SQL Server 安全透明数据加密

即使是一个安全设计很好的数据库，如果攻击者能够访问包含数据库的磁盘文件，就容易受到攻击。单元级加密可以保护一些数据，但对这种攻击要完全保护，就有必要加密文件，而不只是数据。这正是透明数据加密(TDE)。

透明数据加密(TDE)

在 SQL Server 2008，微软推出透明数据加密。TDE 是实时在文件级别加密和解密数据文件和日志文件。相比第八篇讲到的在列级上的数据加密，这种加密数据库中的所有数据。在第八篇你必须明确使用 T-SQL 函数加密数据，使用对称密钥，非对称密钥或证书来保护和解密数据。

TDE 在文件级加密数据库。加密是透明的，它不会影响你访问数据。不需要任何特殊的编程，有专门为管理 TDE 支持的 T-SQL。这使得 TDE 很容易设置和维护，虽然它需要更多的工作，在复制和移动数据库到 SQL Server 实例的不同位置。一旦你为数据库启用了 TDE，当数据写入到数据库时 TDE 会自动加密，当读取时会自动解密。TDE 的目的是在数据和日志文件保护数据，保护它的安全防止攻击者试图直接从文件访问数据。如果你需要通过一个过夜的包裹递送服务来运送你的数据库文件，比如 FedEx 或者 UPS，这样的场景就很有用了。没有 TDE，攻击者可以从货车后面偷你的包裹，然后在一个拥有系统管理员权限的实例上附加，并获取数据。但如果在数据库中启用了 TDE，整个数据集是安全加密的；在没有密钥的情况下，是没有办法访问数据的。其他一些 TDE 可用场景是你担心攻击者，比如内部人员，可能会以某种方式访问物理数据文件，或者你需要保护副本存档的数据库安全。

TDE 是如何工作的

TDE 需要一个证书来访问物理数据文件。没有用于加密数据库的证书，数据是不可用的。这意味着，不要把证书备份和它所保护的数据库保存在很近的地方！否则攻击者会得到所有她需要的来解密你的数据。她需要做的是在她控制的 SQL Server 实例上安装证书，并用它来附加数据库，然后就可以获取解密后的数据。需要注意的是 TDE 加密所有写入到数据库的数据，然后根据查询解密需要的数据。所以，只有当数据是静止的、存储在数据库中，它才被 TDE 保护。TDE 以每个 8K 的页加密和解密数据。

如果在一个 SQL Server 实例的任何数据库设置成 TDE 保护，SQL Server 会自动用 TDE 保护 tempdb 库。这是有道理的，因为一些受保护数据库的数据可以暂时存储在 tempdb。问题是，加密会导致性能下降。存储在 tempdb 中的所有数据都要加密，实例中的其他没有使用 TDE 保护的数据库存储到 tempdb 也需要加密。因此，这些数据库的性能可能会受到影响。

为了让一切运转，你需要有权限在 master 数据库创建 database master key 和证书，在用户数据库上有 control 权限启用 TDE。大多数时候，系统管理员会启用数据库的 TDE，所以权限需求应该不是问题。

TDE 的局限性

为了有效地使用 TDE，你应该明白 TDE 能做什么以及它不擅于做什么。考虑 TDE 的这些局限：

->你不能加密数据库中的数据的一个子集，它是全部或无

->TDE 是无法通过数据库引擎限制数据的访问。通过 TDE 数据访问是不变的。如果你有一个用户正在运行一个应用程序，并且应用程序或用户有访问数据库中的

数据所需的权限，TDE 不会以任何方式改变数据访问。它只是保护数据库文件。

->TDE 不是服务器、SQL Server 实例、或存储在数据库中的敏感数据安全的替代品。你还需要仔细考虑使用防御纵深的分层安全保护你的数据。

->TDE 最大的缺点之一是，文件流数据是不加密的。

->TDE 只在 SQL Server 企业版和开发版中能使用。

正如你可以看到，TDE 有这些“限制”，这表明它并不能作为一种安全的灵丹妙药。但如果你的数据威胁刚好对应于它所提供的保护，TDE 可以是一个重要的安全功能。

TDE 性能

TDE 一个你需要考虑的问题是性能。加密是一个昂贵的操作。TDE 的表现相当不错，因为 TDE 不加密 SQL 缓存中的数据。只有当数据被写入磁盘，SQL Server 才对它进行加密。因此，如果数据在缓存中，加密并不一定会发生，从而提高整体效率。微软已经花费了大量的时间在数据库中尽可能高效的加密，但在读或写数据的时候仍然会导致性能问题，因为加密使用复杂的算法。

使用 TDE

示例代码显示如何使用 TDE 以及对数据库的影响。脚本中的部分代码，可能需要更改文件路径以适应你本地 SQL Server 的实例，比如数据库备份位置和证书备份位置。

```
    -- Make a copy of AdventureWorks2012 database
    -- Change the path to the appropriate backup directory
    BACKUP DATABASE AdventureWorks2012
        TO DISK = N'D:\SQL2012\MSSQL11.SQL12\MSSQL\Backup\A
dventureWorks2012.bak'
```

```
       WITH NOFORMAT, INIT, NAME = N'AdventureWorks2012 Ful
l Database Backup',

       SKIP, NOREWIND, NOUNLOAD, COMPRESSION, STATS = 10;

   GO



   RESTORE DATABASE AdventureWorks2012Copy

       FROM DISK = N'D:\SQL2012\MSSQL11.SQL12\MSSQL\Backup
\AdventureWorks2012.bak'

       WITH

           FILE = 1, NOUNLOAD, REPLACE, STATS = 10,

           MOVE 'AdventureWorks2012_Data' TO N'D:\SQL2012\M
SSQL11.SQL12\MSSQL\DATA\AdventureWorks2012Copy.mdf',

           MOVE 'AdventureWorks2012_Log' TO N'D:\SQL2012\MS
SQL11.SQL12\MSSQL\DATA\AdventureWorks2012Copy.ldf';

   GO
```

示例代码会执行下面四步来启用数据库上的 TDE：

1、master 库创建一个 master key

2、master 库创建一个用 master key 保护的证书

3、创建数据库加密密钥，用证书保护

4、启用数据库 TDE

代码 9.1 首先用一个强密码创建 database master key，然后创建 AdventureWorks2012TDECert 证书。作为预防，代码备份证书。你应该把它保存到一个安全的地方，以防迁移数据库时需要：

```
   -- Create a certificate in master to use with TDE

   USE master;

   GO
```

```
    -- TDE hooks into encryption key hierarchy in SQL Serve
r

    CREATE MASTER KEY ENCRYPTION BY PASSWORD = '!drJP9QXC&V
i%cs';

    GO

    -- Create the certificate used to protect the database
encryption key

    CREATE CERTIFICATE AdventureWorks2012TDECert WITH SUBJE
CT = 'Certificate to implement TDE on AdventureWorks2012Co
py';

    GO



    -- Backup the certificate

    -- Either create the D:\SQL2012 folder or change it in t
he code below

    BACKUP CERTIFICATE AdventureWorks2012TDECert TO FILE =
'D:\SQL2012\AdventureWorks2012TDECert'

       WITH PRIVATE KEY ( FILE = 'D:\SQL2012\AdventureWorks
2012TDECertPrivateKey' ,

       ENCRYPTION BY PASSWORD = 'RISiS9Ul%CByEk6' );

    GO

    -- Must backup private key as well
```

代码 9.1 创建证书并备份证书

在深入配置 TDE 之前先运行代码 9.2。语句会列出当前实例下加密的数据库，以及加密状态。在一个全新、原始的实例上，应该不会返回任何结果集。在应用 TDE 之后，你会看到结果有所改变：

```
    SELECT DB_NAME(database_id) AS DatabaseName,

       key_algorithm AS [Algorithm],

       key_length AS KeyLength,
```

```
       CASE encryption_state
          WHEN 0 THEN 'No database encryption key present,
no encryption'
          WHEN 1 THEN 'Unencrypted'
          WHEN 2 THEN 'Encryption in progress'
          WHEN 3 THEN 'Encrypted'
          WHEN 4 THEN 'Key change in progress'
          WHEN 5 THEN 'Decryption in progress'
       END AS EncryptionStateDesc,
       percent_complete AS PercentComplete
   FROM sys.dm_database_encryption_keys;
   GO
```

代码 9.2 罗列 SQL Server 实例上加密的数据库

现在是时候启用 AdventureWorks2012Copy 的 TDE。执行代码 9.3，使用之前在 master 库创建的证书创建一个数据库加密密钥。如果你没有备份证书的话会收到一条警告。然后使用 ALTER DATABASE 带上 SET ENCRYPTION ON 子句启用 TDE，根据数据库的大小和服务器的速度，它可能需要一段时间才能完全加密数据库，可能数小时或数天。

```
   USE AdventureWorks2012Copy;
   GO
   -- Create the database encryption key for TDE. Analogou
s to database master key for data encryption.
   CREATE DATABASE ENCRYPTION KEY
       WITH ALGORITHM = TRIPLE_DES_3KEY
       ENCRYPTION BY SERVER CERTIFICATE AdventureWorks2012
TDECert;
```

```
    GO

    -- Get a warning about backing up the key, if you haven'
t already

    -- ...take the advice and back it up!


    -- Now need to turn TDE on.

    ALTER DATABASE AdventureWorks2012Copy SET ENCRYPTION O
N;

    GO
```

代码 9.3 创建数据库加密密钥并启用 TDE

当它在加密数据库的时候，重新执行代码 9.2。你会看到类似图 9.1 的结果。注意图中显示两个数据库：AdventureWorks2012Copy 数据库启用了 TDE，tempdb 库是自动应用加密。一旦初始化加密完成，PercentComplete 列会显示为 0(是的，列名有点误导人，它完成时并不是真实的完成百分比)。

结果 消息

	DatabaseName	Algorithm	KeyLength	EncryptionStateDesc	PercentComplete
1	tempdb	AES	256	Encrypted	0
2	AdventureWorks2012Copy	TRIPLE_DES_3KEY	192	Encryption in progress	74.2683

图 9.1 sys.dm_database_encryption_keys 监视加密进度

注意图中 AdventureWorks2012Copy 是由 TRIPLE_DES_3KEY 算法加密的。这是因为在代码 9.3 中我们指定的加密算法。你可以选择 SQL Server 中支持的任何加密算法。同时注意 tempdb 库使用默认的 AES 加密算法。

使用代码 9.4 测试加密。只要在开启 TDE 之前有访问数据的必要权限，你就依然能够访问数据。应用程序访问数据也是一样。

```
    SELECT TOP 500 * FROM Production.Product;
```

代码 9.4 测试开启 TDE 后访问数据

如果你想关闭数据库的 TDE，可以使用代码 9.5

```
ALTER DATABASE AdventureWorks2012Copy
    SET ENCRYPTION OFF;
```

代码 9.5 禁用数据库的 TDE 功能

测试 TDE

测试安全特性以确保其按照你期望的方式工作通常是明智的。示例代码包含部分测试 TDE 步骤，包括假设你意外地删除 TDE 中用于保护数据库加密密钥的证书。代码 9.6 先备份 AdventureWorks2012Copy 数据库，接着删除 AdventureWorks2012Copy 数据库，然后删除 AdventureWorks2012TDECert 证书，从而模拟证书丢失。(这也模拟了当你试图附加加密数据库到不同的、且没有安装原始证书的 SQL Server 实例)

```
BACKUP DATABASE AdventureWorks2012Copy
    TO DISK = N'D:\SQL2012\MSSQL11.SQL12\MSSQL\Backup\A
dventureWorks2012Copy.bak'
    WITH NOFORMAT, INIT, NAME = N'AdventureWorks2012Cop
y Full Database Backup',
    SKIP, NOREWIND, NOUNLOAD, COMPRESSION, STATS = 10;
GO
USE master
GO
DROP DATABASE AdventureWorks2012Copy;
GO
-- Oops! We lost the certificate and don't have a copy!
```

```
    -- Or, going to restore the database to another server i
nstance

    DROP CERTIFICATE AdventureWorks2012TDECert;

    GO
```

代码 9.6 备份数据库、删除数据库、删除证书

接下来，尝试用代码 9.7 还原数据库。你会看到如图 9.2 所示的错误信息。这是 TDE 的保护机制：无法还原或附加数据库到一个没有安装原始加密证书的 SQL Server 实例上。

```
    RESTORE DATABASE AdventureWorks2012Copy

        FROM DISK = N'D:\SQL2012\MSSQL11.SQL12\MSSQL\Backup
\AdventureWorks2012Copy.bak'

        WITH

            FILE = 1, NOUNLOAD, REPLACE, STATS = 10;
```

代码 9.7 尝试还原数据库

图 9.2 在没有安装原始加密证书的实例还原 TDE 数据库的出错信息

但是如果你备份了证书，那么可以用代码 9.8 来从备份文件中还原证书，使用之前在备份证书时保护证书的密码，然后尝试还原数据库。这次数据库成功被还原了。

```
    -- Restore the certificate

    CREATE CERTIFICATE AdventureWorks2012TDECert

        FROM FILE = 'D:\SQL2012\AdventureWorks2012TDECert'
```

```
       WITH PRIVATE KEY ( FILE = 'D:\SQL2012\AdventureWorks
2012TDECertPrivateKey',

       DECRYPTION BY PASSWORD = 'RISiS9Ul%CByEk6');

   -- Now try to restore the database

   RESTORE DATABASE AdventureWorks2012Copy

       FROM DISK = N'D:\SQL2012\MSSQL11.SQL12\MSSQL\Backup
\AdventureWorks2012Copy.bak'

       WITH

           FILE = 1, NOUNLOAD, REPLACE, STATS = 10;
```

代码 9.8 从备份文件中还原证书，然后还原数据库