BLUECORE RESEARCH

Database Security – Masking

Static and Dynamic Data Masking

Whitepaper

www.bluecoreresearch.com
info@bluecoreresearch.com

FEBRUARY 22, 2022

Introduction

Data Masking has been a hot topic in recent years, but many don't fully understand it. This paper is about static and dynamic data masking, the benefits and limitation, alternatives, and what to expect from the technologies.

Static Data Masking

Static data masking is a way to secure non-production databases. When talking about non-production databases, we refer to databases like test and development that contain sensitive production data. Many customers use production data in non-production systems to improve software quality and shorten development cycles.

Using production data earlier in the development cycles helps improve software quality by performing more tests with the production data. It also helps detect flaws earlier in the development cycles. That means quicker repairs, thereby reducing the total time to deployment.

Threat vectors

Storing sensitive data outside the protected production environment increases the exposure of the data significantly. The increased exposure is in three areas:

- QA & Development personnel often have unrestricted access to the non-production databases that contain sensitive information. By connecting to the database and running queries, they can extract any sensitive information.
- QA & Development personnel have unrestricted access to the application running on the non-production databases. Using the application, they can access any sensitive data.
- Non-production systems, both database and application servers, are not well secured. That increases the likelihood of exposure to unauthorized personnel that gained access to these systems.

Threats

The threat landscape of non-production systems is massive. From internal abuse of privilege to external hacks - non-production systems are poorly secured.

Non-production systems threats include:

- A large number of individuals with unrestricted access.
- Individuals without security clearance to sensitive data.
- Poorly managed authorization controls giving access to many unknown individuals.
- Poorly managed access control, patches, vulnerabilities, etc., giving potential access to any intruder at the operating system, application, and database levels.
- Poorly managed physical security giving potential access to the hardware and storage that contain the data. Sometimes, even portable hardware is used, such as laptops and external storage.
- There are no significant security measures like firewalls, auditing, etc.

Objectives

Static data masking aims to remove sensitive information from non-production systems. That is the simplest and cheapest way to eliminate the risk.

Static data masking replaces sensitive data with harmless fake data. The idea is that if non-production systems don't contain sensitive data, there is no risk.

However, the masked must be usable for testing purposes and allow testing quality equal to or better than what is achievable with the sensitive data.

For the fake masked data to be considered good, it must satisfy three basic conditions:

- The masking process removed the sensitive information. There should be no way to discover the original data from the masked data.
- The masked data must be valid and consistent. Fields that have special rules or checksums must maintain those. Masked primary keys have to be unique. Masked foreign keys (or equivalent references) must match. City, state, and zip code must be consistent, etc.
- The masked data must retain or improve test quality. Among other things, it means that the masked data should "feel" real. For example, it should retain data frequency like the ratio between male and female. It should continue to use the same patterns. If there were valid and invalid values, they should continue to exist. And more.

Challenges

Static data masking should enable you to create "good fakes". Giving you the tools that can analyze and manipulate the data to create a realistic mask. That includes, for example:

- Analyze column patterns to ensure the masked data contains all the patterns that exist in the original data.
- Analyze value distribution to maintain, for example, the male to female ratio.
- Ensure generated primary keys are unique.
- Maintain correct mapping between primary keys and referencing columns such as foreign keys.

Strategies

There are four primary strategies for static data masking:

- Value manipulation - this strategy changes each field independently. The masked data is a function applied to the original data. Depending on the masking function, the masked data can retain some aspects of the original data. This simple form of masking is often effective, though it has a higher potential of exposing sensitive information. In some cases, for example, when applied to names, it will produce data that is not realistic.
- Data generation - this strategy creates new data unrelated to the original sensitive information. While the masked information won't disclose anything about the original data, it is also completely unrelated. Therefore, if the masked columns are part of the test, testing quality will be poor.

- Automatic profiling - this strategy includes three steps: analyzing the original data, creating a data profile, and generating data based on that profile. This strategy is a type of data generation based on the original data. It provides high-quality realistic data that does not expose the original data and even breaks row association.
- Custom profiles - this is an advanced form of automatic profiling that allows you to customize the data profile. It lets you create data that fit particular scenarios, resembles past data profiles, or is similar to other datasets. It is also an effective means of creating datasets of any size - both bigger and smaller.

Since each approach has different benefits and drawbacks, a good data masking solution will offer all these strategies allowing you to use the most appropriate method for each situation.

Implementing strategies in different types of data requires different algorithms. For example, value manipulation on textual data requires character substitution algorithms, while manipulating numerical quantities is done with noise infusion - adding random noise to each number.

Another example is profiling: limited data sets such as gender or country require a different algorithm than patterns such as phone numbers or credit cards.

Special types of data can also require special algorithms. For example, date or time in database format, parsing a date or a time from text fields, LOBs, and more.

Finally, it's important to note the quality of the algorithms. For example, a popular algorithm for masking primary and foreign keys is predictive masking by fixing the seed of the random numbers. However, in real-world scenarios, that method breaks. That's why Core Audit uses dictionaries to ensure consistency. See a separate whitepaper on consistency methods.

Dynamic Data Masking

Dynamic data masking is for securing production databases. Don't be confused by the similar name - dynamic data masking addresses entirely different threats with vastly different objectives and solutions.

Unlike the simple solution of static data masking for non-production, you cannot remove sensitive data from production. Therefore, you must secure the data in production.

Threats

The application and other database users must use production data, including sensitive information, for the business to function. That is unlike non-production, where the mere existence of sensitive data poses an unnecessary risk.

Dynamic data masking could, in theory, be used to mask data in the application. However, that is only possible in reality when the database is aware of the application's end-user. For example, when the application doesn't use an application server or the end-user authenticates directly against the database.

In most environments, the database cannot distinguish between application SQL that requires mask data and one that requires unmasked data. Therefore, masking data in the application is almost always done by the application.

As a result, dynamic masking is only for individuals with direct SQL access to the database. These include DBAs, analysts, developers with production access, and more.

However, in most cases, these users should not have access to sensitive data. Analysts, developers, and other non-privileged personnel can have only limited permissions preventing them from accessing sensitive columns. You can achieve that using standard database permissions.

You can block DBAs and privileged users from accessing sensitive data using solutions like Core Audit. That is not part of built-in database security but does not require dynamic masking.

Therefore, dynamic masking is only for analysts, developers, or DBAs who need to see a masked version of the data and not be blocked altogether.

Objectives

Unlike static data masking, dynamic masking doesn't aim to create realistic data. It only aims to prevent certain accounts from viewing sensitive information.

Due to its real-time nature, dynamic masking can only perform value manipulation by applying a function to sensitive columns. Dynamic masking doesn't have the luxury of time and resources to analyze the entire column data and produce good fakes. Usually, that is sufficient for hiding parts of the data from the relevant users.

However, dynamic masking is useless for testing purposes since the masked data is not realistic, will not allow for a high-quality test, and in some cases produce invalid data that will not allow the test to run at all.

As mentioned in the alternatives below, dynamic masking is particularly useful when users need to regularly view part of the sensitive information but not all of it. There are simpler and more effective alternatives for most other situations.

Alternatives

Production databases protection includes a vast array of detective and preventive measures. While this paper will not go into great detail, here are some examples:

- Database permissions - all databases include table and column level permissions able to restrict access for non-privileged accounts. Only administrators can bypass these restrictions, and they are simple and effective in blocking all non-privileged accounts.
- SQL blocking - when administrators should not view sensitive information, blocking the activity is a much better approach than masking part of it.
- Separation of duties - if there's a need for temporary access to sensitive information, using an Amnesty policy is the best approach.
- Auditing reports - these can show, among other things, who accessed sensitive information. That is often an effective

strategy for deterring users and administrators from abusing their privileges.

- Anomaly analysis & alerts - by analyzing database activity profiles, it is possible to detect, for example, unusual access to sensitive information.
- Proactive forensic - giving security personnel visibility into who does what in the database opens a myriad of investigation and analyses options.

As mentioned above, there are, often, easier and more effective means of protecting production data than using dynamic masking. A good database security solution will offer all these options giving you the flexibility to use the right strategy for each particular situation.

Technologies & Challenges

Dynamic masking targets individuals with direct SQL access. Because these individuals are well-versed in databases and SQL, a dynamic masking solution shouldn't have limitations that these individuals can easily bypass.

The core technology in dynamic data is SQL rewrite. For example, by rewriting this SQL:

```
SELECT phone FROM employees;
```

Into this version that returns a masked phone number:

```
SELECT mask_phone(phone) FROM employees;
```

Depending on the core technology of the solution, different solutions implement the rewrite in different locations:

- Inside the database. Many databases include native capabilities for dynamic masking like Oracle Data Reduction and SQL Server Dynamic Data Masking.
- On the network before the query enters the database.
- Inside the database engine using a solution like Core Audit.

Database Native Dynamic Masking

Native dynamic masking is the most "full proof" as it's built by the database vendor directly into the database's SQL parser. However, it presents with two inherent flaws:

- Masking functions - native dynamic masking usually offer a limited set of masking functions. Those may not satisfy your requirements.
- DBAs - native dynamic masking is always administered by DBAs. As such, it is ineffective in controlling DBA activity - one of the reasons for using it in the first place.

Network-based Dynamic Masking

Network-based dynamic masking has significant limitations inherent in the technology - what the tool can see on the network.

The most crippling limitation is that these tools cannot see encrypted network activity. That is a significant problem because individuals with direct SQL access can always connect with encryption.

Another limitation is that these tools can only see the messages sent to the database and not the internal activity in the database.

For example:

```
declare
@s1 nvarchar(100)='pme morf enohp tceles',
@s2 nvarchar(100)='';
while len(@s1)>0
begin
  set @s2 = substring(@s1,1,1) + @s2;
  set @s1 = substring(@s1,2,len(@s1)-1);
end
exec sp_executesql @s2
```

This SQL Server block reverses the string and executes it. When that block runs, it will execute this SQL:

```
select phone from emp
```

By looking at the network traffic alone, it's impossible to know that the block will read phone numbers from the `emp` table, let alone rewrite that block to return masked data.

That is one of an infinite number of examples for the limitations in network-based database security in general and dynamic masking in particular.

Dynamic Masking in the Database Engine

The Core Audit Full Capture technology works directly with the database engine and can rewrite the SQL during the database's SQL parse.

Network encryption or dynamic SQLs, like the examples above, cannot bypass this technology. This technology is also impervious to DBAs and privileged users, unlike the native database capabilities.

While this option is the most air-tight dynamic masking solution, it is only one of a myriad of capabilities offered by Core Audit that are often more appropriate than dynamic masking.

Final thoughts

While static data masking is a fundamental measure for protecting non-production databases, dynamic masking is one of the least significant capabilities for protecting production systems.

The confusing name and significant marketing efforts brought dynamic masking into the forefront of database security. However, there are far more critical security measures to apply to production databases.