Paweł Rajba
pawel@ii.uni.wroc.pl
http://kursy24.eu/

Application Security

Database security

Agenda

- Authentication
- Server-Level security
- Database-Level security
- Encryption in a database
- Communication with a database
- SQL Server Audit
- Policy based management
- Other imporant topics

Authentication modes

- Windows
- Mixed mode

Server-level security

- Basic operation: create a login
  - From MGMT studio (let’s see)
  - CREATE LOGIN statement
- Logins can be:
  - Local (if mixed mode)
  - From Windows
    - For users
    - For groups (!)
- Some options
  - MUST_CHANGE, DEFAULT_DATABASE = „…”, CHECK_EXPIRATION = ON, CHECK_POLICY = ON
    - If there is no mixed mode, one can create a local login, but policies are not checked

Server-level security

- Server-Roles (actually instance level)
  - A lot of builtin roles
    - sysadmin – the highest level
    - dbcreator
    - diskadmin
    - ...
  - Basic operations:
    - CREATE SERVER ROLE [SomeRole]
      - It is possibility to create custom server roles
    - ALTER SERVER ROLE [sysadmin] ADD MEMBER [auser]
  - List permissions for role
    - sp_srvrolepermission 'securityadmin'

Server-level security

- Managing permissions
  - Open Server properties
  - Change tab to permissions
  - Let’s review what is there
- Tip: one can always click „Script” button to see what commands are behind

Database-level security

- Database users
  - Basic operation:
    - CREATE USER [TestUser] FOR LOGIN [CustomUser] WITH DEFAULT SCHEMA=[dbo]
      - After user is created there is no permission associated
  - Common way to give permission is to assign a role

Database-level security

- Database roles
  - There are 2 types:
    - Fixed (predefined)
    - Flexible (defined by user)
  - How to create a role?
    - From SQL
      - CREATE ROLE rolename
      - ALTER ROLE rolename
        ADD MEMBER {username|rolename}
    - From MGMT Studio
  - List of permissions for role:
    - sp_dbfixedrolepermission rolename
  - More operations and possibilities
    - http://technet.microsoft.com/en-us/library/ms189121(v=sql.110).aspx
  - Let's see database roles
- Application roles
  - Gives possibility to assign permission to a specific application
  - After connection, a sp_setapprole procedure is invoked
    - More on http://technet.microsoft.com/en-us/library/ms190998(v=sql.110).aspx

Database-level security

- Managing permissions
  - From MGMT Studio
    - Open Database properties
    - Change tab to permissions
    - Let's review what is there
  - From SQL
    - GRANT privilege_name_list
      [ON object_name]
      TO {user_name |PUBLIC |role_name}
      [WITH GRANT OPTION]
    - REVOKE privilege_name_list
      [ON object_name]
      FROM {user_name |PUBLIC |role_name}

Database-level security

- Schema
  - Consider as security container for different objects
  - Allows to organize objects as well
- Basic operation:
  - CREATE SCHEMA <Warehouse> [AUTHORIZATION <User>]
    - Authorization defines an owner
  - Accesing schemas: [schema].[object]
    - E.g. CREATE TABLE [Warehouse].[Invoice] (...)
  - Default schema: usually [dbo] but can be changed in user properties

Database-level security

- Ownership
  - Principles
    - Owner manages objects he/she owns and anyone can revoke him/her these privileges
    - There is no possibility to drop user if it owns something
  - Change ownership
    - ALTER AUTHORIZATION ON <Object> TO <User>
  - More
    - ALTER AUTHORIZATION
      - http://technet.microsoft.com/en-us/library/ms187359(v=sql.110).aspx
      - http://msdn.microsoft.com/en-us/library/ms187359.aspx

Database-level security

- Ownership chains
  - Let’s assume that there is a chain of calls
    $O1 \rightarrow O2 \rightarrow O3 \rightarrow \ldots \rightarrow On$
    and all Oi has the same owner
  - Then permissions are checked only on access to O1
- Let’s see the consequences

Database-level security

- Practical example: roles usage
  - There is default good way to give an EXECUTE permission to a user
  - The solution

    -
      ```
      CREATE ROLE db_executor
      GRANT EXECUTE TO db_executor
      EXEC sp_addrolemember 'db_executor', 'username'
      ```

Database-level security

- Practical example: the ownership chain consequences

```
CREATE TABLE SomeData (Number INT)
GO

CREATE PROCEDURE ShowSomeData AS SELECT * FROM SomeData
GO

--ALTER AUTHORIZATION ON SomeData TO dbo --SCHEMA OWNER
--ALTER AUTHORIZATION ON ShowSomeData TO dbo --SCHEMA OWNER
--GO

SELECT * FROM sys.all_objects WHERE name LIKE '%SomeData'
GO

GRANT EXECUTE ON ShowSomeData TO Test
DENY SELECT ON SomeData TO Test
GO

EXECUTE AS USER = 'Test'
GO
SELECT * FROM SomeData
GO
EXEC ShowSomeData
GO
REVERT
GO
```

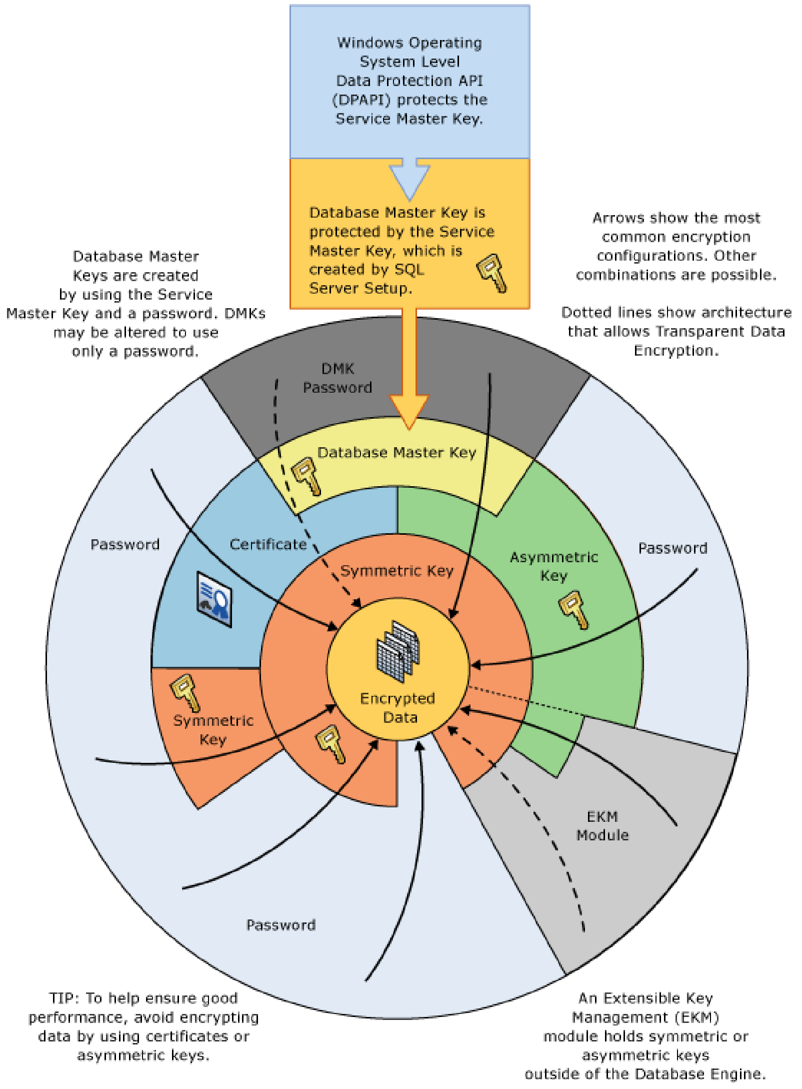
Encryption in a database

- There are situations in which protecting access to a database is not enough
  - Someone breach this access level protection
  - Access rights are assigned in a wrong way
  - Backup files are stolen
  - Protection of filesystem is compromised
  - And many others…
- If we have a very sensitive data, encryption in a database is a one more layer of defense

Encryption Hierarchy

- Encryption can be achieved through different ways
- Every way is implied by a different chain of keys
- Every way has pros and cons, so should be evaluated according to the requirements

More:
http://technet.microsoft.com/en-us/library/ms189586(v=sql.110).aspx



Windows Operating System Level Data Protection API (DPAPI) protects the Service Master Key.
Database Master Key is protected by the Service Master Key, which is created by SQL Server Setup.
Database Master Keys are created by using the Service Master Key and a password. DMKs may be altered to use only a password.
Arrows show the most common encryption configurations. Other combinations are possible.
Dotted lines show architecture that allows Transparent Data Encryption.
DMK Password
Database Master Key
Password
Certificate
Symmetric Key
Asymmetric Key
Password
Encrypted Data
Symmetric Key
EKM Module
Password
TIP: To help ensure good performance, avoid encrypting data by using certificates or asymmetric keys.
An Extensible Key Management (EKM) module holds symmetric or asymmetric keys outside of the Database Engine.

Encryption hierarchy components

- Asymmetric Keys
- Symmetric Keys
- Certificates
- Extensible Key Management (EKM)
    - Since SQL Server 2008
    - Gives a possibility to manage some cryptographic keys from hierarchy by an external source such as Hardware Security Module (HSM)

Column Encryption vs. Transparent Data Encryption

- Column Enryption: data is encrypted explicitly
  - Applications and users are impacted
  - One can choose what exactly should be encrypted – no overhead for encryption less sensitive data
- TDE: the whole database is encrypted
  - Encryption is hidden and transparent, so if one can connect, one can see the data
  - Everything is encrypted, also less sensitive data
- The choice depends on business needs

Column Encryption

- This is supported by set of built-in functions and procedures together with key hierarchy
- Operations are performed manually
- Encrypted data needs to be stored in a varbinary column type
- Main steps
  - Create database master key for every database
    - Notice: service master key has been created when the instance has been created
  - Create a certificate to protect keys
  - Create a symmetric key which is protected by the certificate created in the previous step
  - Enjoy encrypting data: open the symmetric key, encrypt the data, close the key
- Decryption is similar to encryption, but a function for decryption should be used

Example

```
USE Test

CREATE TABLE Person
(
    ID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    CreditCard VARBINARY(200)
)
GO

INSERT INTO Person (ID, FirstName, LastName) VALUES(1, 'J1', 'K1');
INSERT INTO Person (ID, FirstName, LastName) VALUES(2, 'J1', 'K1');
INSERT INTO Person (ID, FirstName, LastName) VALUES(3, 'J1', 'K1');
GO

CREATE MASTER KEY ENCRYPTION BY PASSWORD='SomePassword'
GO

CREATE CERTIFICATE CertForTest WITH SUBJECT='Test'
GO

CREATE SYMMETRIC KEY CreditCardKey WITH ALGORITHM=TRIPLE_DES ENCRYPTION BY CERTIFICATE CertForTest
GO

OPEN SYMMETRIC KEY CreditCardKey DECRYPTION BY CERTIFICATE CertForTest
UPDATE Person SET CreditCard = ENCRYPTBYKEY(KEY_GUID('CreditCardKey'), '11111') WHERE ID=1;
UPDATE Person SET CreditCard = ENCRYPTBYKEY(KEY_GUID('CreditCardKey'), '22222') WHERE ID=2;
UPDATE Person SET CreditCard = ENCRYPTBYKEY(KEY_GUID('CreditCardKey'), '33333') WHERE ID=3;
CLOSE SYMMETRIC KEY CreditCardKey
GO

SELECT * FROM Person
GO

OPEN SYMMETRIC KEY CreditCardKey DECRYPTION BY CERTIFICATE CertForTest
SELECT ID, FirstName, LastName, CONVERT(VARCHAR, DECRYPTBYKEY(CreditCard)) [Credit Card] FROM Person
CLOSE SYMMETRIC KEY CreditCardKey
GO
```

Transparent Data Encryption

- TDE is one of usages of encryption by symmetric keys
- There is whole database encrypted by a symmetric key called database encryption key
- Database encryption key is protected by certificate which is protected by database master key or asymmetric key from EKM
- Available only on Enterprise Edition or Developer Edition
- Provides query optimization
- Main steps
    - Create master key encryption password
    - Create a certificate
    - Backup the certificate
    - Create a database encryption symmetric key
    - Alter the database to set encryption on
    - Optionally monitor the encryption process
- More: http://msdn.microsoft.com/en-us/library/bb934049.aspx

Example

```sql
USE master

CREATE MASTER KEY ENCRYPTION BY PASSWORD='SomePassword'
GO

CREATE CERTIFICATE TestDatabaseServerCertificate WITH SUBJECT='Test Certificate'
GO

BACKUP CERTIFICATE TestDatabaseServerCertificate
TO FILE ='C:\Temp\TestDatabaseServerCertificate'
WITH PRIVATE KEY(
    FILE = 'C:\Temp\TestDatabaseServerCertificate.private',
    ENCRYPTION BY PASSWORD = 'AnotherPassword')

USE Test

CREATE DATABASE ENCRYPTION KEY WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER CERTIFICATE TestDatabaseServerCertificate
GO

ALTER DATABASE Test SET ENCRYPTION ON
GO

SELECT DB_NAME(database_id), encryption_state, key_algorithm, key_length
FROM sys.dm_database_encryption_keys
GO
```

Encryption algorithms

- DES
- Triple DES
- TRIPLE_DES_3KEY
- RC2
- RC4
- 128-bit RC4
- DESX
- 128-bit AES
- 192-bit AES
- 256-bit AES

- More: http://technet.microsoft.com/en-us/library/ms345262(v=sql.110).aspx

Communication with a database

- When it comes to communication we consider two challenges
  - Storing credentials to a database server in a secure way
    - This was covered in OWASP Top 10 topic
  - Encrypting communication channel
    - SQL Server supports encrypting connection using SSL
      - A valid certificate is required
    - DEMO
      - Open: Configuration Tools → SQL Server Configuration Manager
      - Open: Properties for SQL Server Network Configuration
    - More
      - http://msdn.microsoft.com/en-us/library/ms191192(v=sql.110).aspx
      - http://technet.microsoft.com/en-us/library/ms189067(v=sql.105).aspx

SQL Server Audit

- It is a mechanism which allows to monitor who is doing what on which objects
- There a lot of possibilities what can be audited
- It is based on Extended Events, new feature since SQL Server 2008
  - Audit is specialized usage of Extended Events

SQL Server Audit

- DEMO: Let's create Server-Level audit
  - MGMT → Security → Audits
    - Create an audit DatabaseRoleMemberChange
  - MGMT → Security → Server Audit Specifications
    - Create a specification DatabaseRoleMemberChange related to DatabaseRoleMemberChange event
  - Add any user to any role
    - USE Test; ALTER ROLE db_owner ADD MEMBER test
  - MGMT → Security → Audits
    - Pick the audit
    - Choose View Audits Logs option

SQL Server Audit

- DEMO: Let’s create Database-Level audit
  - MGMT → Security → Audits
    - Create TestDatabaseSelect audit
  - MGMT → Test database → Security → Server Audit Specification
    - Create TestDatabaseSelect specification on
      - SELECT event
      - Osoba table
      - [public] role
  - Perform a select on the Osoba table in Test DB
  - View TestDatabaseSelect audit

SQL Server Audit

- There is another way to see audit entries which is based on review files
- DEMO

  -
    ```
    SELECT * INTO Test.dbo.SQLAudits
    FROM sys.fn_get_audit_file(
       'C:\Temp\TestDatabase*.sqlaudit',Default, Default);
    ```

  -
    ```
    SELECT * FROM Test.dbo.SQLAudits
    ```

Policy based management

- Allows to apply and force policies and rules
- Let’s see some examples
  - MGMT → Management → Policy Management
  - Review Facets
  - Create a policy RecoveryModelFull for ensuring that every database has a full recovery model
    - Create a condition using Database Options facet
    - Create a policy based on that condition and evaluate it
  - Create a policy for ensuring that no table is created in dbo schema (do the same for procedure)
    - Create a condition using Table facet (analogously Stored Procedure)
    - Create a policy based on that condition and evaluate it
    - Try to enable that policy and try to create an object in that schema
      - E.g. `CREATE PROCEDURE dbo.GetServerName AS SELECT @@SERVERNAME`

Other important topics

- SQL Profiler and ALTER TRACE risks
- Backups and recovery
  - Backup types
  - Transaction logs and recovery model
- High Availability
  - Failover clustering
  - Database mirroring
  - Log shipping
  - Replication