Annals of DAAAM for 2010 & Proceedings of the 21st International DAAAM Symposium, Volume 21, No. 1, ISSN 1726-9679
ISBN 978-3-901509-73-5, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria, EU, 2010
Make Harmony Between Technology and Nature, and Your Mind will Fly Free as a Bird
Annals of DAAAM International

ENCRYPTION STRATEGIES IN DATABASES

BOICEA, A[lexandru]; GHITA, V[lad]; RADULESCU, F[lorin] & SARBU, A[nca] D[aniela]

Abstract: *In this paper, we approach the databases' security matter, offering two solutions for protecting sensitive data. To give sensitive data the highest level of security, it should be stored in encrypted form. The encryption's goal is to make data unintelligible to unauthorized readers and extremely difficult to decipher when attacked. Depending on chosen encryption strategy, the system's overall performance could dramatically increase, especially when working with large volumes of data. The latest trend in database security is to keep the performance overhead of data encryption process away from the database, by using dedicated encryption machines: Crypto Servers. We try to identify the benefits of this new solution compared to ordinary DBMS encryption.*

Key words: *database, encryption, security, sensitive*

1. INTRODUCTION

From source towards destination, data passes through one or more intermediaries. Usually, messages across a network may be persistent for some periods of time, from message queues to files or databases. Some data within messages may be sensitive in nature, so its disclosure can result in loss or damage, such as identity theft or criminal offenses. Encryption is used to protect sensitive data inside a message. Unencrypted data - plaintext - is converted to encrypted data - ciphertext. Data is encrypted using a cryptographic key and an encryption algorithm, making it unintelligible to parties other than the intended recipient. At its destination, ciphertext is converted back to plaintext. Encryption can be symmetric or asymmetric, but usually a mean solution is used for data encryption: a symmetric algorithm is used to encrypt the message, then asymmetrically encrypt the shared key (Microsoft, 2005).

2. PLANNING A STRATEGY FOR DATABASE ENCRYPTION

Before starting designing a secure database encryption' strategy, it is important to understand three things: how encryption works, what is the data flow in the target application and how database's encryption fits in the company's overall security policy.

The encryption process may differ between situations. The security of encrypted data depends on several factors such as the type of algorithm used in encryption, the key size and the actual implementation of the algorithm. There is a tension between security and access to data; for example, a low security results in easy access to data - even by unauthorized persons - but a high security may result in very hard access - even by the authorized persons, like company's employees.

Encryption keys' management is also an important factor of database security. (Oracle Corp.,2001).

In order to obtain a good management, the following questions should be considered:

- *Where will the keys be stored?*
- *How many keys are needed?*
- *Who will access the keys?*
- *How often will the keys be changed?*

Furthermore, database encryption may be optimized even from the beginning, depending on its way of implementation. In the following sections we present two solutions of data encryption: encrypting the data using DBMS facilities and encrypting the data outside DBMS, using dedicated cryptographic servers (RSA Security Inc, 2002).

3. SOLUTION ONE - ENCRYPTING DATA WITHIN DBMS

While encryption within DBMS has a less impact on application environment, there are some performance and security issues which must be considered. Depending on algorithms and their implementation, the encryption may degrade DBMS'performance. An inherent vulnerability of this solution is the storing location of keys used in encryption; they are often kept near the data, within a database table.

Encrypting data within DBMS is usually implemented as stored procedures: before storing the data in database, an encryption procedure is called; for decrypting the data, the reverse procedure is used. The cryptographic procedures exist in DBMS as implemented by its vendors. In Tab.1 are presented the *pros* and *cons* for this encryption strategy.

Pros	Cons
Applications are unaffected by encryption system Encryption is already integrated in many DBMSs	Extra processing = performance degradation Encryption keys are stored near the encrypted data Additional hardware is required to separate the keys, which - purchased separately If keys protection is password-based, it is insecure and difficult to manage There is a limited choice of supported algorithms

Tab.1. Pros and Cons for DBMS encryption strategy

4. SOLUTION TWO - ENCRYPTING DATA OUTSIDE DBMS

Using this strategy in a client-server architecture, the sensitive data will travel unencrypted within network in the shortest possible time: encryption is performed by source application - which introduce them in the system, then data travels encrypted and will be stored in database in an encrypted form. A reliable way to use this solution is by creating a cryptographic server (further called *Crypto Server* or simply *CS*) which will centralize the entire encryption process for database environment. The administration and control tasks become easier for complex applications with many databases. A great advantage of this solution is the key management strategy, which is one of the safest. The strategy implies separating the encryption keys by the encrypted data in database: encrypted data are located at database server, while encryption keys never leave the crypto server (RSA Security Inc., 2002). All in all, this strategy provides higher security, better performance, and also a lower cost of implementation - you can maximize the investment by using one secure encryption solution across multiple applications and lock down access to encryption keys. In Tab. 2 are presented pros and cons for this encryption solution (Suse, 2000).

The cryptographic server is a specialized system used to centralize the encryption process in complex database systems. Its task is to dispose the database server with the overhead and performance degradation involved in encryption process. The Crypto Server is actually a new layer standing between application server and database server. Data comes from application server and passes encrypted to the database server. Also, there is a secured channel - usually SSL - between application server and cryptographic server. At the client side (i.e. application server), different levels of transparency could be implemented, depending on the way the SQL query is built. For example, the below queries are equal, but the second one forces CS to search for the field in the internal repository and check its encryption status, algorithm and keys used:

1. *select decrypt(pwd, 'DES', ABCDEF1234567890) from accounts;*
2. *select pwd from accounts;*

Pros	Cons
Database server is unloaded with encryption processing	Network overhead
Encryption keys are separated by data	Needs to administrate more than one server
Easy to implement strong authentication methods	Applications must be modified in order to support the crypto server
Can separate roles for administrators	Must implement solutions for securing and monitoring the crypto server
Much more control on users accessing the data	
Scalable - can work with many applications and databases	
If DBMS doesn't include encryption facilities, there is no need to be replaced	

Tab. 2. Pros and Cons for Crypto Server encryption strategy

The main modules of Crypto Server are presented in Fig.1.

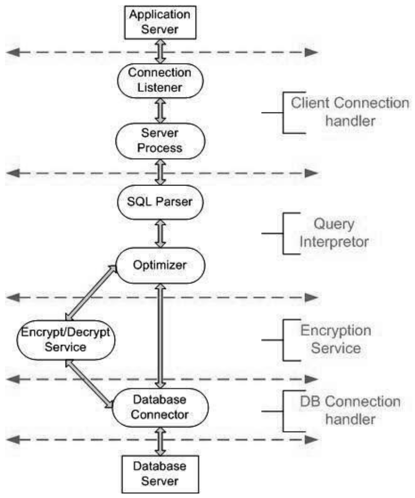

Fig. 1. Block diagram of the cryptographic server

- *Client Connection Handler* - this module manages the connection with the application server; The Connection Listener waits for connection requests and passes them to a Server Process
- *Query Interpreter* - is the "brain" of the system, being able to make decisions on encrypted elements, algorithms and keys used, based on analyzed SQL statements. The SQL Parser builds the tokens tree and passes the query to the Optimizer. The last one finds out if an element is encrypted or not, which algorithm and keys were used for encryption (based on internal repository) and which way the query will next follow.
- *Repository* – data and metadata storage location; is an internal database of encrypted elements - fields, tables.
- *Encryption Service* - If an element is believed to be encrypted, it will pass through an encryption/decryption process before being sent to database server.
- *Database Connector* - handles connections to DB server.

5. CONCLUSIONS

Database security is a delicate subject but also a very important one. Databases are a favorite target for attackers, just because of the data they are containing. Therefore the strategy of security must cover the entire system, starting with the physical layer to data layer - network, server, application, data (Burtescu, 2009). The best solution in database security is to keep performance degradation and overhead caused by encryption service away from database, by using specialized encryption machines, like cryptographic servers.

6. REFERENCES

Burtescu, Emil (2009) *Database security: attacks and control methods*, JAQM Vol. 4 No. 4, Romania

Microsoft Corp. (2005) http://msdn.microsoft.com/en-us/library/aa480570.aspx, *Accessed on 2010-05-10*

Oracle Corp. (2001) *Database Encryption in Oracle9i*, An Oracle Technical White Paper, USA

RSA Security Inc. (2002) *Securing Data at Rest: Developing a Database Encryption Strategy*, DDES WP 0702, Ireland

Suse. (2000) *Symmetric vs. Asymmetric Algorithms* http://www.suse.de/~garloff/Writings/mutt_gpg/node3.html *Accessed on 2010-05-12*