SQLskills Immersion Event
IE0: Accidental/Junior DBA

Module 3: Security

Jonathan Kehayias
Jonathan@SQLskills.com

Overview

- Securing SQL Server
- Authentication
- Authorization
- Auditing access
- SQL injection risks
- Database encryption

Overview

- **Securing SQL Server**
  - Physical security
  - Secure installation
- **Authentication**
- **Authorization**
- **Auditing access**
- **SQL injection risks**
- **Database encryption**

SQLskills immerse yourself in sql server

Securing SQL Server

- **How do you ensure that SQL Server is completely secure?**
  - Power it off
  - Unknown vulnerabilities make it impossible to guarantee a server is 100% secure unless it is not available at all
- **Understanding and following best practices for security will make SQL Server more secure and problems less likely**
  - Defense in depth
  - Hardening server configurations to minimize easy access
  - Might not completely protect a server, but simple configuration best practices will discourage most attacks

SQLskills immerse yourself in sql server

Physical Security

- Access to the physical server hardware should be secure and available only to those who need access
- Ideally access to the physical hardware should be logged
  - Simple as a log book or as complex as a door badge system that logs all accesses to a specific door reader
- Annual reviews of who currently has access to physical server hardware should occur
- Backup media should be secured to prevent unauthorized restores to different installations to access important data

Secure Installation

- Only install the services and features required for the server
  - Installing Reporting Services or Analysis Services where they are not used, opens additional security and patching requirements with no benefits
- Apply the principal of least privilege for all service accounts to minimize exposure if compromised
- Antivirus software should be installed and configured with appropriate SQL Server exclusions
  - http://support.microsoft.com/kb/309422

Overview

- **Securing SQL Server**
- **Authentication**
  - Authentication modes
  - Principals
  - Server and database roles
- **Authorization**
- **Auditing access**
- **SQL injection risks**
- **Database encryption**

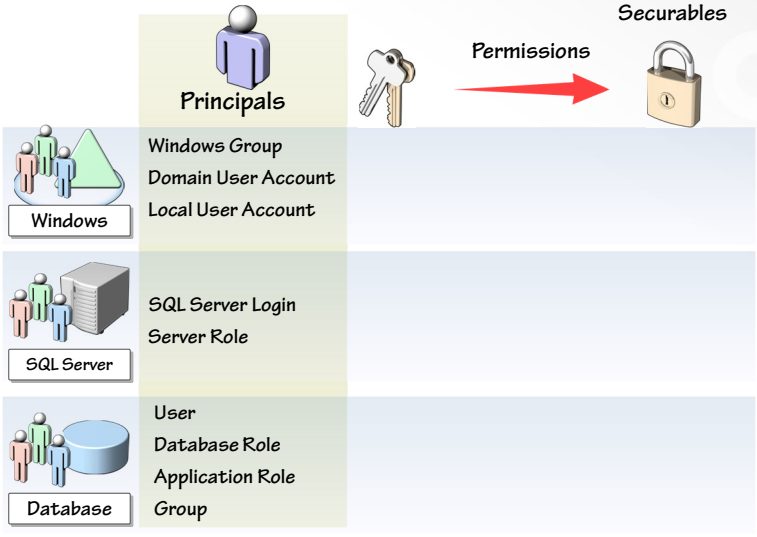
Authentication Modes

- **SQL Server offers two methods of authentication**
  - Windows Authentication only – commonly referred to as Integrated Security
  - SQL Server and Windows Authentication – commonly referred to as Mixed-mode Security
- **If possible rely on Windows Authentication to take advantage of Active Directory**
  - Groups, account policies, and separation of duties provide easier management and auditing controls on changes
- **For applications requiring SQL Server authentication**
  - Secure the sa account with a secure password and do not use this account for general administration purposes
  - Configure Windows password policies and enforce password complexity and expiration in SQL Server
    - May require coding application to handle expiration error messages

Authentication and Principals

- **Authentication in SQL Server identifies who you are to the application and vouches for your credentials**
  - What you know (e.g. login and password combination)
  - What you have (e.g. Kerberos ticket, common-access-card)
  - Who you are (e.g. biometrics)
- **Authentication principals in SQL Server are Logins**
  - Windows Logins – provides single sign-on for OS and SQL Server
    - User or Groups
  - SQL Logins – allows application specific security and mixed OS access
    - Defined internally in SQL Server
    - May be a login and password combination or certificate based

What Are Principals?



The “sa” Login in SQL

- **The sa account is incredibly important and must be secured**
  - Cannot be disabled or locked out if using SQL Authentication
  - Some service providers and hosting companies will rename this account
- **Should not be used for administration tasks**
  - Each administrator should have their own login for administration that is ideally associated with a Windows authenticated account
  - Commonly the dbo for databases and the SQL Agent Job owner to prevent orphaned login issues from user accounts

SQL Logins - Secure By Default

- **Login Policies**
  - Server Side
    - Check_Policy - Default ON
    - Check_Expiration - Default OFF
      - Requires Check_Policy ON
    - Must_Change - Default OFF
      - Requires - Check_Expiration ON
  - Client Side API Support
    - Password Change at login
- **User must have a way to change password**
  - Available in SSMS, SQLCMD
  - API available for clients

SIDs

- **Logins are assigned a SID and an ID**
  - Windows Login - SID of the Windows principal
  - Certificates and Asymmetric Keys - SID based on hash of public key
  - SQL Logins - SID generated by SQL Server
- **ID is assigned by SQL Server**
  - IDs are reused
- **Use SSIS transfer login task to migrate logins**
  - Or script in KB918992 (sp_help_revlogin)
  - Uses hashed password

Server Roles

- **SQL Server includes pre-defined server roles**
  - Manage members with
    - sp_[add/drop]srvrolemember pre-SQL Server 2012
    - ALTER SERVER ROLE in SQL Server 2012 onward
  - sys.server_role_members to get a list
  - Check membership with IS_SRVROLEMEMBER
- **Each server role has pre-defined privileges**
  - sp_srvrolepermission to get a list
- **You can define you own server roles from SQL Server 2012 onward**

Sysadmin role

- **Sysadmin is the "superuser role"**
  - SA is mapped to sysadmin
  - NT Authority\System and other system accts
    - For Windows Update, other functions
  - SQL Server and SQL Agent services accounts
  - Windows admins are mapped to sysadmin
    - Until SQL Server 2008
- **Sysadmin is dbo of every database**
- **Granting CONTROL SERVER has most of the same permissions as sysadmin**

Database Users

- **Each database contains its own users**
  - Users map to logins and can be:
    - Windows logins
    - SQL logins
    - Windows groups
    - User without login
- **Special Users**
  - DBO - Database Owner
    - The (one) owner of the database
    - Sysadmin mapped to DBO in every database
    - DBO bypasses security checks on database objects
  - Guest
    - Logins without a mapping map to Guest
    - Guest cannot connect by default to user databases

Mismatched SIDs

- User ID points to Login ID (SID)
- Login SID for same named use will differ between instances for SQL Logins but be identical for Windows Logins
- When a database is moved to a new instance, the SQL Login SID will not match unless:
  - You use sp_change_users_login or ALTER USER to fix the SID
  - You create the SQL Login with a specific SID using the script in KB918992

Database Roles

- SQL Server includes pre-defined roles
  - Some pre-defined roles in every database
    - db_owner (!= dbo) and others
  - Some pre-defined roles in MSDB for
    - SSIS admin and users
    - Data Collection
    - Policy.. and others
- You can create your own roles
- A user can be a member of 1-n roles
- Each user is member of public role
  - Equivalent to everyone role in Windows
  - You cannot remove a user from public role

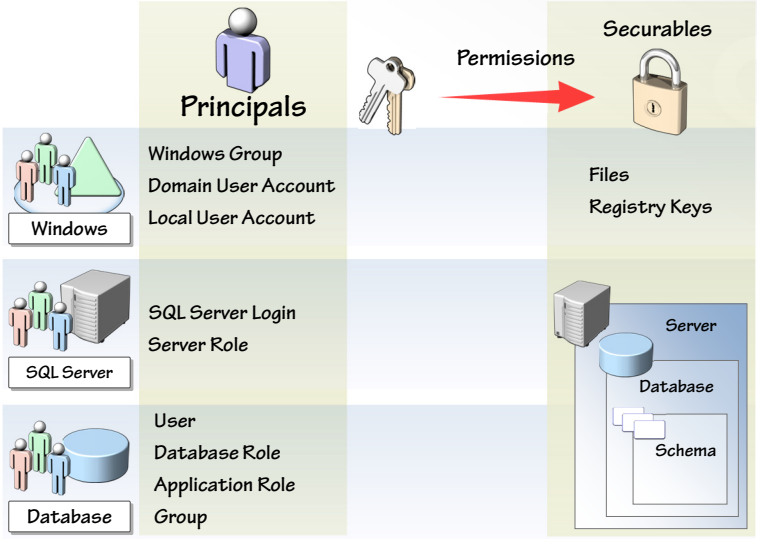
Managing Database Roles

- **Manage role members with**
  - sp_[add/drop]rolemember pre-SQL Server 2012
  - ALTER ROLE from SQL Server 2012 onward
- **Use sys.database_role_members to list**
- **IS_ROLEMEMBER to determine membership**

Overview

- **Securing SQL Server**
- **Authentication**
- **Authorization**
  - Securables
  - Permissions
- **Auditing access**
- **SQL injection risks**
- **Database encryption**

What Are Securables?



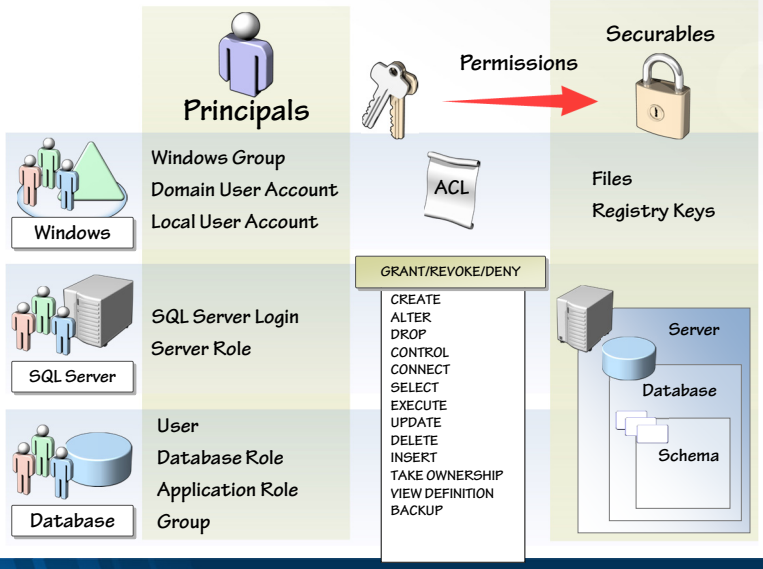
Securables

Server Scope			
Endpoint	Login	Database	Server Role
Availability Group	Event Session		

Database Scope			
User	Role	Application Role	Assembly
Remote Service Binding	Route	Service	Message Type
Asymmetric Key	Schema	Certificate	Fulltext Catalog
Symmetric Key	Contract	Search Property List	Fulltext Stoplist

Schema Scope			
Sequence	Type	Object	Aggregate
Procedure	Constraint	Function	Queue
Synonym	Statistic	View	Table

What Are Permissions?



Database Security Language

- **Three database security verbs**
  - GRANT – positive privilege
  - DENY – negative privilege
  - REVOKE – negate a GRANT or a deny
- **DENY overrides GRANT**
  - If multiple privileges based on different roles

Permission Naming Conventions

- **CONTROL – all permissions**
- **ALTER – change properties (not ownership)**
- **ALTER ANY – create, alter, drop**
  - For server or database level securables
- **TAKE OWNERSHIP**
- **IMPERSONATE**
  - Login or user
- **CREATE**
  - All level of securables
  - Create an object requires alter on the schema
- **VIEW DEFINITION**
- **REFERENCES**
  - For Foreign Key, Schemabinding, etc

Permission Hierarchy

- **Covering Permissions**
  - CONTROL on table gives you SELECT on table
  - SELECT on schema, gives you SELECT on tables
- **Permissions at parent scope**
  - ALTER ANY AUDIT
  - ALTER ANY EVENT NOTIFICATION
- **Full list of permissions in SQL Server BOL**
  - select * from sys.fn_builtin_permissions(null)
  - ...Or by securable class
- **Implying permissions script in BOL**

Database Objects and Permissions

- **Objects and permissions can live at**
  - Instance level
    - e.g ENDPOINTs
  - Database level
    - e.g ASSEMBLY, SERVICES
  - Schema level
    - e.g. TABLES, VIEWS, PROCEDURES, QUEUES
  - Permission at column level

Special Permissions

- **CREATE DATABASE permission**
  - Needed to restore database
  - DBO can restore database
- **ALTER ANY LOGIN**
  - Can reset passwords
- **Server roles have a fixed set of permissions**
  - Permissions fixed for fixed database roles
  - Can assign permissions to user db roles
- **DBCC command and some system procedures still check only role membership**

Overview

- **Securing SQL Server**
- **Authentication**
- **Authorization**
- **Auditing access**
  - Compliance audit requirements
  - SQL Server auditing
  - Auditing with SQL Trace
  - Server Audits
- **SQL injection risks**
- **Database encryption**

Auditing and Compliance

- **Auditing is required for compliance with trans-national and industry standards**
  - European Union Data Protection Directive
  - HIPAA
  - Sarbanes-Oxley
  - Payment Card Industry Data Security Standard

Login Auditing

- **Login information can be written to the error log and therefore the Windows Application Event Log by configuring an instance property**
  - Failed logins only
  - Failed logins and Successful logins
  - None
- **Logging successful logins in this manner can bloat the Application Event Log in Windows making it more difficult to identify problems when they occur**

Auditing

- **SQL Server can support C2 security**
  - C2 superseded by Common Criteria
- **SQL Server 2005 onward supports Common Criteria**
  - SP2 required on SQL Server 2005 (EAL 4+)
  - RIP (residual information protection) – memory overwritten with bit pattern before reuse
  - Login statistics can be viewed
  - Login triggers added
  - Column level GRANT does not override DENY

Auditing with SQL Trace

- **SQL Trace includes Audit Events**
  - Choose events to be audited
  - Run audit trace via server-side trace to a file for best performance and to prevent losing events
- **Both C2 and Common Criteria use Profiler**
  - Overhead when tracing
  - Granularity by using events and filters
  - C2 prohibits startup if audit file full

Limitations of Auditing with SQL Trace

- **Requires a manual implementation**
  - Custom trace scripts inside a stored procedure
  - Marking the stored procedure as a startup procedure
  - Manual file name management to prevent failures
  - Not configured with DDL
    - Unless you know all of the event ids
  - Can be configured using SMO libraries
- **Often captures more data than necessary for auditing requirements due to filtering limitations in SQL Trace**
  - Must load trace into SQL Server to query and further filter results for reporting, or read through lots of events
- **Security of trace file cannot be guaranteed**

SQL Server Auditing

- **Auditing is built into SQL Server 2008 and higher**
- **Configurable with DDL, SSMS, SMO**
- **Auditing goals**
  - Security – audit objects secure
    - Even from DBA
  - Performance – minimal impact
  - Management – easy to manage
  - Discoverability – easy to query
- **Enterprise-only feature**
  - From SQL Server 2012 onward, Server-level audit is in all Editions

Auditing Targets

- **Audit events can be written to**
  - File (local or remote file system)
  - Windows application log
  - Windows security log
- **Writing to security log requires privilege**
  - For SQL Server service account
  - Generate Security Audits

Audit DDL

- **Audits defined at server or database level**
  - CREATE SERVER AUDIT
    - Where to write the audit
    - What to do if audit target unavailable
  - CREATE SERVER AUDIT SPECIFICATION
    - What to audit at server level
    - Can also audit database items
  - CREATE DATABASE AUDIT SPECIFICATION
    - What to audit at database and table level
    - Not column level
- **One AUDIT per audit specification**

Controlling Audit

- **Audits and Specifications not on by default**
  - ALTER SERVER AUDIT
  - ALTER SERVER AUDIT SPECIFICATION
  - ALTER DATABASE AUDIT SPECIFICATION
  - To turn it on/off
- **This action is always audited**

Overview

- **Securing SQL Server**
- **Authentication**
- **Authorization**
- **Auditing access**
- **SQL injection risks**
  - Common patterns
  - Parameterization
  - Minimizing privileges
  - Tools
- **Database encryption**

SQL Injection

- **Exploits database layer of application**
  - Usually caused by concatenating user input to produce a SQL statement
  - Results in execution of user input without validation
  - Also possible with binary injection, but is much more difficult
- **Some results of SQL Injection**
  - Obtain or guess passwords
  - Obtain data
  - Obtain database metadata
  - Drop database objects

Common Patterns

- String concatenation attacks
- String truncation attacks
- Two common patterns
  - Ending a statement with user input

```
'SELECT id, name FROM Table1 WHERE id = ' + input
  with input of: '1; DROP TABLE Table2'
```

  - Ending a conditional statement with user input

```
'SELECT id, name FROM Table1 WHERE id = ' + input
  with input of: '1 OR 0=0'
```


Parameterization

- Always using parameterized statements is the most effective way to prevent
- Useable in
  - Stored procedures (best) or
  - Parameterized queries or
  - sp_executesql (not exec())
- Parameters usable even if multiple SQL statements in a batch
- Do not concatenate user input in stored procedures

Minimizing Privileges

- **Use an account with minimal privileges in applications**
- **Use SQL EXECUTE permission only**
  - Avoid and/or Audit direct table access
- **System metadata secured as of SQL Server 2005**
- **Deny access to system stored procedures**
- **Turn off xp_cmdshell, sp_oa_create, OpenRowset, OpenQuery**
  - Security Best Practices Policies

Tools

- **Microsoft Security Advisory (KB954462) – Rise in SQL Injection Attacks Exploiting Unverified User Data Input**
  - Microsoft Source Code Analyzer for SQL Injection
    - http://support.microsoft.com/kb/954476
  - URLScan
    - Restrict Web requests IIS will process
    - Monitors URLs for suspicious parameters
  - HP Scrawlr
    - Recurse through web code, with start URL
    - Send HTTP request and Analyze HTTP Responses
- **More tools**
  - http://sqlservercode.blogspot.com/2007/05/check-your-sql-for-sql-injection.html

Overview

- Securing SQL Server
- Authentication
- Authorization
- Auditing access
- SQL injection risks
- Database encryption

Transparent Data Encryption

- Entire database is encrypted and protected
- No application changes required
  - Applications do not need to explicitly encrypt/decrypt data
- No restrictions with indexes or data types (except FILESTREAM cannot be encrypted)
- Performance cost is small
- Backups are unusable without key

Enabling TDE

- **Create a master key in master database**
  - CREATE MASTER KEY ENCRYPTION BY PASSWORD = '<UseStrongPwdHere>';
- **Create or obtain a certificate protected by the master key in master database**
  - CREATE CERTIFICATE MyDEKCert WITH SUBJECT = 'My DEK Certificate';
- **Create a database encryption key in the encrypted database and protect it by the server certificate**
  - CREATE DATABASE ENCRYPTION KEY WITH ALGORITHM = AES_128 ENCRYPTION BY SERVER CERTIFICATE MyDEKCert;
- **Set the database to use encryption**
  - ALTER DATABASE MyDatabase SET ENCRYPTION ON;

TDE Mechanism

- **Very simple:**
  - Database pages are encrypted before being written to disk
  - Page protection (e.g. checksums) applied after encryption
  - Page protection (e.g. checksums) checked before decryption
  - Database pages are decrypted when read into memory
- **When TDE is enabled, initial encryption of existing pages happens as a background process**
  - Similar mechanism for disabling TDE
  - Monitor with encryption_state column of sys.dm_database_encryption_keys
    - Encryption state 2 means the background process has not completed
    - Encryption state 3 means the database is fully encrypted

Backups and TDE

- **A backup of a TDE encrypted database is also encrypted using the database encryption key**
- **To restore the backup OR attach the database, the DEK must be available!**
  - There is no way around this – if you lose the DEK, you lose the ability to restore the backup
  - Maintain backups of server certificates too
- **After DEK is rotated twice, a log backup must be performed before rotating again**
  - Or use SIMPLE recovery model

TDE Limitations

- **If TDE is enabled, a database will not use instant file initialization**
  - Can cause significant performance drop
  - Especially important for database restore operations during disaster recovery
- **FILESTREAM data is not encrypted**

Key Takeaways

- Securing SQL Server requires a defense-in-depth approach, starting with physical security and the principal of least privilege for service configuration and account provisioning
- Auditing access in SQL Server using Server Audits or SQL Trace can meet compliance requirements and provide a means of tracking all activity against important systems
- Database(s) can be protected at rest using Transparent Data Encryption, but it is imperative to maintain a separate backup of the encryption certificate, and to understand the steps required to restore the database(s) in the event of a disaster

Summary

- Securing SQL Server
- Authentication
- Authorization
- Auditing access
- SQL injection risks
- Database encryption

Questions?

SQLskills
immerse yourself in sql server

Appendix: Principal Metadata

- **Information about principals**
- **Server**
  - sys.server_principals (includes SID)
  - sys.sql_logins
  - sys.server_role_members
- **Database**
  - sys.database_principals
  - sys.database_role_members

Appendix: Permissions Metadata

- sys.securable_classes
- sys.server_permissions
- sys.database_permissions
- sys.fn_builtin_permissions()
- fn_my_permissions()
- has_perms_by_name(…)
  - Permission checks in applications
- Tokens
  - sys.login_token
  - sys.user_token

Appendix: Audit Metadata

- Information
  - sys.server_audits
  - sys.file_audits
  - sys.server_audit_specifications
  - sys.server_audit_specification_details
  - sys.database_audit_specifications
  - sys.database_audit_specification_details
- Status
  - sys.dm_server_audit_stats
  - sys.dm_audit_actions