

ENCRIPCION DE DATOS

Muchas veces se maneja mucha información que es sensible y que las compañías quieren proteger para evitar que caiga en manos maliciosas. Ejemplos de esta información son los números de tarjetas de crédito, números de documentos de identidad, números de teléfono, etc.

MSSQL Server, a partir de la versión 2005, nos ofrece una variedad de formas para encriptar la información en la base de datos, entre las cuales tenemos:

- Passphrase
- Certificate
- Symmetric key

Passphrase: Esta es la forma más simple y básica de encriptación de datos que se puede tener en SQL Server y es la que se utiliza por default en la plataforma de Neotel. Bajo este método, lo único que se necesita para encriptar los datos es una frase o contraseña “segura” (semilla).

Si alguien llega a saber la contraseña toda nuestra información quedaría expuesta sin necesidad de tener permisos adicionales más que leer la información de la tabla.

Certificate: SQL Server permite la encriptación de la información a través de certificados digitales los cuales pueden ser adquiridos en alguna de las entidades que los expiden. Adicionalmente SQL Server también permite la creación de certificados “self-signed” los cuales permiten al usuario crear un certificado propio con una sentencia simple de T-SQL.

Los certificados en SQL Server pueden estar encriptados por un password, o por la Database Master Key, la cual es la primera llave que se debe crear en la base de datos para que a partir de ésta se encripten los demás objetos. A través de los certificados se puede tener un poco mas de seguridad ya que para poder usar un certificado es necesario tener permiso al mismo para poder utilizarlo.

Symmetric Key: La encriptación a partir de llaves simétricas tiene como principio que para encriptar y desencriptar la información se necesita la misma llave. Es decir que si nuestra llave llega a ser pública y accesible para todos los usuarios, estos podrían ver la información sensible que tenemos encriptada en la base de datos.

Para crear una llave simétrica, esta debe ser encriptada a partir de un certificado, de una llave asimétrica o de otra llave simétrica, lo cual nos brinda mayor seguridad porque el usuario deberá pasar por encima de todos estos métodos de encriptación para poder acceder a la llave que le permitirá encriptar o desencriptar la información.

- **Encriptación Passphrase (semilla)**

Como mencionamos anteriormente, este es el método utilizado por Default en la plataforma de Neotel para la encriptación de información dentro de la BBDD.

Para hacer uso de esta funcionalidad lo primero que debemos hacer es en la pantalla de Configuración Básica definir la clave o semilla que utilizaremos para encriptar los campos de tipo Texto-Encriptado o Texto-Password-Encriptado.

Una vez definida la semilla dentro del sistema podemos proceder a dar de alta los campos en los que vamos a almacenar la información encriptada dentro de nuestra BBDD.

Con esto ya podemos trabajar sin problemas siendo que la información cargada en estos campos no sera accesible de forma directa por BBDD o bien por la plataforma sin la utilización de la semilla de encriptación.

- **Encriptación Certificate**

Para utilizar este método dentro de Neotel, podemos hacer uso de las funciones ya creadas para la encriptación por semilla, pero previo a esto deberemos realizar algunos pases adicionales.

1. Creamos la DATABASE MASTER KEY que utilizaremos para generar el CERTIFICADO

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'MiClaveSegura'
```

GO

2. Creamos el CERTIFICADO partiendo de la MASTER KEY

```
CREATE CERTIFICATE MiPrimerCertificado
WITH SUBJECT = 'DBAMemories Certificate' , EXPIRY_DATE = '10/10/2050'
GO
```

3. Generamos una copia de seguridad del CERTIFICADO ya que en caso de que este se pierda, si no contamos con la misma no podremos acceder de ninguna forma a la información almacenada en nuestra BBDD.

```
BACKUP CERTIFICATE MiPrimerCertificado TO FILE = 'D:\DBAMemories\MiPrimerCertificado.cert'
WITH PRIVATE KEY ( FILE = 'D:\DBAMemories\MiPrimerCertificado.key' ,
ENCRIPTION BY PASSWORD = 'pass' );
GO
```

4. En caso de que tengamos algún problema con el certificado en el futuro, podremos restaurar el mismo con el siguiente comando.

```
CREATE CERTIFICATE MiPrimerCertificado
FROM FILE = 'D:\DBAMemories\MiPrimerCertificado.cert'
WITH PRIVATE KEY (FILE = 'D:\DBAMemories\MiPrimerCertificado.key',
DECRYPTION BY PASSWORD = 'pass');
GO
```

5. Ahora debemos modificar dentro de la base de datos las funciones dbo.ENCRYPTAR y dbo.DESENCRIPTAR.

```
ALTER FUNCTION [dbo].[ENCRYPTAR]
( @clave VARCHAR(500), @Semilla VARCHAR(20) )
RETURNS varbinary(8000)
AS
BEGIN

    DECLARE @pass AS VarBinary(8000)

    -----
    -----
    --SET @pass = ENCRYPTBYPASSPHRASE(@Semilla,@clave)
    set @pass = ENCRYPTBYCERT(CERT_ID(@Semilla),@clave)
    -----
    -----

    RETURN @pass

END
```

```
-----

ALTER FUNCTION [dbo].[DESENCRIPTAR]
( @clave VARBINARY(8000), @Semilla varchar(20) )
RETURNS VARCHAR(50)
AS
BEGIN

    DECLARE @pass AS VARCHAR(50)

    -----
    -----
    --SET @pass = DECRYPTBYPASSPHRASE(@Semilla,@clave)
    set @pass = DECRYPTBYCERT(CERT_ID(@Semilla),@clave)
    -----
    -----

    RETURN @pass
```

6. Por ultimo en la pantalla de Configuración Básica configuraremos en el campo de la semilla el nombre del certificado y crearemos los datos como se explico anteriormente.

- **Encriptación Symmetric Key:**

Para utilizar este método dentro de Neotel, podemos hacer uso de las funciones ya creadas para la encriptación por semilla y por certificado, pero deberemos realizar algunos pasos adicionales.

1. Creamos la DATABASE MASTER KEY que utilizaremos para generar el CERTIFICADO

```
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'MiClaveSegura'  
GO
```

2. Creamos el CERTIFICADO partiendo de la MASTER KEY

```
CREATE CERTIFICATE MiPrimerCertificado WITH SUBJECT='DBAMemories Certificate',  
EXPIRY_DATE = '12/31/2012'  
GO
```

3. Creamos la llave simétrica encriptada con el certificado utilizado previamente.

```
CREATE SYMMETRIC KEY MiLlaveSimetrica  
WITH  
KEY_SOURCE = 'MyKeySource',  
IDENTITY_VALUE = 'MyIdentityValue',  
ALGORITHM = AES_256  
ENCRYPTION BY CERTIFICATE MiPrimerCertificado;  
GO
```

4. Generamos una copia de seguridad de la MASTER KEY y del CERTIFICADO ya que en caso de que este se pierda, si no contamos con la misma no podremos acceder de ninguna forma a la información almacenada en nuestra BBDD.

-- Copia de la Master Key

```
BACKUP MASTER KEY TO FILE = 'D:\DBAMemories\DMK.dat'  
ENCRYPTION BY PASSWORD = 'MiPasswordSeguro';  
GO
```

-- Copia del Certificado

```
BACKUP CERTIFICATE MiPrimerCertificado  
TO FILE = 'D:\DBAMemories\MiPrimerCertificado.cert'  
WITH PRIVATE KEY ( FILE = 'D:\DBAMemories\MiPrimerCertificado.key' ,  
ENCRYPTION BY PASSWORD = 'pass' );  
GO
```

5. En caso de que se nos presente algún problema con la clave, o el certificado, podemos hacer la restauración de los mismos con los siguientes pasos para poder acceder a la información encriptada en la BBDD

-- Restauramos la Master Key

```
RESTORE MASTER KEY  
FROM FILE = 'D:\DBAMemories\DMK.dat'  
DECRYPTION BY PASSWORD = 'MiPasswordSeguro'  
ENCRYPTION BY PASSWORD = 'MiClaveSegura'  
GO
```

-- Abrimos la Master Key para poder restaurar el Certificado

```
OPEN MASTER KEY DECRYPTION BY PASSWORD = 'MiClaveSegura'  
GO
```

-- Restauramos el Certificado

```
CREATE CERTIFICATE MiPrimerCertificado  
FROM FILE = 'D:\DBAMemories\MiPrimerCertificado.cert'  
WITH PRIVATE KEY (FILE = 'D:\DBAMemories\MiPrimerCertificado.key',  
DECRYPTION BY PASSWORD = 'pass');  
GO
```

```
-- Regeneramos la Llave Simétrica
CREATE SYMMETRIC KEY MiLlaveSimetrica
WITH
KEY_SOURCE = 'MyKeySource',
IDENTITY_VALUE = 'MyIdentityValue',
ALGORITHM = AES_256
ENCRYPTION BY CERTIFICATE MiPrimerCertificado
GO
```

6. Ahora debemos modificar dentro de la base de datos las funciones dbo.ENCRIPTAR y dbo.DESENCRIPTAR.

```
ALTER FUNCTION [dbo].[ENCRIPTAR]
( @clave VARCHAR(50), @Semilla VARCHAR(20) )
RETURNS varbinary(8000)
AS
BEGIN
    OPEN SYMMETRIC KEY @Semilla DECRYPTION BY CERTIFICATE MiPrimerCertificado;

    DECLARE @pass AS VarBinary(8000)
    -----
    -----
    SET @pass = ENCRYPTBYKEY(KEY_GUID(@Semilla),@clave)
    -----
    -----
    CLOSE SYMMETRIC KEY @Semilla

    RETURN @pass
END
```

```
ALTER FUNCTION [dbo].[DESENCRIPTAR]
( @clave VARBINARY(8000), @Semilla varchar(20) )
RETURNS VARCHAR(50)
AS
BEGIN
    OPEN SYMMETRIC KEY @Semilla DECRYPTION BY CERTIFICATE MiPrimerCertificado;

    DECLARE @pass AS VARCHAR(50)
    -----
    -----
    SET @pass = CONVERT(VARCHAR(50),DECRYPTBYKEY(@clave))
    -----
    -----
    CLOSE SYMMETRIC KEY @Semilla

    RETURN @pass
END
```

7. Por ultimo en la pantalla de Configuración Básica configuraremos en el campo de la semilla el nombre de la Clave Simétrica y crearemos los datos en el CRM como se explico anteriormente.

IMPORTANTE: Debemos tener en cuenta que todos los comandos que se ejecuten directamente en el SQL deben hacerse con el usuario “SA”, caso contrario NO FUNCIONARA la encriptacion en la plataforma.