, etc.)
- Provides support for account lockout
- And the great thing is, this is all handled for you based on the company’s already existing AD policies – more effective and more **simple**

Principals – Logins

“Windows authentication Mode” Uses **Kerberos** security protocol

- Frequent issue where it drops down to NTLM if the SQL SPN (Service Principal Name) is not registered properly on the domain
- NOTE- if you’ve seen the Linked Server “double-hop” issue for AD Authentication, it’s probably due to this issue
- To Check:



```
select auth_scheme from sys.dm_exec_connections where session_id=@@spid
```

```
auth_scheme
NTLM
```

- To Fix: **setspn** -A MSSQLSvc/<SQL Server FQDN>:<Port> <Domain\Account>
    - (May need to call your Windows/AD admin for help if you don’t have domain-level permissions)
- https://technet.microsoft.com/en-us/library/bb735885.aspx

Securables

Securables are the resources to which principals can request access or privileges.

The permissions roll down from higher level to lower level.

- Server Level
  - Availability Group, Login, Server Role, Database
- Database Level
  - Database Role, Schema, User
- Schema Level
  - Objects
- Object Level
  - Tables, Procedures, Functions, Views
- Column Level

Permissions

Most permission statements have the format:

```
AUTHORIZATION PERMISSION SECURABLE::NAME TO PRINCIPAL
```

- AUTHORIZATION must be GRANT, REVOKE or DENY.
- PERMISSION is listed in the charts below.
- ON SECURABLE::NAME is the server, server object, database, or database object and its name.
    - (omitted for server-wide & database-wide permissions.)
- PRINCIPAL is the login, user, or role which receives or loses the permission.
- Grant permissions to roles whenever possible.
- Sample grant statement:

```
GRANT UPDATE ON OBJECT::Production.Parts TO PartsTeam
```

Permissions - GRANT, REVOKE, DENY

GRANT – Grants permissions on a securable to a principal

DENY – Denies a permission to a principal

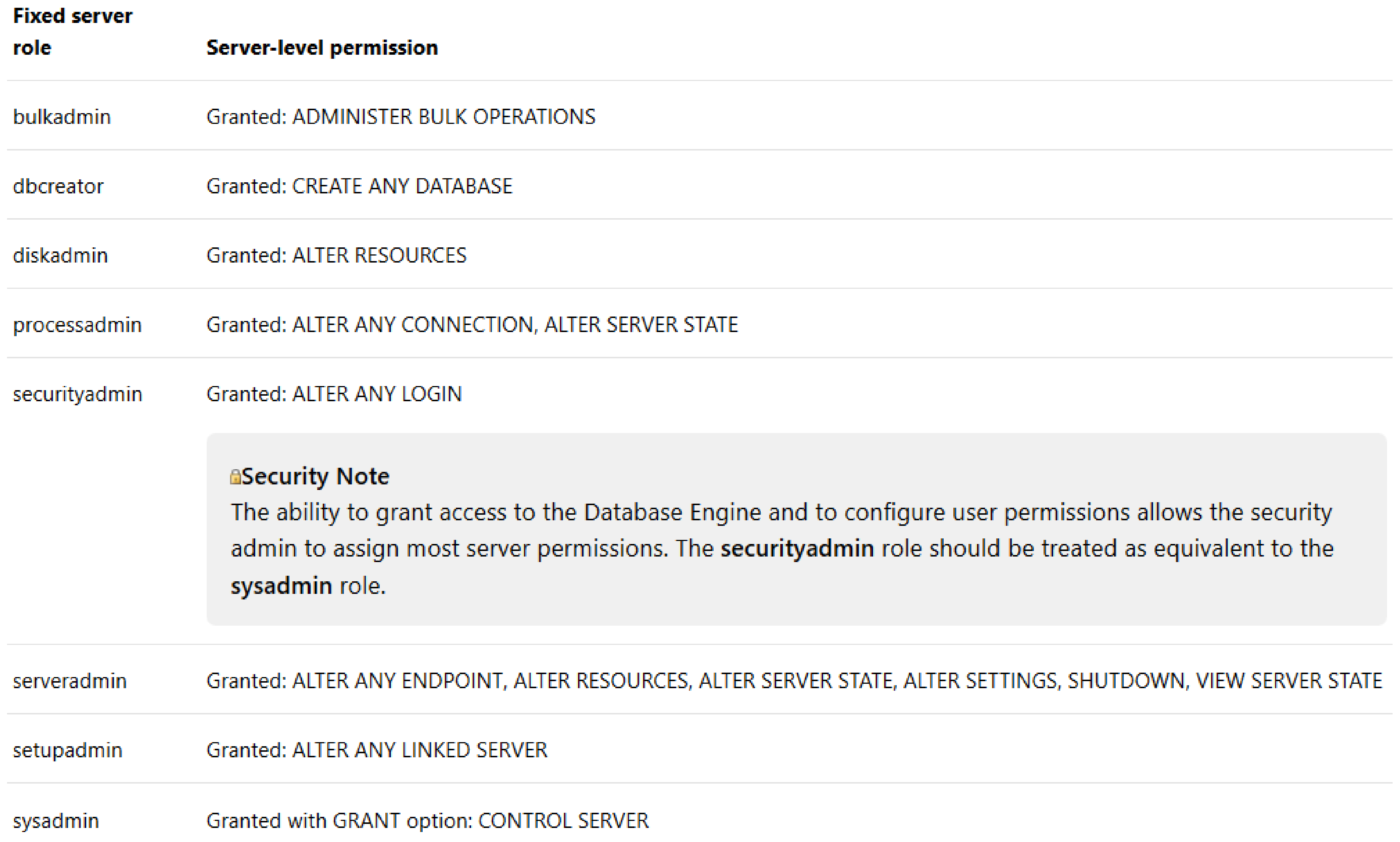
REVOKE – Removes a previously granted or denied permission

- Confusion between DENY vs REVOKE
  - REVOKE merely removes a previously entered permission record
  - So GRANT & DENY add entries to the permissions table; REVOKE deletes GRANT/DENY entries from the permissions table
  - So if you REVOKE a DENY, what happens?
    - Depends on other permissions that apply to that principal
- If there are conflicting GRANT/DENY permissions (based on individual vs groups, roles, etc) who wins?
  - **DENY** ALWAYS WINS
  - This is a good thing

Permissions - GRANT, REVOKE, DENY

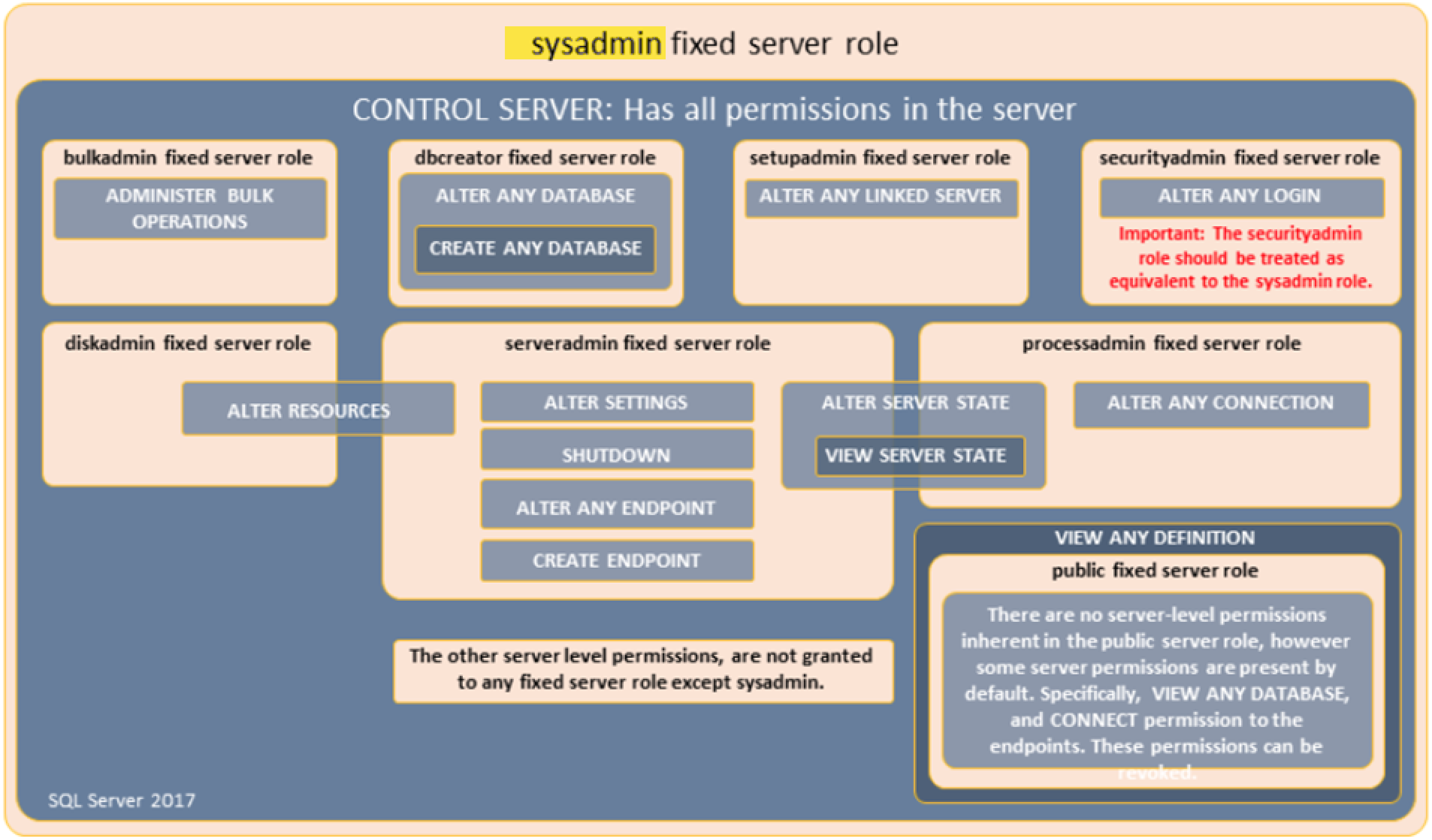
DEMO

Built-in “Fixed Server roles”



Fixed server role	Server-level permission
bulkadmin	Granted: ADMINISTER BULK OPERATIONS
dbcreator	Granted: CREATE ANY DATABASE
diskadmin	Granted: ALTER RESOURCES
processadmin	Granted: ALTER ANY CONNECTION, ALTER SERVER STATE
securityadmin	Granted: ALTER ANY LOGIN **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.
serveradmin	Granted: ALTER ANY ENDPOINT, ALTER RESOURCES, ALTER SERVER STATE, ALTER SETTINGS, SHUTDOWN, VIEW SERVER STATE
setupadmin	Granted: ALTER ANY LINKED SERVER
sysadmin	Granted with GRANT option: CONTROL SERVER

Server Level Roles – one role to rule them all

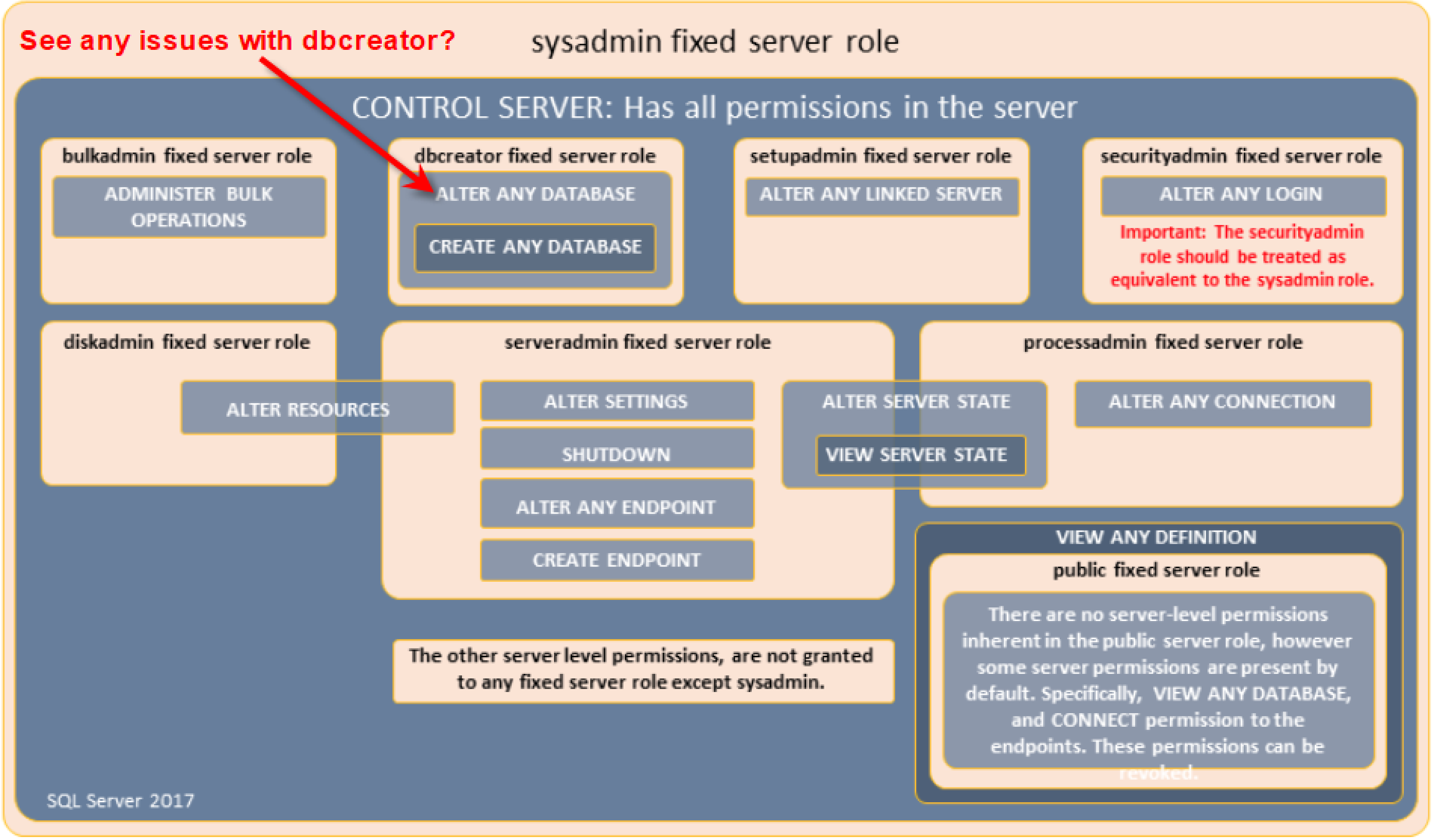

sysadmin fixed server role
CONTROL SERVER: Has all permissions in the server
bulkadmin fixed server role
ADMINISTER BULK OPERATIONS
dbcreator fixed server role
ALTER ANY DATABASE
CREATE ANY DATABASE
setupadmin fixed server role
ALTER ANY LINKED SERVER
securityadmin fixed server role
ALTER ANY LOGIN
Important: 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
ALTER SERVER STATE
VIEW SERVER STATE
processadmin fixed server role
ALTER ANY CONNECTION
VIEW ANY DEFINITION
public fixed server role
SQL Server 2017

The other server level permissions, are not granted to any fixed server role except sysadmin.

There are no server-level permissions inherent in the public server role, however some server permissions are present by default. Specifically, VIEW ANY DATABASE, and CONNECT permission to the endpoints. These permissions can be revoked.

https://docs.microsoft.com/en-us/sql/relational-databases/security/authentication-access/server-level-roles?view=sql-server-2017

Server Level Roles


See any issues with dbcreator?
sysadmin fixed server role
CONTROL SERVER: Has all permissions in the server
bulkadmin fixed server role
ADMINISTER BULK OPERATIONS
dbcreator fixed server role
ALTER ANY DATABASE
CREATE ANY DATABASE
setupadmin fixed server role
ALTER ANY LINKED SERVER
securityadmin fixed server role
ALTER ANY LOGIN
diskadmin fixed server role
ALTER RESOURCES
serveradmin fixed server role
ALTER SETTINGS
SHUTDOWN
ALTER ANY ENDPOINT
CREATE ENDPOINT
ALTER SERVER STATE
VIEW SERVER STATE
processadmin fixed server role
ALTER ANY CONNECTION
VIEW ANY DEFINITION
public fixed server role
SQL Server 2017

Important: The securityadmin role should be treated as equivalent to the sysadmin role.

The other server level permissions, are not granted to any fixed server role except sysadmin.

There are no server-level permissions inherent in the public server role, however some server permissions are present by default. Specifically, VIEW ANY DATABASE, and CONNECT permission to the endpoints. These permissions can be revoked.

https://docs.microsoft.com/en-us/sql/relational-databases/security/authentication-access/server-level-roles?view=sql-server-2017

Built in “Fixed database roles”

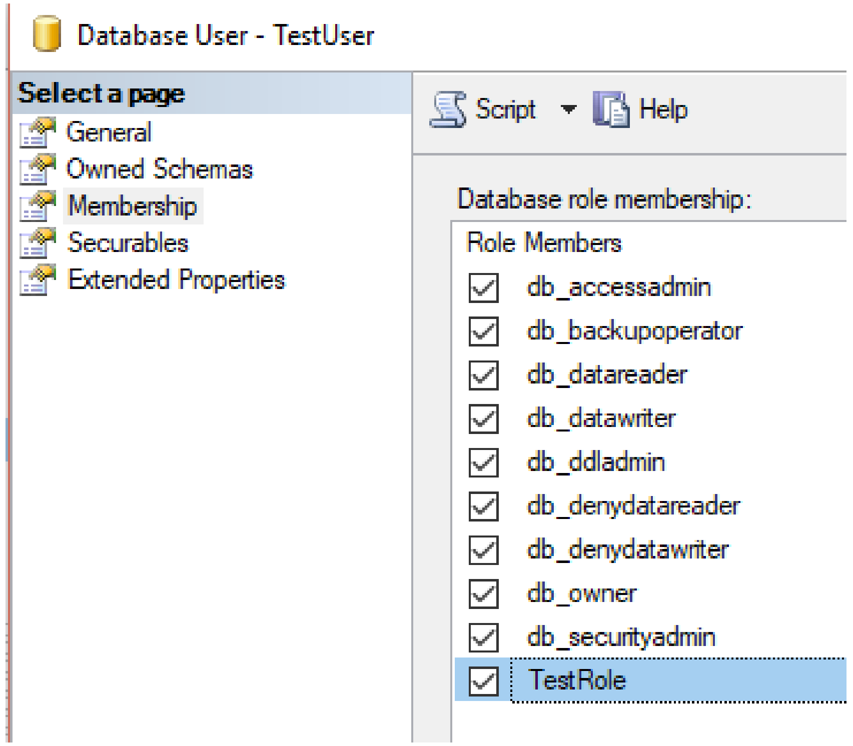
- **db_backupoperator** - Granted: BACKUP DATABASE, BACKUP LOG, CHECKPOINT
- **db_datareader** - Granted: SELECT
- **db_datawriter** - Granted: DELETE, INSERT, UPDATE
- **db_ddladmin** - Granted: ALTER ANY ASSEMBLY, ALTER ANY ASYMMETRIC KEY, ALTER ANY CERTIFICATE, ALTER ANY CONTRACT, ALTER ANY DATABASE DDL TRIGGER, ALTER ANY DATABASE EVENT, NOTIFICATION, ALTER ANY DATASPACE, ALTER ANY FULLTEXT CATALOG, ALTER ANY MESSAGE TYPE, ALTER ANY REMOTE SERVICE BINDING, ALTER ANY ROUTE, ALTER ANY SCHEMA, ALTER ANY SERVICE, ALTER ANY SYMMETRIC KEY, CHECKPOINT, CREATE AGGREGATE, CREATE DEFAULT, CREATE FUNCTION, CREATE PROCEDURE, CREATE QUEUE, CREATE RULE, CREATE SYNONYM, CREATE TABLE, CREATE TYPE, CREATE VIEW, CREATE XML SCHEMA COLLECTION, REFERENCES
- **db_denydatareader** - **Denied**: SELECT
- **db_denydatawriter** - **Denied**: DELETE, INSERT, UPDATE
- **db_owner** - Granted **with GRANT option**: CONTROL
- **db_securityadmin** - Granted: ALTER ANY APPLICATION ROLE, ALTER ANY ROLE, CREATE SCHEMA, VIEW DEFINITION
- **dbm_monitor** - Granted: VIEW most recent status in Database Mirroring Monitor

Built in “Fixed database roles”

Most people have seen these

What’s wrong with this picture? I’ve seen this many times..

Might as well just check them all just in case, right?



Database role membership:

Role Members
✓ db_accessadmin
✓ db_backupoperator
✓ db_datareader
✓ db_datawriter
✓ db_ddladmin
✓ db_denydatareader
✓ db_denydatawriter
✓ db_owner
✓ db_securityadmin
✓ TestRole

Principals, Securables, and Permissions

- How does it all fit together?
- Simple!

Microsoft SQL Server 2016 and Azure SQL Database
Database Engine Permissions

Permission Syntax

Most permission statements have the format:

AUTHORIZATION PERMISSION ON SECURABLE::NAME TO PRINCIPAL

- AUTHORIZATION must be GRANT, REVOKE or DENY.
- PERMISSION is listed in the charts below.
- ON SECURABLE::NAME is the server, server object, database, or database object and its name. (ON SECURABLE::NAME is omitted for server-wide and database-wide permissions.)
- PRINCIPAL is the login, user, or role which receives or loses the permission. Grant permissions to roles whenever possible.

Sample grant statement: GRANT UPDATE ON OBJECT::Production.Parts TO PartsTeam

Denying a permission at any level, overrides a related grant.

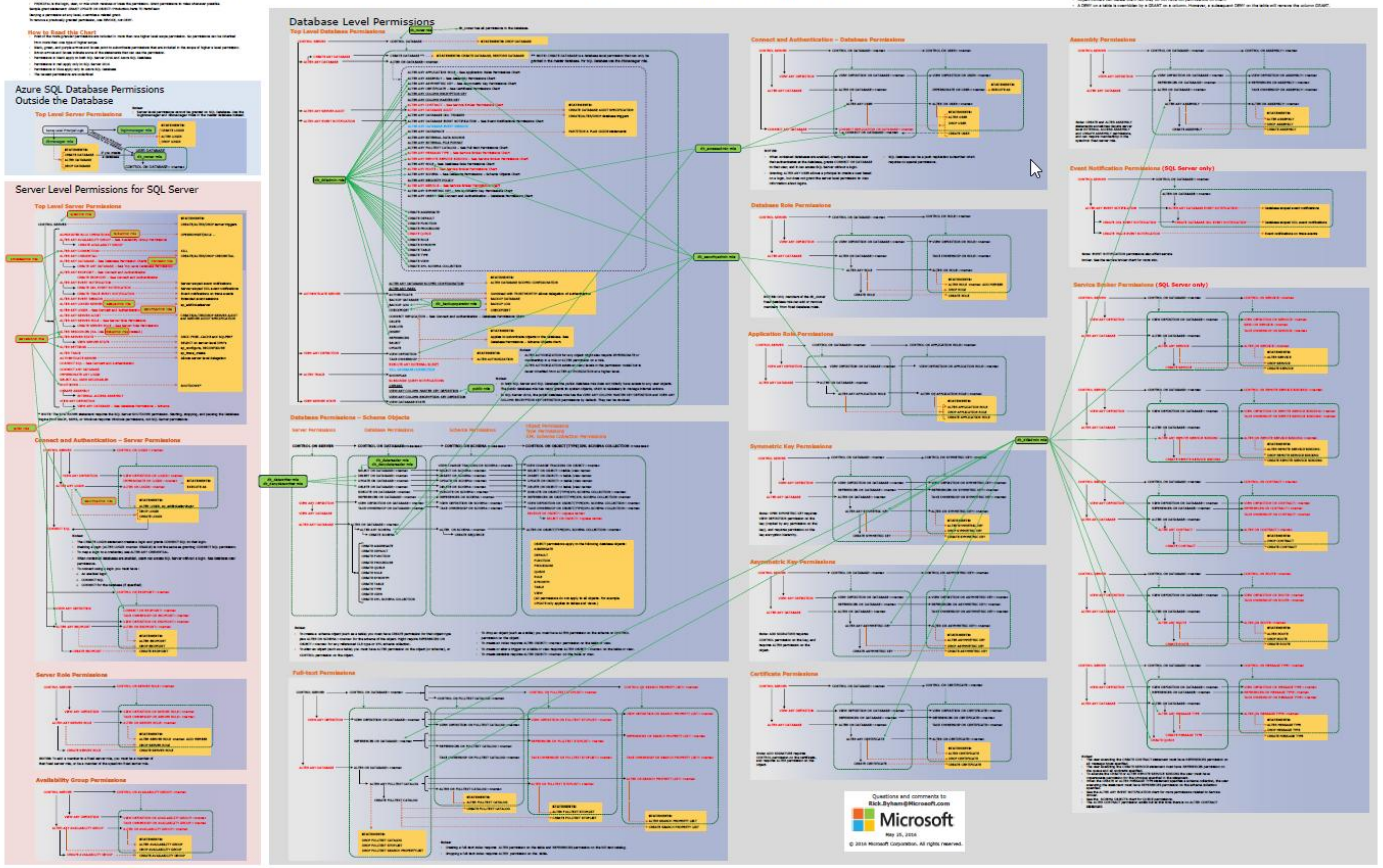
To remove a previously granted permission, use REVOKE, not DENY.

How to Read this Chart

- Most of the more granular permissions are included in more than one higher level scope permission. So permissions can be inherited from more than one type of higher scope.
- Black, green, and purple arrows and boxes point to subordinate permissions that are included in the scope of higher a level permission.
- Brown arrows and boxes indicate some of the statements that can use the permission.
- Permissions in black apply to both SQL Server 2016 and Azure SQL Database
- Permissions in red apply only to SQL Server 2016
- Permissions in blue apply only to Azure SQL Database
- The newest permissions are underlined

NOTES:

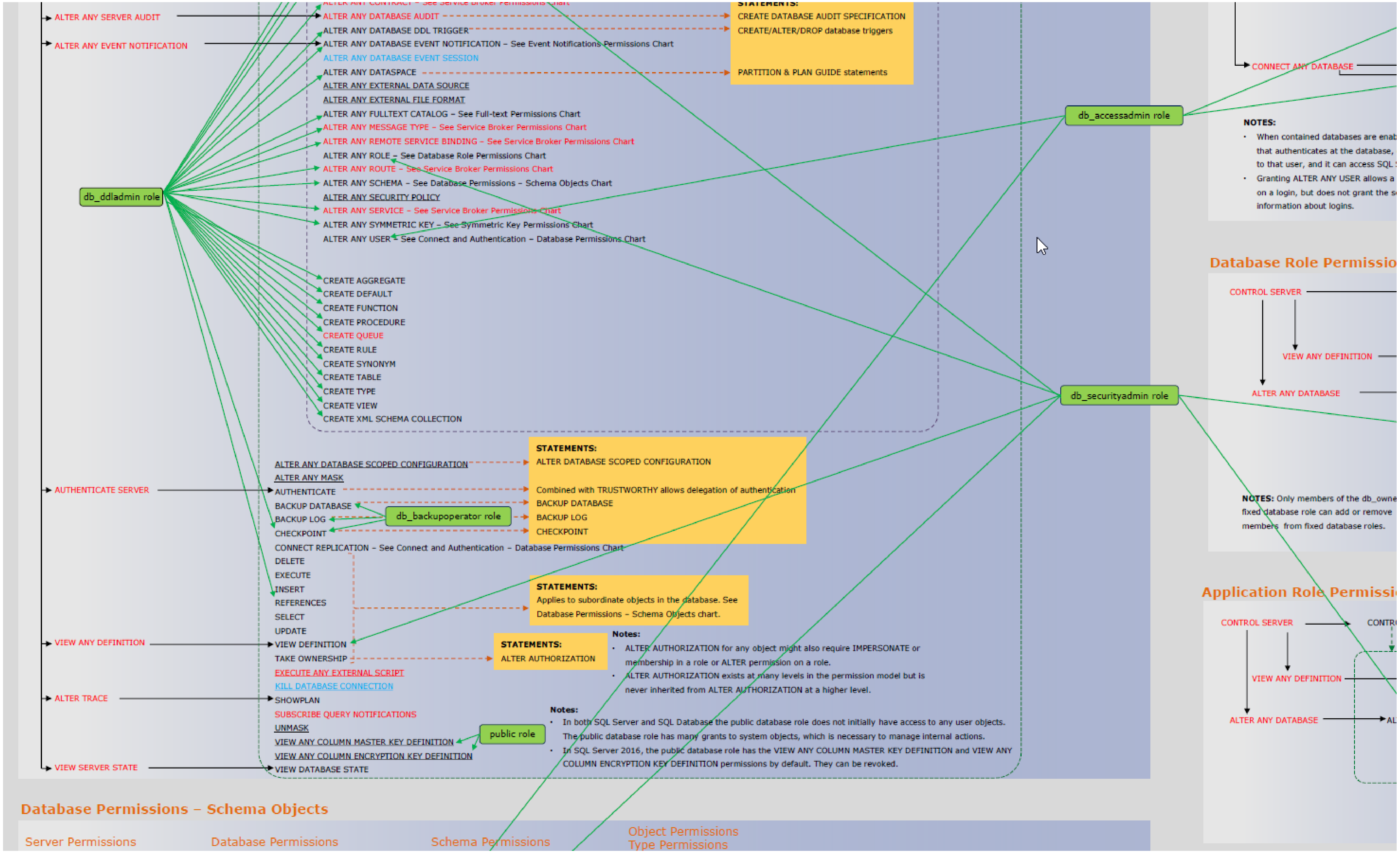
- The CONTROL SERVER permission has all permissions on the instance of SQL Server or SQL Database.
- The CONTROL DATABASE permission has all permissions on the database.
- Permissions do not imply role memberships and role memberships do not grant permissions. (E.g. CONTROL SERVER does not imply membership in the sysadmin fixed server role. Membership in the db_owner role does not grant the CONTROL DATABASE permission.) However, it is sometimes possible to impersonate between roles and equivalent permissions.
- Granting any permission on a securable allows VIEW DEFINITION on that securable. It is an implied permission and it cannot be revoked, but it can be explicitly denied by using the DENY VIEW DEFINITION statement.
- SQL Database permissions refer to version 12.
- Object owners can delete them but they do not have full permissions on them.
- A DENY on a table is overridden by a GRANT on a column. However, a subsequent DENY on the table will remove the column GRANT.


Azure SQL Database Permissions Outside the Database
Top Level Server Permissions
Server Level Permissions for SQL Server
Top Level Server Permissions
Connect and Authentication – Server Permissions
Server Role Permissions
Availability Group Permissions
Database Level Permissions
Top Level Database Permissions
Database Permissions – Schema Objects
Server Permissions
Database Permissions
Schema Permissions
Object Permissions
Full-text Permissions
Connect and Authentication – Database Permissions
Database Role Permissions
Application Role Permissions
Symmetric Key Permissions
Asymmetric Key Permissions
Certificate Permissions
Assembly Permissions
Event Notification Permissions (SQL Server only)
Service Broker Permissions (SQL Server only)
db_owner role
db_securityadmin role
db_accessadmin role
db_ddladmin role
db_datareader role
db_denydatareader role
db_backupoperator role
public role

Questions and comments to
Rick.Byham@Microsoft.com

Microsoft

May 25, 2016


ALTER ANY SERVER AUDIT
ALTER ANY EVENT NOTIFICATION
ALTER ANY CONTRACT – See Service Broker Permissions Chart
ALTER ANY DATABASE AUDIT
ALTER ANY DATABASE DDL TRIGGER
ALTER ANY DATABASE EVENT NOTIFICATION – See Event Notifications Permissions Chart
ALTER ANY DATABASE EVENT SESSION
ALTER ANY DATASPACE
ALTER ANY EXTERNAL DATA SOURCE
ALTER ANY EXTERNAL FILE FORMAT
ALTER ANY FULLTEXT CATALOG – See Full-text Permissions Chart
ALTER ANY MESSAGE TYPE – See Service Broker Permissions Chart
ALTER ANY REMOTE SERVICE BINDING – See Service Broker Permissions Chart
ALTER ANY ROLE – See Database Role Permissions Chart
ALTER ANY ROUTE – See Service Broker Permissions Chart
ALTER ANY SCHEMA – See Database Permissions – Schema Objects Chart
ALTER ANY SECURITY POLICY
ALTER ANY SERVICE – See Service Broker Permissions Chart
ALTER ANY SYMMETRIC KEY – See Symmetric Key Permissions Chart
ALTER ANY USER – See Connect and Authentication – Database Permissions Chart
STATEMENTS:
CREATE DATABASE AUDIT SPECIFICATION
CREATE/ALTER/DROP database triggers
PARTITION & PLAN GUIDE statements
db_ddladmin role
db_accessadmin role
db_securityadmin role
CONNECT ANY DATABASE
CREATE AGGREGATE
CREATE DEFAULT
CREATE FUNCTION
CREATE PROCEDURE
CREATE QUEUE
CREATE RULE
CREATE SYNONYM
CREATE TABLE
CREATE TYPE
CREATE VIEW
CREATE XML SCHEMA COLLECTION
AUTHENTICATE SERVER
ALTER ANY DATABASE SCOPED CONFIGURATION
ALTER ANY MASK
AUTHENTICATE
BACKUP DATABASE
BACKUP LOG
CHECKPOINT
db_backupoperator role
STATEMENTS:
ALTER DATABASE SCOPED CONFIGURATION
Combined with TRUSTWORTHY allows delegation of authentication
BACKUP DATABASE
BACKUP LOG
CHECKPOINT
CONNECT REPLICATION – See Connect and Authentication – Database Permissions Chart
DELETE
EXECUTE
INSERT
REFERENCES
SELECT
UPDATE
STATEMENTS:
Applies to subordinate objects in the database. See Database Permissions – Schema Objects chart.
VIEW ANY DEFINITION
VIEW DEFINITION
TAKE OWNERSHIP
STATEMENTS:
ALTER AUTHORIZATION
EXECUTE ANY EXTERNAL SCRIPT
KILL DATABASE CONNECTION
ALTER TRACE
SHOWPLAN
SUBSCRIBE QUERY NOTIFICATIONS
UNMASK
VIEW ANY COLUMN MASTER KEY DEFINITION
VIEW ANY COLUMN ENCRYPTION KEY DEFINITION
public role
VIEW SERVER STATE
VIEW DATABASE STATE
Database Role Permissio
CONTROL SERVER
VIEW ANY DEFINITION
ALTER ANY DATABASE
Application Role Permissi
CONTROL SERVER
CONTR
VIEW ANY DEFINITION
ALTER ANY DATABASE
AL
Server Permissions
Database Permissions
Schema Permissions
Object Permissions
Type Permissions

NOTES:

- When contained databases are enab that authenticates at the database, to that user, and it can access SQL
- Granting ALTER ANY USER allows a on a login, but does not grant the s information about logins.

Notes:

- ALTER AUTHORIZATION for any object might also require IMPERSONATE or membership in a role or ALTER permission on a role.
- ALTER AUTHORIZATION exists at many levels in the permission model but is never inherited from ALTER AUTHORIZATION at a higher level.

Notes:

- In both SQL Server and SQL Database the public database role does not initially have access to any user objects. The public database role has many grants to system objects, which is necessary to manage internal actions.
- In SQL Server 2016, the public database role has the VIEW ANY COLUMN MASTER KEY DEFINITION and VIEW ANY COLUMN ENCRYPTION KEY DEFINITION permissions by default. They can be revoked.

NOTES: Only members of the db_owne fixed database role can add or remove members from fixed database roles.

Database Permissions – Schema Objects

Encryption

- When dealing with regulators, the subject of encryption comes up a lot.
- If you're not doing it, it's an automatic fail.
  - At first this seemed a little silly, we're all in the same protected network, we're all employees with background checks, etc..
  - However, it makes sense in terms of having multiple layers of security. If someone gets hold of a database backup, or if they try a man-in-the-middle attack, we don't want to just hand over the keys to the kingdom.

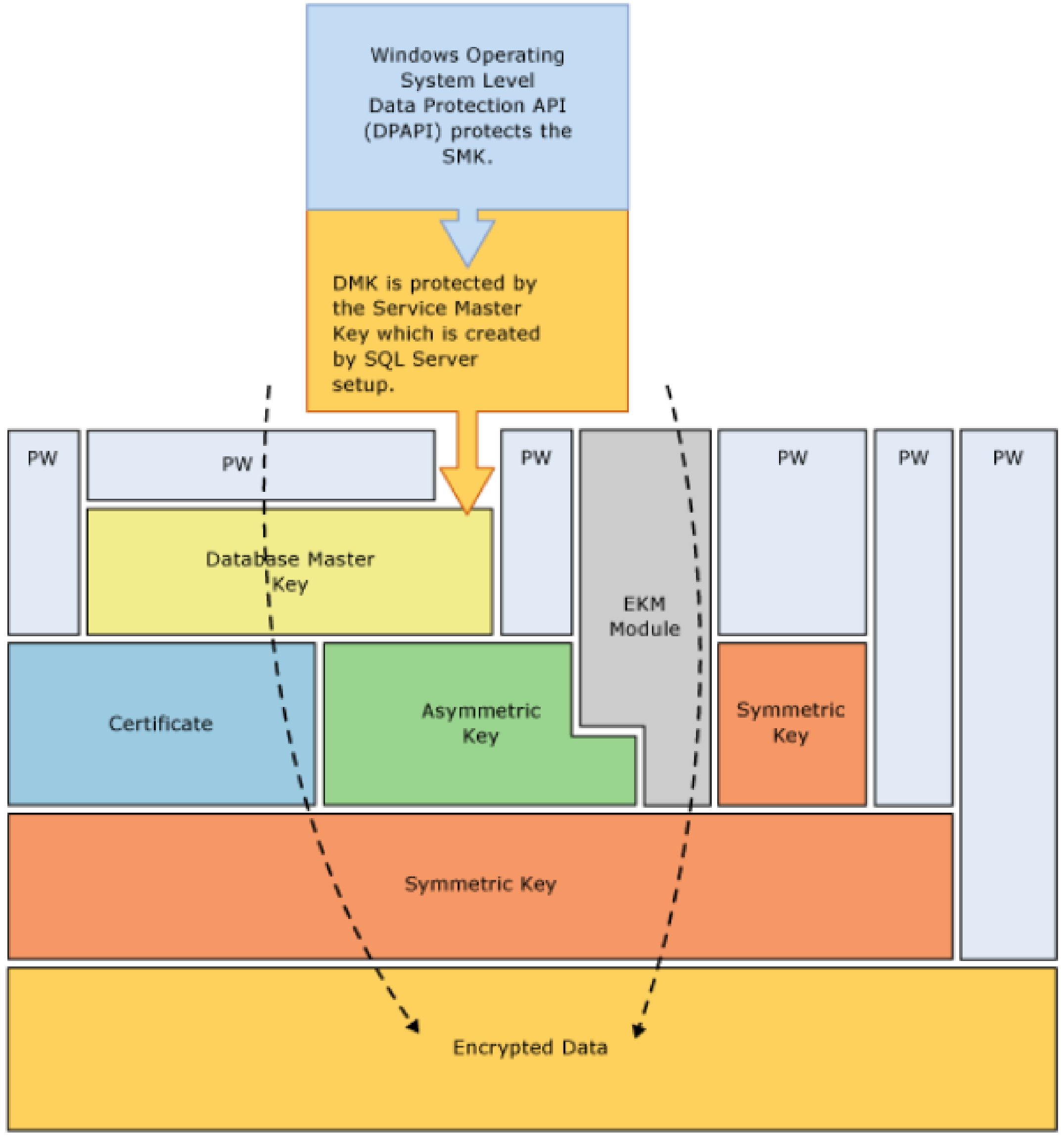
Encryption - Concepts

- Symmetric Encryption
  - Same key can be used to encrypt and decrypt
  - Faster/more efficient, but less secure
- Assymetric Encryption
  - Key used to encrypt cannot decrypt
  - Pubilc/Private keys
  - More secure, less efficient

Encryption - Concepts

- **Multiple layers of Protection for SQL Encryption Keys/Certs.**
- **DPAPI** – Windows OS Data Protection API
- **SMK** – Service Master Key
  - Created by the instance at setup. Managed by both the service account and the system account.
  - Note – don’t change both accounts at the same time or you’re gonna be in big trouble.
- **DMK** – Database Master Key
  - Lives in master, is protected by the SMK.
- **Certificate**
  - Also lives in master, Protected by the DMK. **DEK** – Database Encryption Key.
  - This key is decrypted by the Certificate.
  - This is the key that is actually used to encrypt/decrypt the database

Encryption – SQL Server Encryption Hierarchy



Windows Operating System Level Data Protection API (DPAPI) protects the SMK.
DMK is protected by the Service Master Key which is created by SQL Server setup.
PW
PW
PW
PW
PW
PW
Database Master Key
EKM Module
Certificate
Asymmetric Key
Symmetric Key
Symmetric Key
Encrypted Data

https://docs.microsoft.com/en-us/sql/relational-databases/security/encryption/encryption-hierarchy?view=sql-server-2017

Encryption - Concepts

- Storing and protecting SQL Encryption Keys is crucial
  - Third party key management systems are recommended
  - One expert suggests storing them on 2 different CD’s, in sealed envelopes with manager’s signature on it, locked away in 2 different offsite locations.

Encryption – “At Rest” and “On the wire”

We’ll deal with 2 main categories of encryption:

- **At Rest** (on disk/storage)
  - In the Backup files
  - In the Database files themselves
  - Different choices for encryption algorithm, but don’t even bother with older ones. Minimum is AES 128. There is also AES 192 and AES 256.
  - Tradeoff of Performance vs Security –
    - My analysis showed AES 256 slowed performance on index rebuilds by 50% over AES 128, but we eventually went with it due to regulatory requirements.
- **On the wire** (over the network)
  - All network communication with clients.
  - SSL encryption. Can be optional or forced.

Encryption – “At Rest”

- **Backups**
  - SQL has the ability to create backups “WITH ENCRYPTION”
  - This reduces the problem of File Security for backups. An encrypted backup is useless without the certificate.
- **Column-Level Encryption**
  - SQL Server can encrypt individual columns of data
  - Even authenticated users cannot access data without the key/password
  - Requires code changes
  - Can cause performance issues with indexes, sorts, “GROUP BY”, etc., since encryption algorithm is nondeterministic.

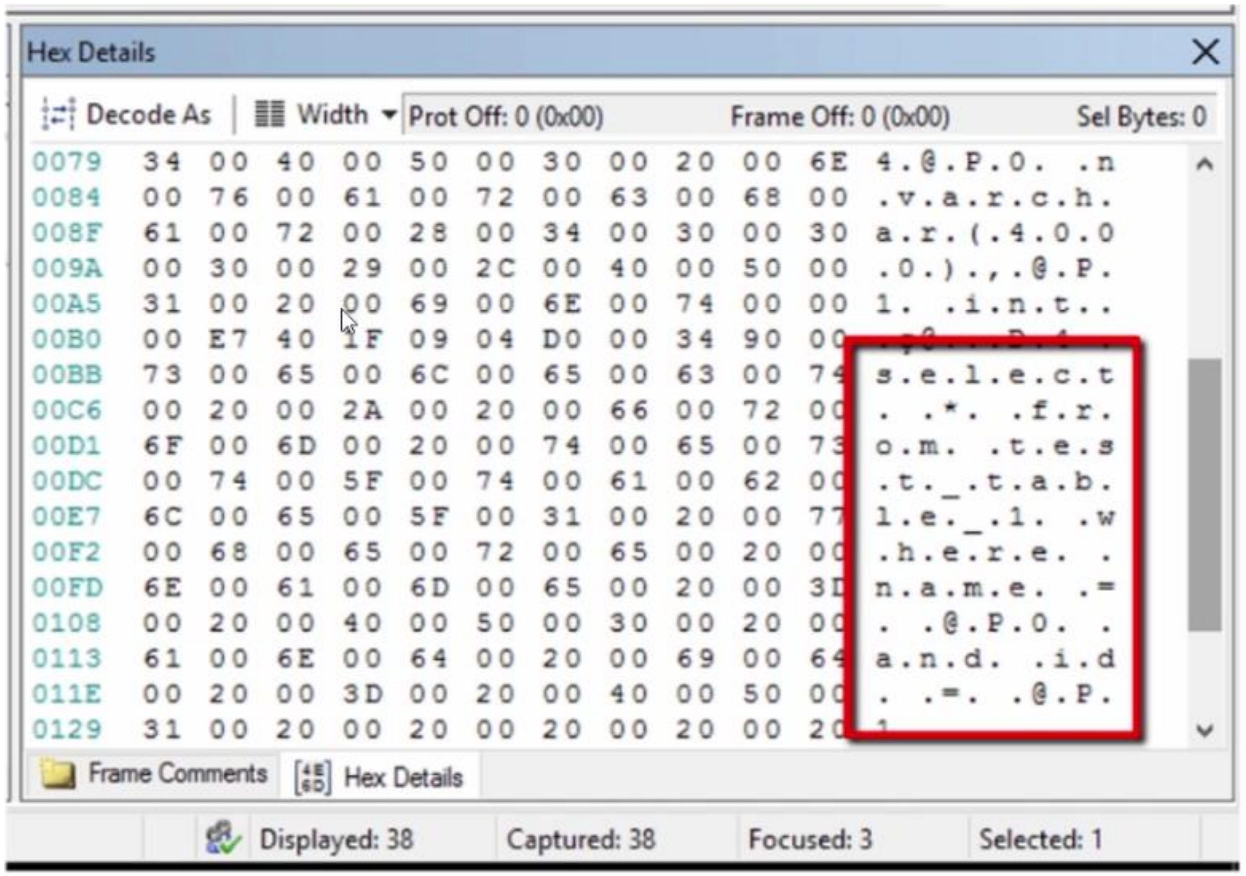
Encryption – "At Rest"

- **TDE** – Transparent Data Encryption
  - The database file(s) and log file are encrypted, but authenticated logins can query normally.
  - **NO CODE CHANGES** required.
  - **Backups** are automatically encrypted as well.
    - NOTE – pre-2016, this kills the ability to compress backups.
    - This was a major impact for my team, as we needed come up with 10+ TB extra space for backups once TDE was implemented
    - Post-2016, compression for TDE backups is back, yay! There were some problems in earlier builds, so make sure you are at *least* at SQL 2016 **SP2 CU2**
  - Performance implications

Encryption – “On the wire”

There are multiple attacks that can be used to intercept network traffic.

WireShark



Hex Details

Decode As | Width | Prot Off: 0 (0x00) | Frame Off: 0 (0x00) | Sel Bytes: 0

```
0079  34 00 40 00 50 00 30 00 20 00 6E  4.@.P.0. .n
0084  00 76 00 61 00 72 00 63 00 68 00  .v.a.r.c.h.
008F  61 00 72 00 28 00 34 00 30 00 30  a.r.(.4.0.0
009A  00 30 00 29 00 2C 00 40 00 50 00  .0.).,.@.P.
00A5  31 00 20 00 69 00 6E 00 74 00 00  1. .i.n.t..
00B0  00 E7 40 1F 09 04 D0 00 34 90 00  [illegible]
00BB  73 00 65 00 6C 00 65 00 63 00 74  s.e.l.e.c.t
00C6  00 20 00 2A 00 20 00 66 00 72 00  . .*. .f.r.
00D1  6F 00 6D 00 20 00 74 00 65 00 73  o.m. .t.e.s
00DC  00 74 00 5F 00 74 00 61 00 62 00  .t._.t.a.b.
00E7  6C 00 65 00 5F 00 31 00 20 00 77  l.e._.1. .w
00F2  00 68 00 65 00 72 00 65 00 20 00  .h.e.r.e. .
00FD  6E 00 61 00 6D 00 65 00 20 00 3D  n.a.m.e. .=
0108  00 20 00 40 00 50 00 30 00 20 00  . .@.P.0. .
0113  61 00 6E 00 64 00 20 00 69 00 64  a.n.d. .i.d
011E  00 20 00 3D 00 20 00 40 00 50 00  . .=. .@.P.
0129  31 00 20 00 20 00 20 00 20 00 20  1
```

Frame Comments | Hex Details

Displayed: 38 | Captured: 38 | Focused: 3 | Selected: 1

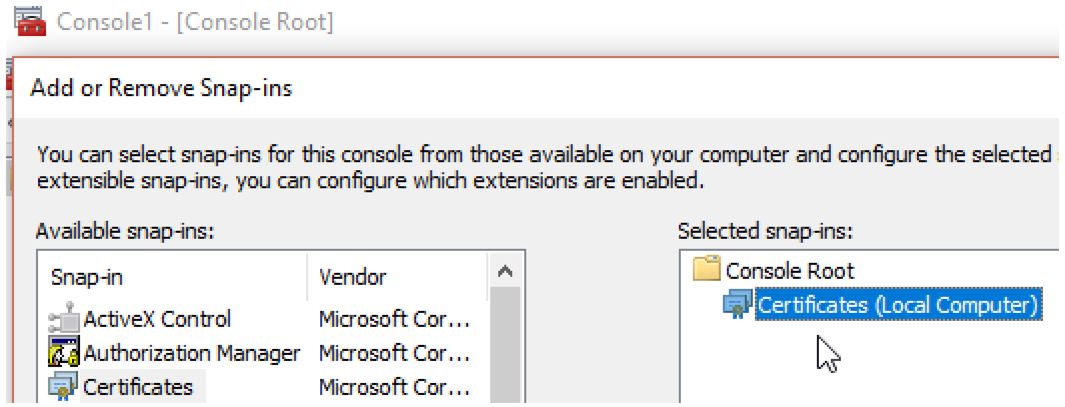
Encryption – “On the wire”

To encrypt data on the wire, we implement SSL.

- First, request and install a certificate
  - Best practice is to get a cert from a Certificate Authority (like DigiCert), or an internal authority if your company has one
  - Self-signed certificates are allowed, but this is not secure from certain attacks. In this case the client connect string must contain “TrustServerCertificate=true”

Encryption – “On the wire”

- Certificate request and installation are done in: Management Console - Snapin – Certificate
- Make sure to request at least 2048 bits encryption.


Console1 - [Console Root]

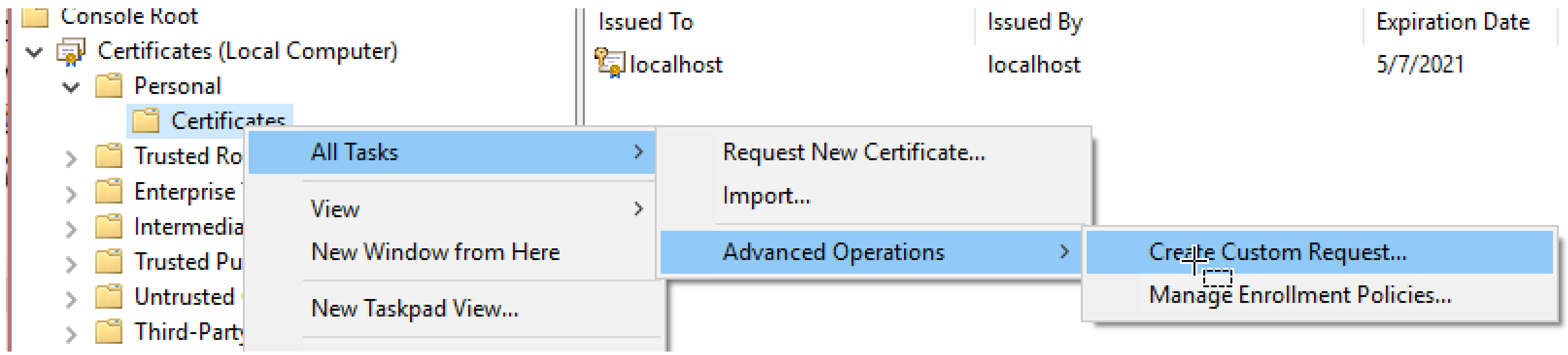
Add or Remove Snap-ins

You can select snap-ins for this console from those available on your computer and configure the selected
extensible snap-ins, you can configure which extensions are enabled.

Available snap-ins:

Snap-in	Vendor
ActiveX Control	Microsoft Cor...
Authorization Manager	Microsoft Cor...
Certificates	Microsoft Cor...

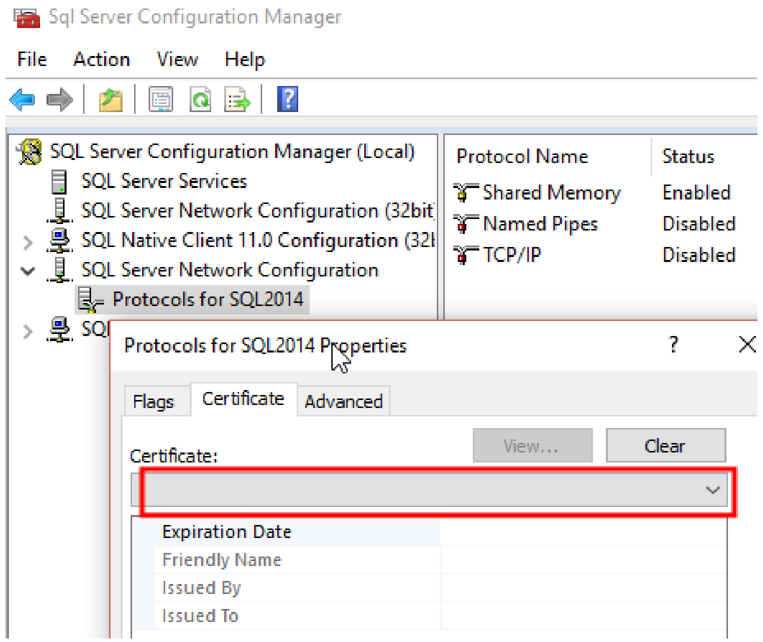
Selected snap-ins:



Issued To	Issued By	Expiration Date
localhost	localhost	5/7/2021

Encryption – “On the wire”

- Once cert is installed, go to SQL Config Mgr, “Certificate” tab, and select your certificate from the drop-down
- At this point any client can request SSL encryption by adding “encrypt=true” in the connect string


Sql Server Configuration Manager
File
Action
View
Help
SQL Server Configuration Manager (Local)
SQL Server Services
SQL Server Network Configuration (32bit
SQL Native Client 11.0 Configuration (32
SQL Server Network Configuration
Protocols for SQL2014
SQ
Protocols for SQL2014 Properties
Flags
Certificate
Advanced
View...
Clear

Protocol Name	Status
Shared Memory	Enabled
Named Pipes	Disabled
TCP/IP	Disabled

Certificate:

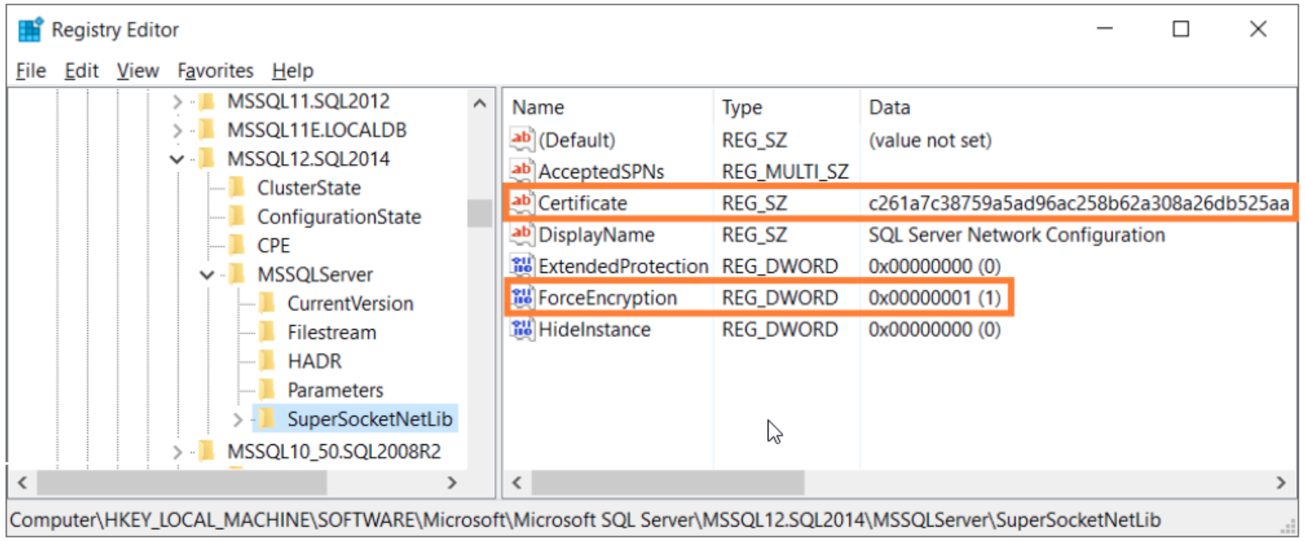
Expiration Date	
Friendly Name	
Issued By	
Issued To	

Encryption – "On the wire"

- Note – the Internally Authorized cert provided by my company didn't appear in that drop-down, but this is a known issue. There is a Registry work-around to enable SSL using the certificate fingerprint.

- https://stackoverflow.com/questions/36817627/ssl-certificate-missing-from-dropdown-in-sql-server-configuration-manager

What one need to do one can in the Registry under the key like `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\MSSQL12.SQL2014\MSSQLServer\SuperSocketNetLib` , where the part `MSSQL12.SQL2014` can be a little different in your case. The SQL Server Configuration Manager help us to set two values in the registry: `ForceEncryption` and `Certificate` :

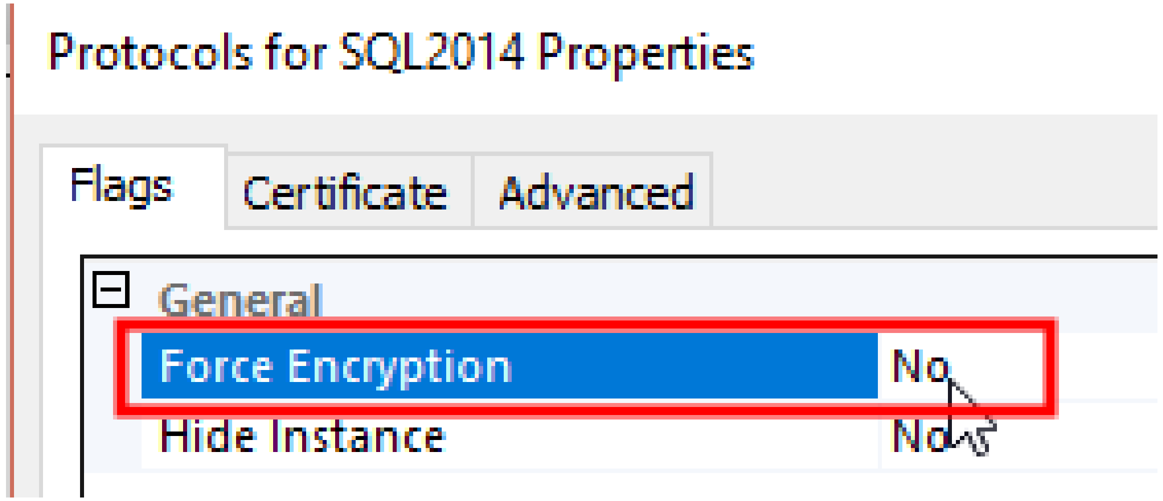


Name	Type	Data
(Default)	REG_SZ	(value not set)
AcceptedSPNs	REG_MULTI_SZ	
Certificate	REG_SZ	c261a7c38759a5ad96ac258b62a308a26db525aa
DisplayName	REG_SZ	SQL Server Network Configuration
ExtendedProtection	REG_DWORD	0x00000000 (0)
ForceEncryption	REG_DWORD	0x00000001 (1)
HideInstance	REG_DWORD	0x00000000 (0)

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\MSSQL12.SQL2014\MSSQLServer\SuperSocketNetLib

Encryption – “On the wire”

- When ready, you can force encryption on all clients. Protocols – “Flags” tab – “Force Encryption” dropdown – set to Yes
- Make sure all clients are prepared for that first!
    - We had a few customers where this broke their connections, we had to assist
    - One customer still can’t authenticate due to a known issue with old JDBC drivers, we’re migrating them to a different environment.



Protocols for SQL2014 Properties

Force Encryption No
Hide Instance No

Encryption – “On the wire”

- SSL protects:
  - Data within network packets in transit
- SSL does *not* protect:
  - Data at rest in SQL Server
  - Data in Memory in SQL Server
  - Data once it reaches the client

- Is there any solution that comes closer to doing all those things?

Encryption – New functionality

“**Always Encrypted**”

- New with SQL 2016
- Data is encrypted on the disk, in memory, and in transit.
- All encryption/decryption is handled by the client.
- Upsides:
  - Additional encryption workload is distributed to multiple clients rather than concentrated on SQL engine
  - Data is encrypted all throughout the process, only the client sees the raw data.
- Downside:
  - Requires changes to client code, so cost and time to implement might be a factor. DBA’s can’t implement without major effort from dev team.

Encryption – Side Note

Side note on encryption – “WITH ENCRYPTION” option on stored procedure code is **NOT SECURE**.

- This is a known vulnerability and can be easily cracked, especially if one has sysadmin access.
- So don’t depend on this to protect anything proprietary.
- https://www.devart.com/dbforge/sql/sqldecryptor/download.html

Other New functionality - DDM

"**DDM**" – Dynamic Data Masking

- Also new with SQL 2016
- Limits sensitive data exposure by masking it to non-privileged users
- Easy to use in existing apps
  - Masking rules are automatically applied in the query results via functions.
  - Sensitive data can be masked without modifying existing queries.
- Summary:
  - A central data masking policy acts directly on sensitive fields in the database.
  - Designate privileged users or roles that do have access to the sensitive data.
  - DDM features full masking and partial masking functions, as well as a random mask for numeric data.
  - Simple Transact-SQL commands define and manage masks.

Other New functionality - DDM

- Note, there are vulnerabilities to DDM we need to be aware of.
- The data still resides unmasked in the database. This means that predicate logic can be applied to infer masked values.
- An example would be Employee A wanting to know Employee B's salary information. The salary column is masked, but this value can be derived using <, >, etc. Eventually Employee A can pinpoint what the salary is. (Bracketing)

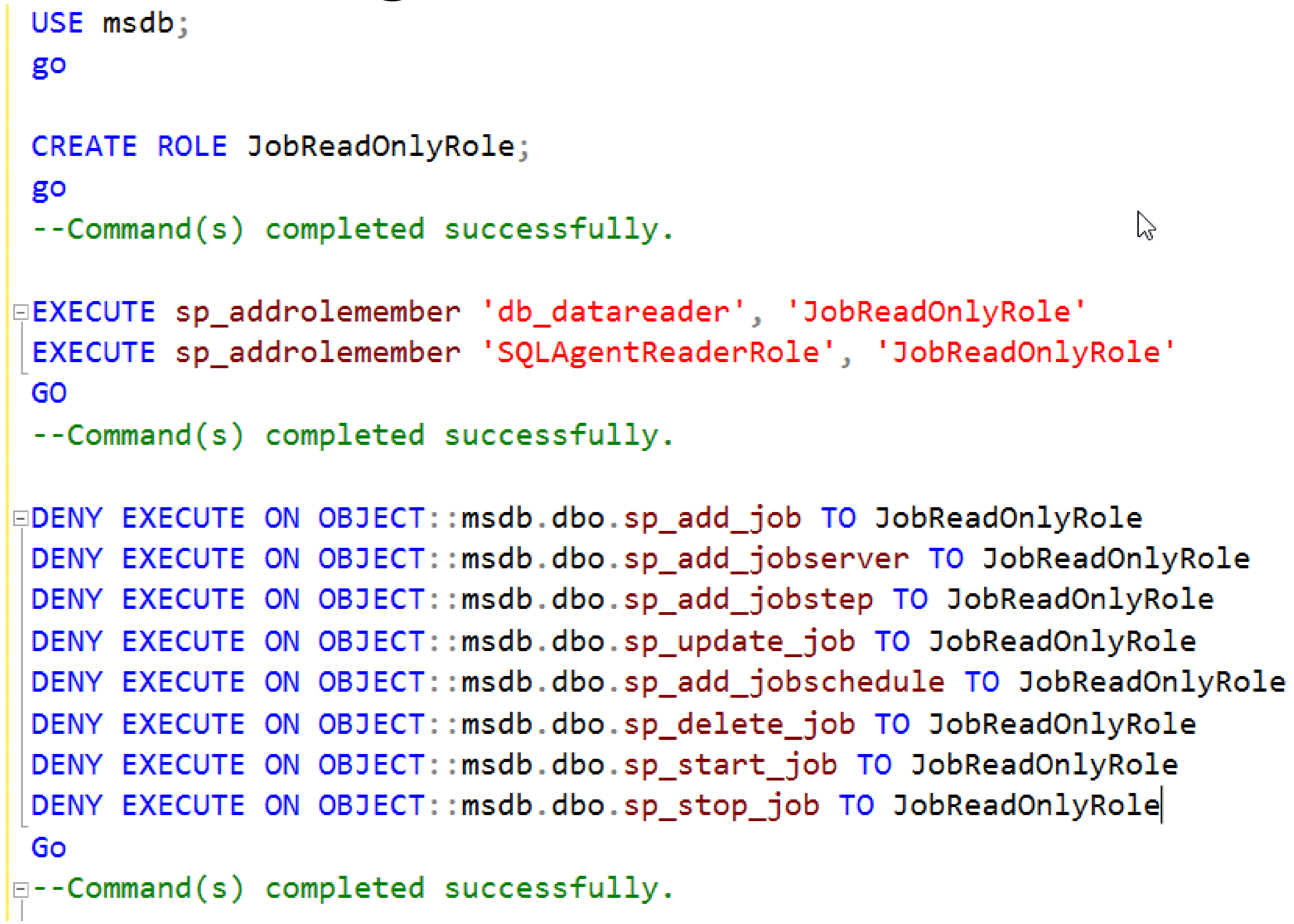
Other New Functioniality - RLS

- **RLS** – Row Level Security
- Provides per-row read and write access control, based on attributes of the executing user, with minimal schema, application, or query changes
- Previously this would be implemented using views
- An example would be letting sales managers see only their own region in a sales table.

Other Thoughts – NonStandard Permissions

- Standard SQL roles sometimes don’t fit the bill
- SQL Server Agent Fixed Database Roles
  - SQLAgentUserRole, SQLAgentReaderRole, SQLAgentOperatorRole
- What if you want developers to be able to see properties and history for all prod jobs, but prevent them from starting, stopping, editing, creating, deleting, etc?
- Can’t do it with these roles, they give either too much or not enough
- Create a role in “msdb”, add it to “SQLAgentOperatorRole”, and then explicitly DENY

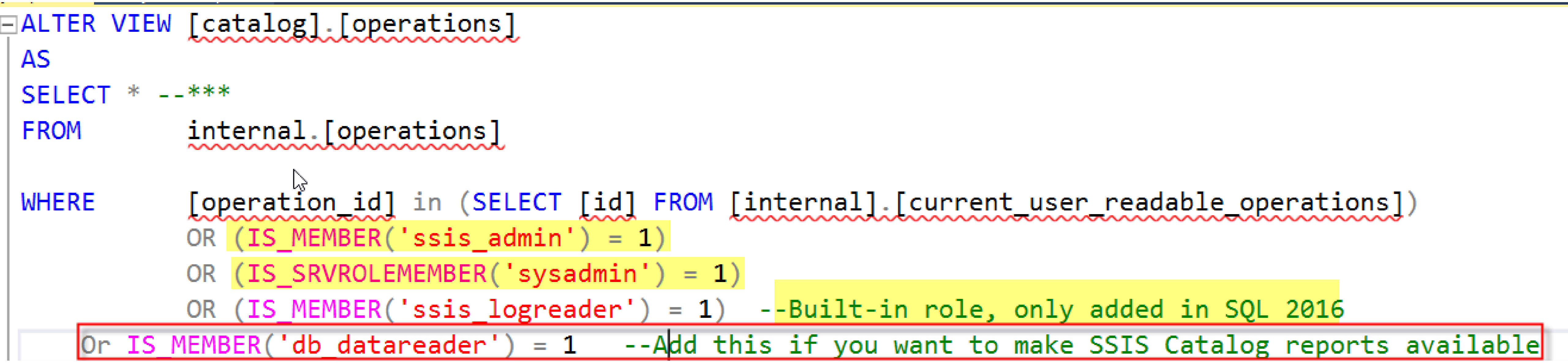
Other Thoughts – NonStandard Permissions



```
USE msdb;
go

CREATE ROLE JobReadOnlyRole;
go
--Command(s) completed successfully.

EXECUTE sp_addrolemember 'db_datareader', 'JobReadOnlyRole'
EXECUTE sp_addrolemember 'SQLAgentReaderRole', 'JobReadOnlyRole'
GO
--Command(s) completed successfully.

DENY EXECUTE ON OBJECT::msdb.dbo.sp_add_job TO JobReadOnlyRole
DENY EXECUTE ON OBJECT::msdb.dbo.sp_add_jobserver TO JobReadOnlyRole
DENY EXECUTE ON OBJECT::msdb.dbo.sp_add_jobstep TO JobReadOnlyRole
DENY EXECUTE ON OBJECT::msdb.dbo.sp_update_job TO JobReadOnlyRole
DENY EXECUTE ON OBJECT::msdb.dbo.sp_add_jobschedule TO JobReadOnlyRole
DENY EXECUTE ON OBJECT::msdb.dbo.sp_delete_job TO JobReadOnlyRole
DENY EXECUTE ON OBJECT::msdb.dbo.sp_start_job TO JobReadOnlyRole
DENY EXECUTE ON OBJECT::msdb.dbo.sp_stop_job TO JobReadOnlyRole
Go
--Command(s) completed successfully.
```

Other Thoughts – NonStandard Permissions

- Another example –
- Allowing developers to see history of SSISDB package executions, not possible without granting way too much other permission.
- Note that the SSIS Catalog views have custom code restricting access to certain built-in roles. Pre-2016, requires altering the view.



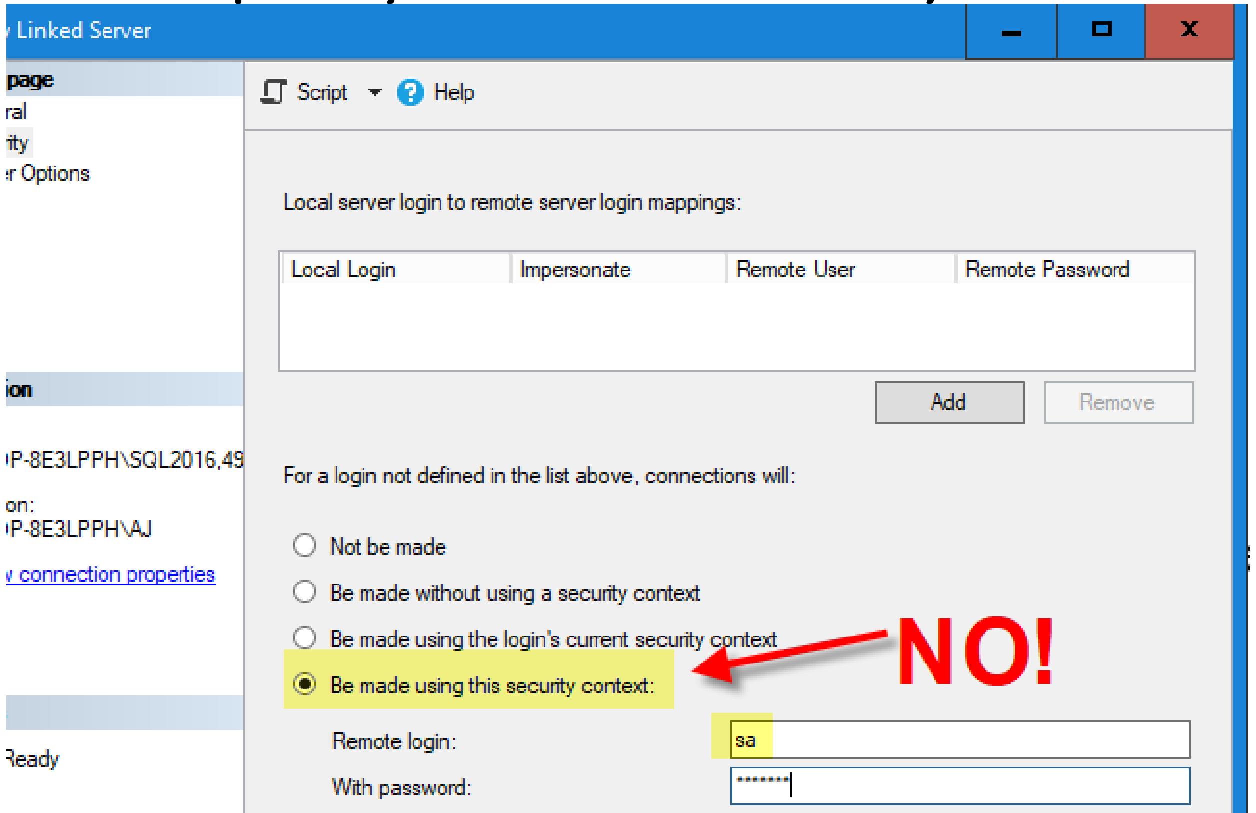
```
ALTER VIEW [catalog].[operations]
AS
SELECT * --***
FROM        internal.[operations]

WHERE       [operation_id] in (SELECT [id] FROM [internal].[current_user_readable_operations])
            OR (IS_MEMBER('ssis_admin') = 1)
            OR (IS_SRVROLEMEMBER('sysadmin') = 1)
            OR (IS_MEMBER('ssis_logreader') = 1)  --Built-in role, only added in SQL 2016
    Or IS_MEMBER('db_datareader') = 1    --Add this if you want to make SSIS Catalog reports available
```

Other Thoughts – Ownership Chaining

- You can use **Ownership Chaining** to grant execute rights to functions or procs that access objects that the user can't access.
- Just put the clause "`WITH EXECUTE AS OWNER`" in the proc/udf declaration, and then deny explicit rights to the underlying objects.
- Users will be able to interface with the objects only through the approved code.
- You can even use "Cross-Database" ownership chaining to give users access to objects in other databases on which they have no rights.

Other Thoughts – Linked Servers

- General wisdom – they're ***EVIL***..
- Can be a sneaky way for unauthorized users to get elevated access
- **Never** specify a default security context



Local server login to remote server login mappings:

Local Login	Impersonate	Remote User	Remote Password

For a login not defined in the list above, connections will:

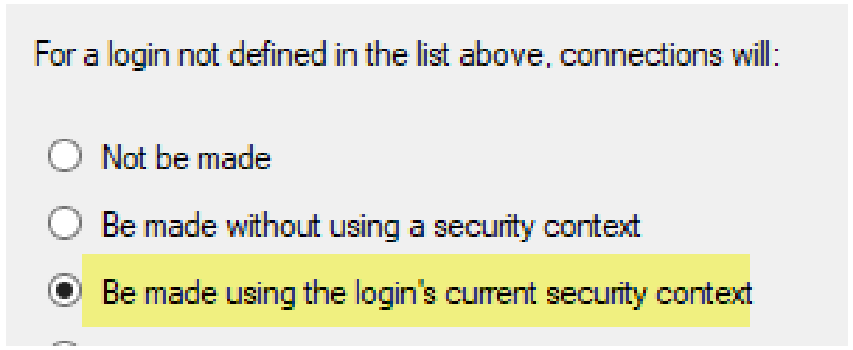
- Not be made
- Be made without using a security context
- Be made using the login's current security context
- [✓ Be made using this security context:]

Remote login: sa

With password: *******

Other Thoughts – Linked Servers

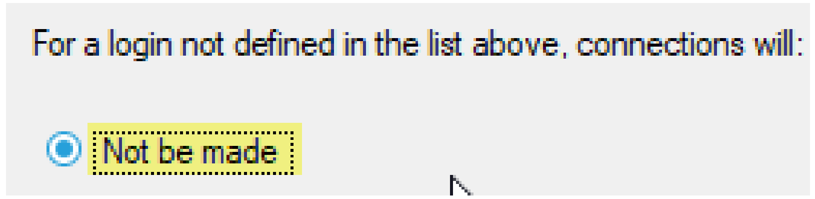
- If you MUST use them –
- In the Security tab, “Be made using the login’s current security context”, which utilizes AD authentication. (Note, this means Kerberos authentication must be enabled)



For a login not defined in the list above, connections will:

- ○ Not be made
- ○ Be made without using a security context
- ● Be made using the login's current security context

- Other Option (not ideal) – specify a list of authorized users, map to a pre-defined remote login with minimum privileges and, disable undefined login connections.

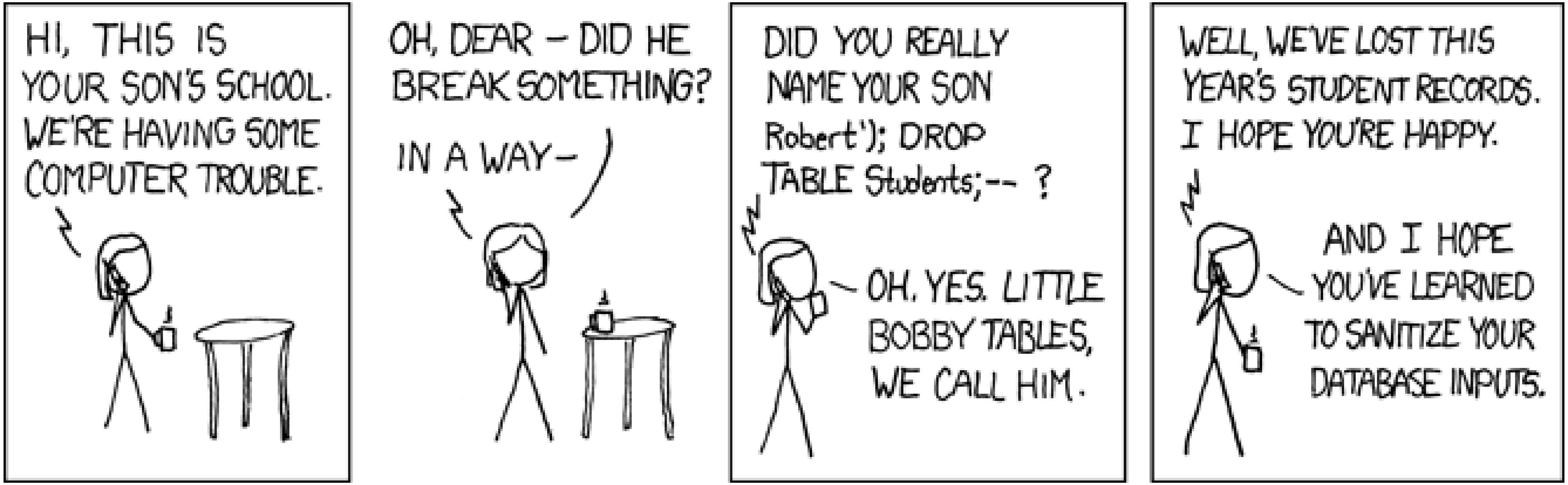


For a login not defined in the list above, connections will:

- ● Not be made

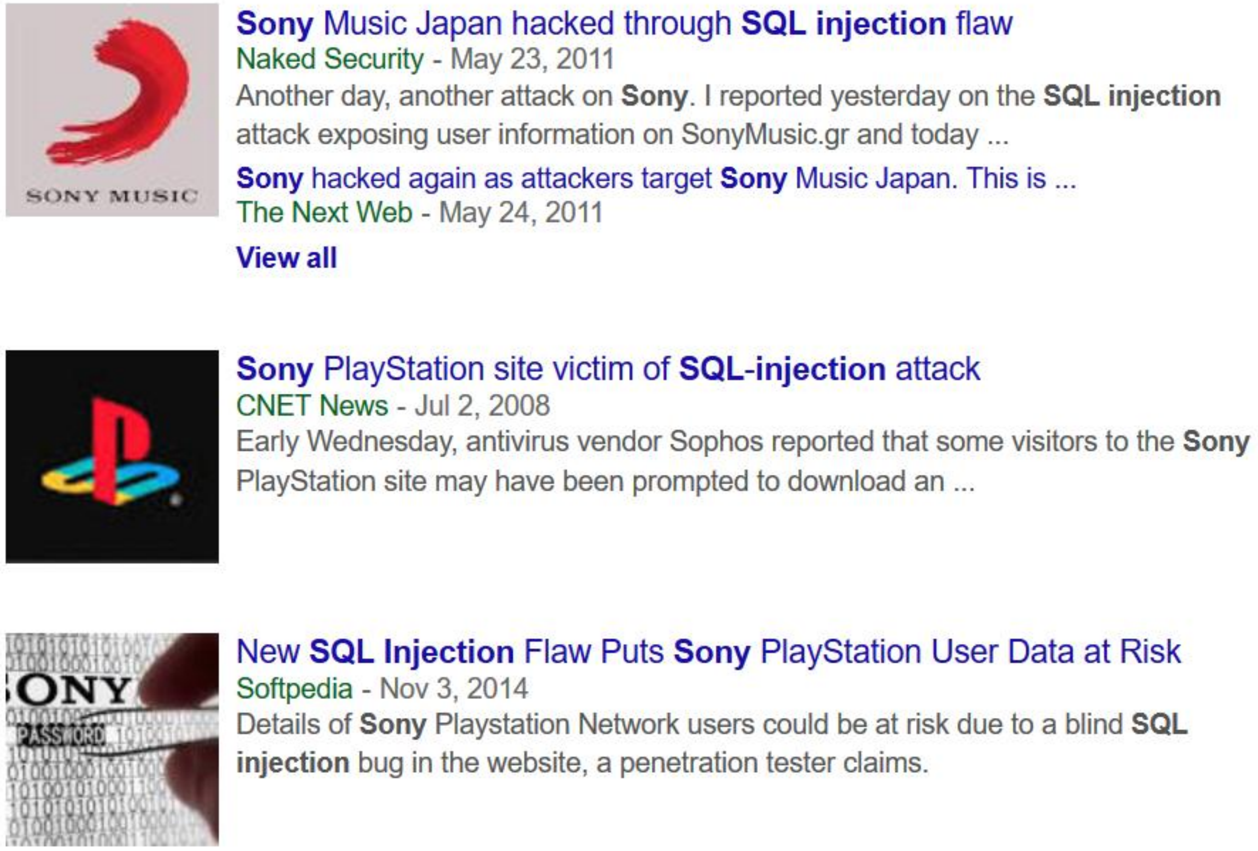
Defending against SQL Injection

- Everybody’s favorite SQL-based attack.


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Defending against SQL Injection

- It was used against Sony Pictures in 2011, a very public failure for their security



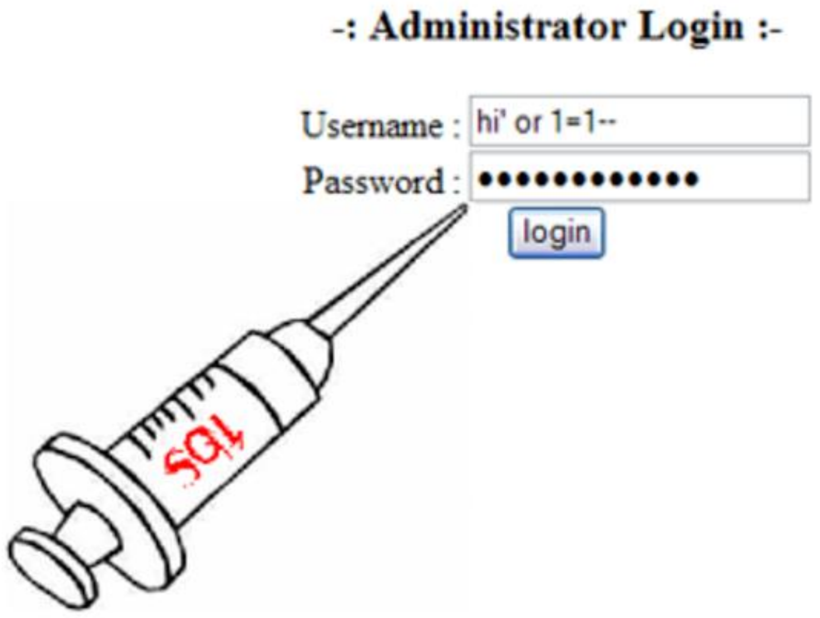
Sony Music Japan hacked through **SQL injection** flaw
Naked Security - May 23, 2011
Another day, another attack on **Sony**. I reported yesterday on the **SQL injection** attack exposing user information on SonyMusic.gr and today ...
Sony hacked again as attackers target **Sony** Music Japan. This is ...
The Next Web - May 24, 2011
View all

Sony PlayStation site victim of **SQL-injection** attack
CNET News - Jul 2, 2008
Early Wednesday, antivirus vendor Sophos reported that some visitors to the **Sony** PlayStation site may have been prompted to download an ...

New **SQL Injection** Flaw Puts **Sony** PlayStation User Data at Risk
Softpedia - Nov 3, 2014
Details of **Sony** Playstation Network users could be at risk due to a blind **SQL injection** bug in the website, a penetration tester claims.

Defending against SQL Injection

- General idea is to send arbitrary SQL commands to the server by adding commands to an already existing query, by sneaking it into a parameter field.


-: Administrator Login :-

Username : hi' or 1=1--

Password : •••••••••••••

Defending against SQL Injection

- Typical SQL Injection Types:
  - SQL where the where clause has been short-circuited by the addition of a line such as 'a'='a' or 1=1
  - SQL where the where clause has been truncated with a comment i.e --
  - SQL where the addition of a union has enabled the reading of a second table or view
  - Stacking Queries, executing more than one query in one transaction
  - SQL where an unintentional SQL statement has been added
  - SQL where an unintentional sub-select has been added
  - SQL where built-in or bespoke package procedures are called where they should not be
  - SQL where access is made to system tables and/or application user and authentication tables

Defending against SQL Injection

- Example:
- An ASP.NET app that dynamically builds a SELECT query with a WHERE clause.
- Vulnerable query would be:

```
select * from customers where customerID = '1234'
```

- Exploit would look like this:

```
select * from customers where customerID = '1234'
union
SELECT table_name,null,null,null,null,null,null,null,null,
          null,null FROM INFORMATION_SCHEMA.TABLES --'
```

Defending against SQL Injection

Defense:

- **Parameterization** is the best solution for SQL injection attacks
  - Or at least be very consistent about handling single quotes
  - There are best practices for this, make sure the developers are aware
- Limit database permissions and segregate users
  - Ownership chaining comes in here
- Use stored procedures for database access
- Isolate the webserver
- Configure error reporting
- Overall, applying **Principle of Least Privilege** will always come in handy.

Dealing with Audits

Dealing with auditors is a delicate balancing act.

You want to be helpful/cooperative, but remember who you’re talking to.

Try to remember Jimmy Conway’s 2 most important rules of life:

Dealing with Audits

- Give them exactly what they asked for, no more.
    - If an auditor asks do you know the time, what answer do you give?
- Don’t try to be “Mr. Helpful” with extra info they didn’t ask for. Whenever I do that it ends up expanding the scope and it becomes a full time job.
- Having good change control, logging, and audit processes will help make things easier.

Thank You!

- Thomas.AJ.Hull@gmail.com
- **@SDSQL**
- Linkedin.com/in/**AJHull**
- **Github: AJHull**