CYBERTEC PGEE

Data Masking and Obfuscation

Date: 2024-09-16
Publisher: CYBERTEC PGEE team

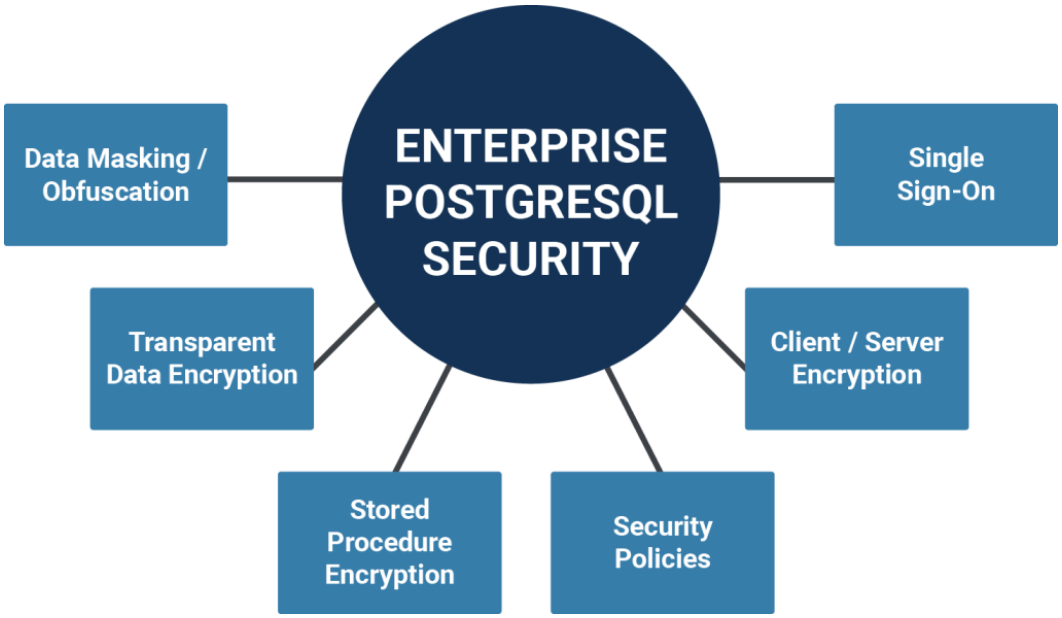
TABLE OF CONTENTS

PGEE: Comprehensive database security

CYBERTEC PostgreSQL Enterprise Edition (PGEE) is a CYBERTEC product which has been designed for enterprise-grade security in critical environments that require additional **security** as well as regular auditing. This solution focuses heavily on **compliance** and **business critical** workloads for various industries, including but not limited to:

- Banking and financial services
- Governments and defense
- Critical national infrastructure
- Business-critical missions

Ensuring security is key and therefore our first priority is to provide customers with **encryption at every level** while delivering cutting edge performance.

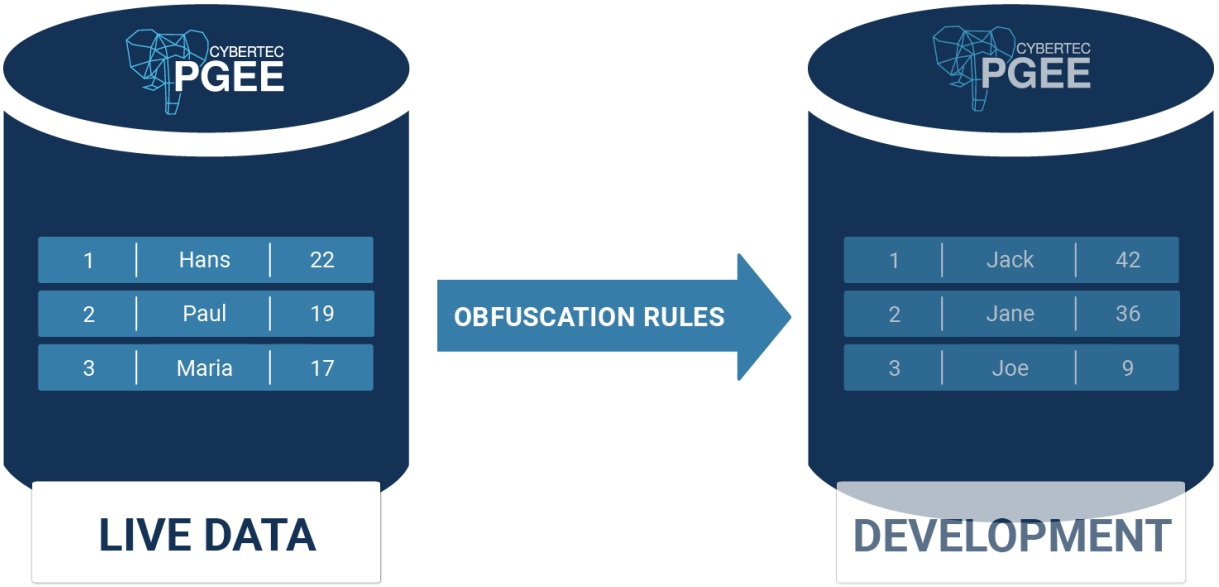


PGEE offers comprehensive database security and provides the necessary tooling to enable enterprise success, focusing on these key aspects:

- Encryption at every level
- Secure software development
- Auditing and certification

Why obfuscation and security matter

Data obfuscation in a relational database makes sense as an **added layer of security** to protect sensitive information. When sensitive data such as personally identifiable information (PII), financial data, or confidential business information is stored in a database, it's crucial to ensure that this data remains private and secure.



Obfuscating this data by **substituting it with artificially generated values** or scrambling the original data can significantly reduce the risk of unauthorized access, theft, or leakage. By making it difficult for attackers to decipher the original data even if they gain access to the database, obfuscation adds an **additional hurdle** for malicious actors to overcome, increasing the overall security posture of the system.

Additionally, obfuscation can also help to **mask patterns** and relationships within the data, further limiting the potential damage in case of a breach.

This document will discuss the following topics:

- How to make use of data masking in PGEE
- Exporting data
- Adding custom modules

Using obfuscation and masking

In PGEE obfuscation is part of the core product. The idea is to apply a function to a column of your choice and allow administrators to export this data to other hosts.

The following process shows the overall workflow:

- Define an obfuscation model in your database
- Export the database in a secure way
- Import the dump on the target system.

Defining an obfuscation model in real life means that only a handful of columns might be secret as shown in the next example:

ID	Name	Credit Card *
23	Hans	4548 3432 9874 2342
24	Paul	6890 3091 4789 0981
25	Jane	5567 4352 1099 4353

PGEE strictly separates the core database from the obfuscated version to reduce any risk of data breaches and leaks.

“Compliance and regulations matter”

Safely hand over data to your development teams and make sure all security regulations are met without exceptions.

“Be ready for the next audit”

In the next section we will dive into the practical aspects of data masking provided by PGEE.

Step 1: Creating a simple data model

The following code snippet contains a simple table which contains important information which needs to be protected:

```
pgee=# CREATE TABLE t_product (
           id               serial,
           product_number   text,
           price            numeric,
           units_available  int
);
CREATE TABLE
```

Let us add some sample data to the table:

```
pgee=# INSERT INTO t_product
           (product_number, price, units_available)
     VALUES     ('SA456BJ3', 459.95, 343),
                ('KLM009', 98.25, 29),
                ('BKL1902', 582.78, 56.34);
INSERT 0 3
pgee=# SELECT * FROM t_product;
 id | product_number | price  | units_available
----+----------------+--------+-----------------
  1 | SA456BJ3       | 459.95 |             343
  2 | KLM009         |  98.25 |              29
  3 | BKL1902        | 582.78 |              56
(3 rows)
```

We will use this trivial example through the entire document to demonstrate what can be done.

Step 2: Deploying a simple obfuscation model

The way PGEE works is that you can apply a simple function to columns which should not be visible to the receiving development team. To configure obfuscation we can simply call `ALTER TABLE … ALTER COLUMN … SET MASK` and apply an expression:

```
pgee=# ALTER TABLE t_product
           ALTER COLUMN product_number
           SET MASK md5(product_number);
ALTER TABLE
```

In this case we apply the md5 function to the column. Note that this does NOT mean that we see different data — data will be extracted later. All we did is to assign rules to the column:

```
pgee=# SELECT * FROM t_product;
 id | product_number | price  | units_available
----+----------------+--------+-----------------
  1 | SA456BJ3       | 459.95 |             343
  2 | KLM009         |  98.25 |              29
  3 | BKL1902        | 582.78 |              56
(3 rows)
```

The crucial piece of information here is that the expression has to be IMMUTABLE. It is not allowed to use a VOLATILE expression which is not guaranteed to return the same data every time.

The following example shows exactly what happens:

```
pgee=# ALTER TABLE t_product
           ALTER COLUMN units_available
           SET MASK units_available*random();
ERROR:  invalid MASK expression
DETAIL:  User-defined or built-in mutable functions
         are not allowed.
```

Using a deterministic expression will fix the problem.

```
pgee=# ALTER TABLE t_product
           ALTER COLUMN units_available
           SET MASK units_available*pi();
ALTER TABLE
```

Step 3: Extract obfuscated data

Now that we have defined the obfuscation model we can extract the data. The way to do that is to use pg_dump with a PGEE-specific command line flag:

```
postgres@hans:~$ pg_dump --masked pgee
--
-- PostgreSQL database dump
--

...

CREATE TABLE public.t_product (
    id integer NOT NULL,
    product_number text,
    price numeric,
    units_available integer
);

…

--
-- Data for Name: t_product; Type: TABLE DATA; Schema:
public; Owner: postgres
--

COPY public.t_product (id, product_number, price, units_available)
FROM stdin;
1    ddc90a55917a4e8f79aada69b8e3ebf8      459.95
1077.56621812991
2    f926a788bda72d04591753a2176d1d72      98.25    91.1061869104
3    143ef26ffce1d497fc3991a7ec418d51      582.78
175.929860102841
\.

…

--
-- PostgreSQL database dump complete
--
```

Extracting the obfuscated data can be facilitated using the “--masked” command line option. You can take this text representation and load it into your target database.

However, **caution** is advised:

- Keep in mind that obfuscation can break constraints
  - Make sure your obfuscation functions comply with your constraints
  - Keep in mind that unique indexes must stay unique
- Be careful with numbers
  - Your financial reports might return inconsistent data
  - Numbers might not add up

Optional: Enabling logical decoding

In PGEE logical decoding is often used to stream data between hosts. By decoding the WAL it is possible to easily dispatch data inside large scale organizations. Of course we are also able to stream obfuscated data. These two commands are the key to success: `CREATE PUBLICATION` and `CREATE SUBSCRIPTION`:

```
pgee=# \h CREATE PUBLICATION
Command:     CREATE PUBLICATION
Description: define a new publication
Syntax:
CREATE PUBLICATION name
    [ FOR ALL TABLES
      | FOR publication_object [, ... ] ]
    [ WITH ( publication_parameter [= value] [, ... ] ) ]

where publication_object is one of:

    TABLE [ ONLY ] table_name [ * ] [ ( column_name [, ... ]
) ] [ WHERE ( expression ) ] [, ... ]
    TABLES IN SCHEMA { schema_name | CURRENT_SCHEMA } [, ...
]
```

Basically a table can be streamed in two ways:

- Plain data
- Obfuscated data

If you want to send the obfuscated version of data you have to add a parameter to the WITH-clause as shown below:

```
CREATE PUBLICATION mypublication
      FOR TABLE t_product
      WITH (mask=on);
```

In this case PGEE will send the data to the subscriber in the desired format thus ensuring that only safe data can ever reach the destination.

Support and getting help

Requesting help

Thank you for using CYBERTEC PGEE and **thank you for being our customer.**
Your feedback is important to us and we are looking forward to hearing from you.
If you are facing any issues or technical questions please reach out to our technical team and make use of our 24x7 support and ticketing system.

CYBERTEC Support Portal

Our consultants are eager to help you with any technical and business related issues.