



ENCRYPTING DATABASES:

WHY AND HOW?

Copyright MIEL e-Security Pvt Ltd

WHITEPAPER

Contents

1] Overview	Pg - 2
2] First Step	Pg - 2
3] A Common Strategy	Pg - 3
4] Best Practice	Pg - 4
5] Security from Internal Breaches	Pg - 5
6] Performance	Pg - 5
7] Encrypting only the Credit Card Number	Pg - 6
8] Key Management for Database Encryption	Pg - 6
9] Solutions	Pg - 7

ENCRYPTING DATABASES: WHY AND HOW?

Copyright MIEL e-Security Pvt Ltd

Overview:

Encrypting data-at-rest in databases has always been overlooked by even the most security conscious administrators and IT managers. While their efforts have been focused on security data-in-transit and controlling access, many feel that encrypting (and subsequently) decrypting data sets in databases would lead to significant loss of performance and is, therefore, not a viable proposition.

However, encryption serves as an important component of any 'Defense-in-depth' strategy and is meant to complement existing security layers such as access control. In the absence of encryption, an attacker who is able to breach the access controls would be able to compromise the integrity of the database.

The scope of this discussion is to build a case for encrypting data-at-rest and discuss an approach towards encryption without affecting performance.

First Step:

Undoubtedly, a first step towards securing databases is to ensure that only authorized users gain access. Within a database, access control means creating users and granting them the privilege to act on objects together with performing certain commands and tasks. Using the built-in controls and mechanisms within the database are the best way of implementing access control.

Once access controls are in place, encrypting databases is the second line of defense. Even if access controls are circumvented, encryption provides an additional barrier and should limit the damage to critical data.

Two things to note about encryption are:

- *Encryption does not protect data from being deleted*
- *Encryption does not protect data from being modified, although it does provide you a way to tell if an unauthorized change has been made*

There are several possible strategies to encrypt “data-at-rest,” and each strategy has certain advantages and disadvantages. The rest of this discussion will outline the strategies and propose the best solution.

A Common Strategy:

Encryption of data-at-rest can be performed in several ways. One way is to encrypt the actual database files at the operating system level. An example of using this strategy is to encrypt an entire database file using Microsoft’s EFS within a Windows environment.

There are many weaknesses to using this strategy:

- You cannot selectively encrypt individual pieces of data. This approach results in encrypting the entire file. This causes serious performance problems for reading from the database.
- Encrypting the entire file not only adds the overhead of reading all data, but also leads to other additional overhead when recording pointers, indexes, and other internal data structures that must be encrypted and decrypted for any operation against the database.

- Another weakness is that different pieces of data cannot be encrypted with different keys, necessary when different user groups (such as HR, Sales) access information from the same database.
- File-based encryption only protects the data from operating system-level attacks. It does not protect the data from a user who breaches the database.

Best Practice:

A more efficient and effective way to encrypt information in a database is to perform the encryption on a column and row basis. To further explain this concept, think of a table containing a list of customers.

Within this customer table, there is the following information:

- Customer ID
- Customer name
- Customer address
- Customer credit card number

In this table there is little reason to encrypt the customer ID. It is most likely that you would only want the credit card information encrypted. You gain several advantages by only encrypting your most sensitive data, which in this case is credit card data. One advantage is that you can minimize the performance hit incurred by only encrypting sensitive information. For instance, when a user attempts to search the table for a specific user, they incur very minimal overhead because only the data that must be decrypted is the data found row – a small subset of the data. Even better is the fact that when you select from other tables, which do not require encryption, there is absolutely no additional overhead added.

Security from Internal Breaches:

One of the serious problems encryption solves is protecting data from being read by administrators. This is accomplished by encrypting data utilizing a secret not known by the database administrator. The most important part of this statement is that the encryption must be dependent on restricting the administrator from discovering this secret, and utilizing it to decrypt the information within the database. For instance, if the administrator can simply reset the password of an account, logon to the account, and access the data, encryption has failed to protect the data from administrators. Encryption should be based on a secret such as a password.

Of course this means that when the user needs to change his or her password, this also involves resetting the decryption keys. Investigating this statement a little closer, the encryption system that we are referring to here would utilize a single key to encrypt and decrypt data in each column. A copy of this key, called the column key, is then stored encrypted with the user's password.

Using this technique, encryption is truly dependent on a secret, providing you a way to store data within your database that even an administrator cannot view.

Performance:

One of the most important decisions is deciding which data to encrypt. Database lookups are designed to be very efficient. Unlike typical file systems, databases are expected to look through millions of rows searching for specific items in seconds. This need for fast access and retrieval places additional hardships on encrypting databases. A database cannot afford to encrypt and decrypt each piece of data it must

search. Therefore, it is critical to properly plan encryption based on how an application will use the database.

For example, let us imagine that we have a table with five columns and one million rows. The table contains customer information with the following columns: Customer ID, Customer Name, Customer Address, Sales Region, and Credit Card Number.

Encrypting only the Credit Card Number:

If I encrypted only the credit card numbers, I would substantially reduce the chance that any query might significantly slow down the database. The only significant performance hit would be if the query processor decided that searching on the Credit Card Number column would be the most efficient search. This would only occur if you were to search for a specific credit card number without providing any other search criteria.

Before actually deciding which columns to encrypt, you should first gather a list of the most common statements executed against the database. Most large applications are highly dependent on a handful of queries. Analyzing the use and frequency of SQL statements allows you to make an informed decision on how encrypting a column will affect performance.

Key Management for Database Encryption:

One of the main problems with encryption is key management. Very often we see encryption implemented by hard-coding a password into a procedure or script. Even if you use the most robust encryption algorithms and the strongest keys, this implementation of database encryption is inherently flawed.

The idea behind “defense in-depth” should not rely on one layer of security to protect another layer. In this case, once access control is circumvented, encryption is also immediately circumvented. This does little to improve the security posture of your database. Encryption should be layered on top of access control and should protect the data when access controls are circumvented.

Solutions:

Deciding to build your own security system is not a task most people should endeavor to undertake.

There are a myriad of details to deal with, such as padding or dealing with NULL values that result in complications, and leaves you open to attack if implemented incorrectly. Your best bet is to find a system that provides all the features you need. While encryption will require some additional administrative work, you should find a solution that minimizes this impact.

When evaluating the possible solution, it is most important to understand how the system works and ascertaining whether the solution provides “true encryption,” or whether it breaks down when access controls break down.

Talk to the vendor to understand the underlying architecture. If the vendor is not open with the underlying architecture, chances are they do not want to know that something is wrong. An encryption system should not attempt to hide the implementation details by concealing how the code works.



Reach us at:

MIEL e-Security Pvt. Ltd.: C – 611 / 612, Floral Deck Plaza
MIDC Central Road, Andheri (East), Mumbai 400 093. INDIA
Tel # General: +91 22 28215050 | Tel # Training: +91 22 39560003 / 04 | Fax#: +91 22 28215838
Email – General: feelsecure@mielesecurity.com | Email – Training: isti@mielesecurity.com
Website: www.mielesecurity.com
