Securing Database by Using with Masking Method

Taufik Hidayat[1]
Herjuno Pramudito[2]
Lisda Fitriana Masitoh[3]

[1]Universitas Wiralodra, Indonesia
[2]Universitas Dharma AUB Surakarta, Indonesia
[3]Universitas Pamulang, Indonesia

Abstract. The development of information technology is growing rapidly, there are so many methods that can be used to collect data in this era. Data has now become one of the most valuable assets. Therefore, we must keep our data as secure as possible. Microsoft SQL Server is a data management application that can use one of the security methods, namely masking. This is done by changing the number of elements (letters, numbers, etc.) and censoring by changing each element of our data identity to X. in this journal, I use that method because it is a method that is no less secure than data encryption and decryption. And also has the advantage of being more efficient for coders. The data used in this study is based on the sample data of the authors and lecturer who guide this journal. This method aims to keep our data safe from the malicious intent of unauthorized 3rd parties.

Key words: masking data, security database, SQL server.

Introduction

Along with advances in the field of information technology, data became an invaluable asset (Gogolin, 2010: 3-8), (Naufal, 2020), (Ukkas et al., 2017: 20-26). Numerous methods are developed in terms of data security, and masking is one of them to transform data (Permana et al., 2020: 1-6), (Ali & Ouda, 2017), (Goyal, 2015: 221-229), as an attempt to hide identity data become censored letters, this aims to prevent it from changing unknown or prevent it from unauthorized use (Ali & Ouda, 2017), (Goyal, 2015: 221-229), (Safa, 2014: 53-58).

This implementation is carried out to secure the identity of authorized user data with the masking method, which is expected to minimize data identity theft in an easy way but with very high quality results. Data masking or data obfuscation is the process of hiding original data with random characters or data (Ali & Ouda, 2017), (Goyal, 2015: 221-229), (Safa, 2014: 53-58).

Microsoft SQL Server is a desktop database server application that client/server has a client component, which functions to display and manipulate data; as well as server components that function to store, call, and secure the database (Grasdal et al., 2003: 53). Management operations of all server's databases in the network are carried out by database administrators using the main administrative tools SQL The server named Enterprise Manager. This results in the database administrator being only able to perform these operations on a computer that has Microsoft SQL Server installed (Husni et al., 2005: 40-45).

Literature Review

Securing SQL Server can be viewed as a series of steps, involving four areas: the platform, authentication, objects (including data), and applications that access the system. The following topics will guide you through creating and implementing an effective security plan.

All data masking platforms replace data elements with similar values, optionally moving masked data to a new location. Masking creates a proxy data substitute which retains part of the value of the original. The point is to provide data that looks and acts like the original data, but which lacks sensitivity and doesn't pose a risk of exposure, enabling use of reduced security controls for masked data repositories. This in turn reduces the scope and complexity of IT security efforts. Masking must work with common data repositories, such as files and databases, without breaking the repository. The mask should make it impossible or impractical to reverse engineer masked values back to the original data without special additional information, such as a shared secret or encryption key (Goyal, 2015: 221-229).

Research Method

In Microsoft SQL Server 2016, we come across a new solution 'Dynamic Data Masking' to prevent unauthorized users to access the defined sensitive information. We need to define curtails rules to mask the data. When any user request for the data, SQL Server checks his access and if he is not having necessary permission, he gets masked data. In this process, there are no changes in the source data.

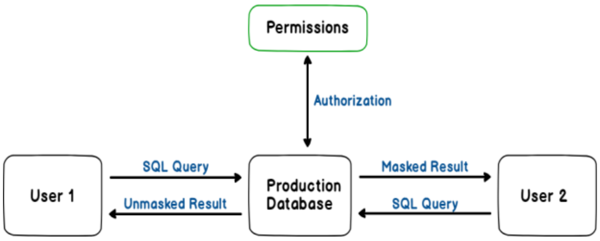


Fig. 1. SQL Data Masking Processing

Masking data, besides being the generic term for the process of data anonymization, means replacing certain fields with a mask character (such as an X). This effectively disguises the data content while preserving the same formatting on front end screen and reports. For example, a column of credit card numbers might look like:

2401 1525 1010 5469
3368 7895 6982 1478
1110 6412 4587 5896

and after the masking operation the information would appear as:

2401 XXXX XXXX 5469
3368 XXXX XXXX 1478
1110 XXXX XXXX 5896

The masking characters effectively remove much of the sensitive content from the record while still preserving the look and feel. It would not be hard to regenerate the original credit card number from a masking operation such as: 2401 1525 1010 54XX since the numbers are generated with a specific and well known checksum algorithm. Also care must be taken not to mask out potentially required information. A masking operation such as XXXX XXXX XXXX 5379 would strip the card issuer details from the credit card number. This may not be desirable (Goyal, 2015: 221-229).

And this is the example how I do it:

1. First step is install Microsoft SQL Server in your PC.
2. Create a new database.
3. Create a new query.
4. Type these codes:

```
CREATE TABLE Membership
(MemberID int IDENTITY PRIMARY KEY,
FirstName varchar(100) MASKED WITH (FUNCTION =
'partial(1, "XXXXXXX", 0)') NULL, LastName varchar(100) NOT
NULL, Phone varchar(12) MASKED WITH (FUNCTION =
'default()') NULL, Email varchar(100) MASKED WITH (FUNCTION = 'email()') NULL);
SELECT * FROM MEMBERSHIP
INSERT Membership (FirstName, LastName, Phone, Email) VALUES
('Yoga', 'Ridwan', '081224199850', 'myoga484880@gmail.com'),
('Taufik', 'Hidayat', '081234567899', 'admaction01@gmail.com');
SELECT * FROM Membership;
CREATE USER TestUser WITHOUT LOGIN;
GRANT SELECT ON Membership TO TestUser;
EXECUTE AS USER = 'TestUser';
SELECT * FROM Membership;
REVERT;
```

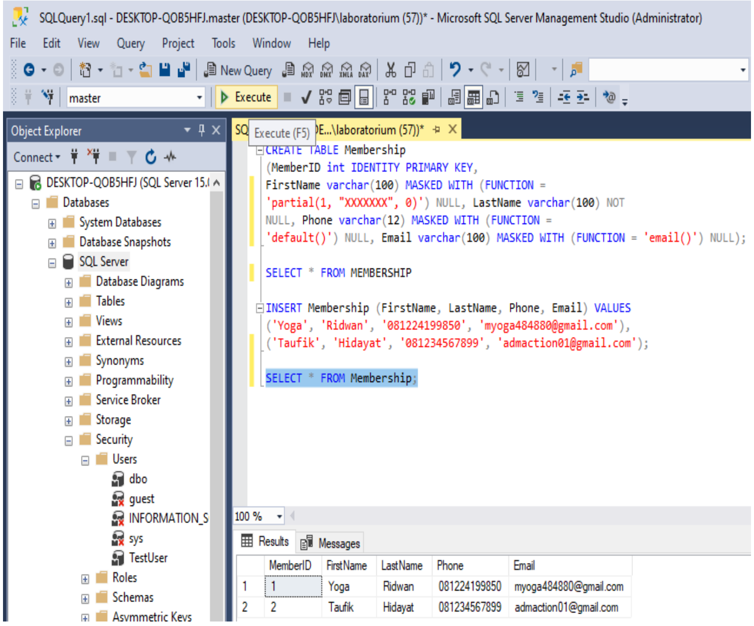
And done.

Results

Research Results

I present the implementation of data masking. The Steps of implementation are as follows:

1. Production Database
2. Unmasked Database
3. Masked Database



```
CREATE TABLE Membership
(MemberID int IDENTITY PRIMARY KEY,
FirstName varchar(100) MASKED WITH (FUNCTION =
'partial(1, "XXXXXXX", 0)') NULL, LastName varchar(100) NOT
NULL, Phone varchar(12) MASKED WITH (FUNCTION =
'default()') NULL, Email varchar(100) MASKED WITH (FUNCTION = 'email()') NULL);

SELECT * FROM MEMBERSHIP

INSERT Membership (FirstName, LastName, Phone, Email) VALUES
('Yoga', 'Ridwan', '081224199850', 'myoga484880@gmail.com'),
('Taufik', 'Hidayat', '081234567899', 'admaction01@gmail.com');

SELECT * FROM Membership;
```

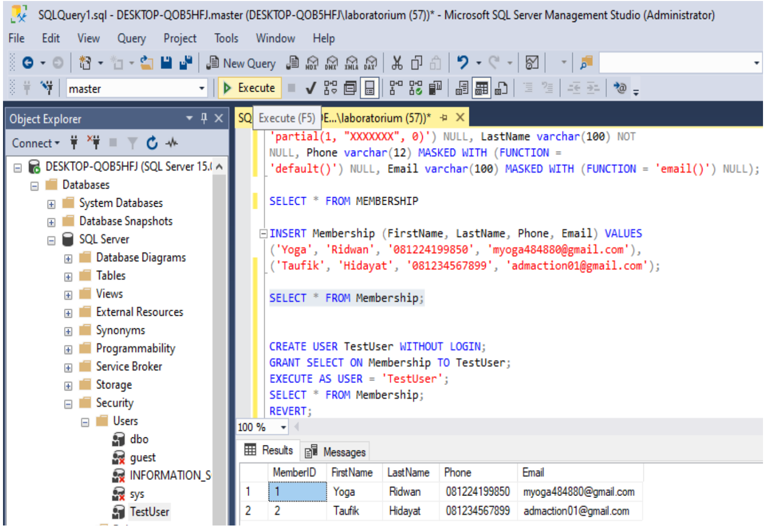
	MemberID	FirstName	LastName	Phone	Email
1	1	Yoga	Ridwan	081224199850	myoga484880@gmail.com
2	2	Taufik	Hidayat	081234567899	admaction01@gmail.com

Fig. 2. Production Database

Production database is the pure data set that has not been shared with authorized users.

This is done by typing those codes above, if it has been typed, block the SELECT * FROM Membership; section then press f5 to execute so that the data display appears as in the image above.

After that the data will be share to authorized users that i named it 'Membership', this step is called Unmasked Database.



```
'partial(1, "XXXXXXX", 0)') NULL, LastName varchar(100) NOT
NULL, Phone varchar(12) MASKED WITH (FUNCTION =
'default()') NULL, Email varchar(100) MASKED WITH (FUNCTION = 'email()') NULL);

SELECT * FROM MEMBERSHIP

INSERT Membership (FirstName, LastName, Phone, Email) VALUES
('Yoga', 'Ridwan', '081224199850', 'myoga484880@gmail.com'),
('Taufik', 'Hidayat', '081234567899', 'admaction01@gmail.com');

SELECT * FROM Membership;


CREATE USER TestUser WITHOUT LOGIN;
GRANT SELECT ON Membership TO TestUser;
EXECUTE AS USER = 'TestUser';
SELECT * FROM Membership;
REVERT;
```

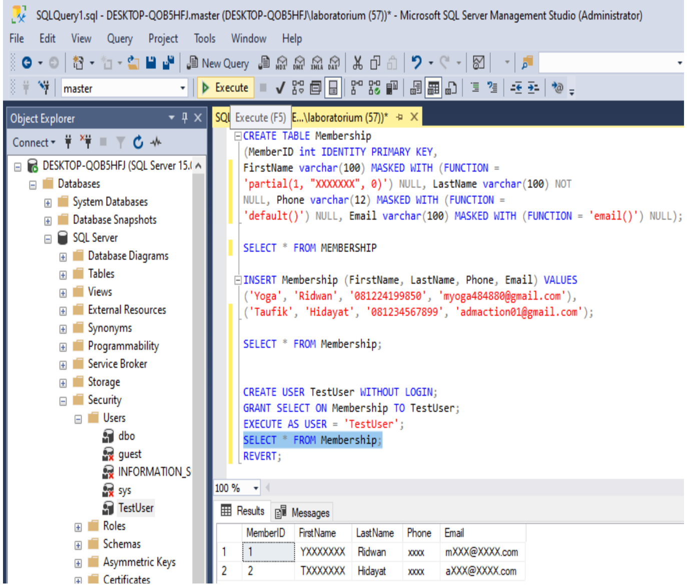
	MemberID	FirstName	LastName	Phone	Email
1	1	Yoga	Ridwan	081224199850	myoga484880@gmail.com
2	2	Taufik	Hidayat	081234567899	admaction01@gmail.com

Fig. 3. Unmasked Database

Unmasked database is the data set that has been shared privately to authorized users, this data is not intended for general consumption.

This method is exactly the same as the previous method, This is done by typing those code above, if it has been typed, block the SELECT * FROM Membership; section then press f5 to execute so that the data display appears as in the image above.

But if the data that will be consumed in general is called Masked Data.



```
CREATE TABLE Membership
(MemberID int IDENTITY PRIMARY KEY,
FirstName varchar(100) MASKED WITH (FUNCTION =
'partial(1, "XXXXXXX", 0)') NULL, LastName varchar(100) NOT
NULL, Phone varchar(12) MASKED WITH (FUNCTION =
'default()') NULL, Email varchar(100) MASKED WITH (FUNCTION = 'email()') NULL);

SELECT * FROM MEMBERSHIP

INSERT Membership (FirstName, LastName, Phone, Email) VALUES
('Yoga', 'Ridwan', '081224199850', 'myoga484880@gmail.com'),
('Taufik', 'Hidayat', '081234567899', 'admaction01@gmail.com');

SELECT * FROM Membership;


CREATE USER TestUser WITHOUT LOGIN;
GRANT SELECT ON Membership TO TestUser;
EXECUTE AS USER = 'TestUser';
SELECT * FROM Membership;
REVERT;
```

	MemberID	FirstName	LastName	Phone	Email
1	1	YXXXXXXX	Ridwan	xxxx	mXXX@XXXX.com
2	2	TXXXXXXX	Hidayat	xxxx	aXXX@XXXX.com

Fig. 4. Masked Database

In this implementation i do the data masking by changing the data set into predefined digits then convert it to censored letters, in this implementation the letters are changed to “XXXXXX”, and leaving the original name on the last name.

This is done by add new code like this:

```
CREATE USER TestUser WITHOUT LOGIN;
GRANT SELECT ON Membership TO TestUser;
EXECUTE AS USER = 'TestUser';
```

Then block the SELECT * FROM Membership; section which is under those code and then press f5 to execute so that the data display appears as in the image above.

I do this because if the authorized user wants to do something example top up, then the filler can confirm the user by mentioning only his last name.

Discussion

Of all the existing journal references, there are very few references about masking data, no one has discussed data masking on Microsoft SQL Server before, which means there are still a lot of things that can still be discussed about this.

Why Mask Data? Because:

Legal Requirements:

The regulatory environment surrounding the duties and obligations of a data holder to protect the information they maintain are becoming increasingly rigorous in just about every legal jurisdiction. It is a pretty safe assumption that the standards for the security and maintenance of data will become increasingly strict in the future.

Loss of Confidence and Public Relations Disasters

It can reasonably be said in most locations, that if a data escape happens at your organization, then the formal legal sanctions applied by governmental bodies is not the only problem you will be facing. Possibly it may not even be the biggest of your immediate worries. Inappropriate data exposure, whether accidental or malicious, can have devastating consequences. For example, what will it cost the organization if potential customers are not willing to provide sensitive information to your company because they read an article about a data escape in the newspaper. Dealing with the public relations aftermath of seeing the company's name in the press will not be cheap. It also does not take much imagination to realize that senior management are not going to be happy about having to give a press conference to re-assure the public. The public relations costs of a data escape usually far exceed the sanctions levied by governmental organizations.

Accidental Exposure

The risk of accidental exposure of information is often neglected when considering the security risks associated with real test data. Often it is thought that “there is no point in masking the test data because everybody has access to production anyways”. Not so, the risks associated with an accidental exposure of the data remain. Often just masking the most sensitive information (credit card numbers, customer email addresses etc) is enough to somewhat mitigate the damage associated with accidental exposure and the masked databases remain just as functional (Goyal, 2015: 221-229).

Conclusion

However, if you need to use production data in a test environment or real-time applications, where the content of the data can be precisely redacted to reduce its sensitivity to data privacy risk exposure, then use data masking. Not only can data masking be more secure than data encryption when using persistent data masking approaches, users may also find it to be a more efficient process.

References

Ali, O., & Ouda, A. (2017). A Content-Based Data Masking Technique for A Built-In Framework in Business Intelligence Platform. IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE) (pp. 1-4). Windsor, ON, Canada: IEEE. https://doi.org/10.1109/CCECE.2017.7946661

Gogolin, G. (2010). The Digital Crime Tsunami. Digital Investigation, 7(1-2), 3-8. https://doi.org/10.1016/j.diin.2010.07.001

Goyal, C. (2015). Data Masking: Need, Techniques & Solutions. International Research Journal of Management Science & Technology (IRJMST), 6(5), 221-229. https://doi.org/10.32804/IRJMST

Grasdal, M., Hunter, L. E., & Cross, M. (2003). Chapter 2 - MCSE 70-293: Planning Server Roles and Server Security. In W. Schmied, R. Shimonski, D. J. Shinder, & T. W. Shinder (Eds.), MCSE 70-293 Training Guide: Planning and Maintaining a Windows Server 2003 Network Infrastructure (pp. 53-146). Que Publishing. https://doi.org/10.1016/B978-193183693-7/50006-3

Husni, M., Jatmiko, N. P., & Prasetyo, A. (2005). Web Based SQL Server Database Management Software Design. Jurnal Ilmiah Teknologi Informasi, 4(1), 40-45. Available at: http://juti.if.its.ac.id/index.php/juti/article/viewFile/244/193

Naufal, R. A. (2020). PT Tokopedia's Responsibility in Cases of Leaking User's Personal Data. Yogyakarta: Universitas Islam Indonesia. Available at: https://dspace.uii.ac.id/123456789/26797

Permana, I. S., Hidayat, T., & Mahardiko, R. (2020). Raw Data Security By Using Elgamal and Sha 256 Public Key Algorithm. TEKNOKOM : Jurnal Teknologi dan Rekayasa Sistem Komputer, 4(1), 1-6. https://doi.org/10.31943/teknokom.v4i1.53

Safa, E. (2014, September). Masking-Filtering Method Analysis in Text Data Insertion. Majalah Ilmiah: Informasi dan Teknologi Ilmiah (INTI), IV(1), 53-58. Available at: https://nanopdf.com/download/analisis-metode-masking-filtering-dalam-penyisipan-data-teks_pdf

Ukkas, M. I., Andrea, R., & Anggen, A. P. (2017). Data Security Techniques With End Of File (EOF) Steganography and Vernam Cipher Cryptography. Sebatik, 17(1), 20-26. Available at: https://jurnal.wicida.ac.id/index.php/sebatik/article/view/82