PolotskSU

UDC 534.781

PROTECTION OF INFORMATION STORED IN DATABASE MEANS MS SQL SERVER (TRANSPARENT DATA ENCRYPTION)

ALEXANDER POPOV, IRYNA BURACHONAK
Polotsk State University, Belarus

With the development of computing tools and information transfer systems, the problem of ensuring its security, preventing unauthorized access to information, physically destroying or modifying protected information is becoming increasingly urgent.

Under the threat to the security of information means an action or event that can lead to the destruction, distortion or unauthorized use of information resources. In order to increase the security of stored data, humanity is constantly inventing various algorithms to prevent unauthorized access to information.

Moreover, the choice of methods and means of protection is determined not only by the importance of the information being processed, but also by the composition of the data storage system, its structure, methods of information processing. At present, among the developers of modern software applications, client-server architecture is popular. It allows you to distribute the load on the solution of tasks among several machines, thereby increasing the speed of data processing. In this article we will look at the option of building a system using MS SQL Server as a data warehouse and using Transparent Data Encryption (TDE).

Encryption of a database file is carried out at the page level. Pages in an encrypted database are encrypted before being written to disk and decrypted when read into memory. Transparent data encryption does not increase the size of the encrypted database. When using transparent data encryption, SQL Database automatically creates a server-level certificate stored in the master database. To move a TDE encrypted database, you need to decrypt it, move it, and then re-enable TDE in the target SQL server. This way data is protected when physical media is stolen, since without a certificate, the attacker will not be able to access the encrypted data [1].

Next we directly consider the encryption process using TDE. Special hierarchy of keys is used to increase the reliability of cryptographic protection and reduce the load on the system. Each database table is encrypted with a special symmetric Database Encryption Key. Database Encryption Key is encrypted by a certificate that is created in the database. The copy of this certificate is placed in the Master database. The master database certificate is encrypted with its master key. The master key of the master database is encrypted with the master key of the Service Master Key (SMK) service. The service master key is the top of the MS SQL Server encryption hierarchy. It is created automatically when it is needed during encryption. However, to enhance security it is also encrypted by the operating system data protection service.

As a result, we have the following chain: Service master key -> Database master key -> Keys and certificates in the database. Accordingly it is not possible to decrypt the data without having one of the components. Therefore it is necessary to make backup copies of keys [2]. The user application that interacts with the database server receives all the necessary data in decrypted form at the stage of encrypting the master key of the master with the master key of the SMK service.

This scheme uses both symmetric encryption algorithms and asymmetric ones. Symmetric encryption is less demanding on system resources, but more vulnerable. The asymmetric method, on the contrary, is protected at the key management stage. For the same reason it requires significantly more computational resources. Using a combination of both methods allows you to neutralize the shortcomings of each of them and to improve security and performance; compared to if all the time used one of the types of encryption. Thus, the database is protected by faster symmetric encryption, which is preferable when taking into account large amounts of information. In turn, base encryption keys are subjected to asymmetric encryption, the size of which is incommensurably small, but the criticality of their protection is higher. Using this approach on servers with low I / O, low CPU consumption and enough RAM to store large amounts of information affects the performance by 3-5% when you turn on TDE. Servers with a smaller amount of RAM, whose applications load the CPU and I / O system, will produce a load of 28% more, the reason being asynchronous execution of procedures in SQL.

Thus, this algorithm provides encryption of the user database for which TDE encryption was enabled, the transaction log of this database, as well as the common temporary database tempdb. But unlike full encryption of the user base, the transaction log and tempdb are encrypted from the moment TDE is enabled — the previous entries in them remain open. Therefore, after encryption is enabled, it is necessary to re-create these files.

PolotskSU

The main disadvantage of this scheme, despite the fact that it ideally opposes the theft or seizure of the database itself, is the vulnerability of data at the network level. That is, during the transfer between the client and the server, data can be stolen or replaced. Additional organization of encrypted channels using TDE would entail additional performance costs. With large amounts of data and the number of requests it could seriously affect the performance of the system as a whole. In order to encrypt all information via the client-server-client channel, it is necessary to issue the root trusted certificate to the server, import it to the client stations and configure the cryptographic algorithm interaction scheme. One of the most effective schemes will be traffic encryption using a symmetric method, while the keys are protected by open certificates.

By combining these two approaches, you can fully protect the data stored in the database.

REFERENCES

1. Microsoft. Прозрачное шифрование данных (TDE). [Электронный ресурс] / MSDN. – Режим доступа: https://msdn.microsoft.com/. – Дата доступа: 10.09.2018.
2. DBASIMPLE. Основы MS SQL Server шифрования [Электронный ресурс] / IBM. – Режим доступа: http://dbasimple.blogspot.com/2013/07/ms-sql-server.html . – Дата доступа: 10.09.2018.