1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт кибербезопасности и защиты информации

**ЛАБОРАТОРНАЯ РАБОТА № 2**

**Шифрование данных**

по дисциплине «Основы построения защищенных баз данных»

1. Выполнила
2. студентка гр. 4851003/90801 Кулеева А.Г.

1. Руководитель
2. ассистент Зубков Е.А.
3. Санкт-Петербург
4. 2023

# **Цель**

Получить навыки обеспечения конфиденциальность баз данных методом шифрования.

# Ход работы

На базе данных, использованной в прошлых лабораторных работах, было проведено исследование различных режимов шифрования, использующихся в рамках защиты СУБД.

Для исследования методов шифрования были выбраны следующие данные используемой базы:

Отношение document: должно быть зашифровано так как содержит информацию о сотрудниках;

Отношение change\_log: должны быть зашифрованы атрибуты operation и fio, так как они содержат чувствительную для организации информацию.

Модуль pgcrypto обычно поставляется вместе с остальными вспомогательными утилитами. Чтобы подключить ее, необходимо выполнить

CREATE EXTENSION IF NOT EXISTS pgcrypto;

## Шифрование нескольких столбцов

В рамках данного этапа исследования были зашифрованы атрибуты таблицы document, указанные ранее.

Для шифрования была использована функция PGP\_SYM\_ENCRYPT. Код ее использования представлен ниже, а полученные результаты на Рисунке 1.

ALTER TABLE change\_log ALTER COLUMN fio TYPE bytea

USING PGP\_SYM\_ENCRYPT(fio::text, 'mysecretkey');

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1 — Зашифрованные столбцы

Для получения доступа к данным необходимо воспользоваться функцией PGP\_SYM\_DECRYPT. Для получения информации необходимо знать ключ шифрования. Использованный код представлен ниже, а полученные результаты на Рисунке 2.

Select PGP\_SYM\_DECRYPT(fio::bytea, 'mysecretkey')::text as text from change\_log;

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 2 — Расшифрованные столбцы incidents

## Шифрование всей таблицы

PostgreSQL не поддерживает функционала по шифрованию отношения полностью, однако этого можно добиться путем шифрования всех атрибутов таблицы. Для автоматизации данного процесса был использован механизм триггеров. Разработанный код представлен ниже.

CREATE OR REPLACE FUNCTION encrypt\_document()

RETURNS TRIGGER AS $$

BEGIN

new.type\_doc := pgp\_sym\_encrypt(new.type\_doc::text, 'mysecretkey');

new.fio\_respons := pgp\_sym\_encrypt(new.fio\_respons::text, 'mysecretkey');

new.department := pgp\_sym\_encrypt(new.department::text, 'mysecretkey');

return new;

END;

$$ language plpgsql;

CREATE OR REPLACE TRIGGER my\_trigger

BEFORE INSERT ON document

FOR EACH ROW

EXECUTE PROCEDURE encrypt\_document();

Как видно по представленному коду, суть триггера состоит в шифровании всех столбцов таблицы после вставки данных в нее. Результат операции INSERT в таблицу с триггером приведен на Рисунке 3.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 3 — Зашифрованная запись

Для получения доступа к зашифрованным данным необходимо воспользоваться методом, описанным в пункте 2.1.

## Шифрование данных при передаче

Далее были исследованы возможности PostgreSQL по организации шифрования передаваемых по сети данных. Для начала было установлено, что передаваемые данные не шифруются с помощью утилиты WireShark. Передаваемые данные представлены ниже на Рисунке 4.

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Автоматически созданное описание Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 4 — Результат запроса

Далее для настройки шифрования передаваемых данных были изменены параметры SSL в файле postgresql.conf, ключи и сертификаты были созданы с помощью утилиты OpenSSL. Использованные настройки представлены ниже:

# - SSL -

ssl = on

ssl\_ca\_file = 'C:/Program Files/PostgreSQL/15/data/server.crt'

ssl\_cert\_file = 'C:/Program Files/PostgreSQL/15/data/client.crt'

#ssl\_crl\_file = ''

#ssl\_crl\_dir = ''

ssl\_key\_file = 'C:/Program Files/PostgreSQL/15/data/client.key'

ssl\_ciphers = 'HIGH:MEDIUM:+3DES:!aNULL' # allowed SSL ciphers

ssl\_prefer\_server\_ciphers = on

ssl\_ecdh\_curve = 'prime256v1'

ssl\_min\_protocol\_version = 'TLSv1.3'

#ssl\_max\_protocol\_version = ''

#ssl\_dh\_params\_file = ''

#ssl\_passphrase\_command = ''

#ssl\_passphrase\_command\_supports\_reload = off

Для подтверждения шифрования соединения после перезагрузки системы можно подключиться к серверу с помощью psql. Уведомление об установке SSL-соединения представлено ниже на Рисунке 5.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 5 — Соединение SSL установлено

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 6 — Результат запроса был зашифрован

## Результаты

В Таблице 2 представлены результаты исследований. По ним видно, что шифрование хранимых в памяти столбцов очень сильно нагружает операции вставки и выборки. Это связано с тем, что перед проведением операции для каждой записи происходил вызов функции Шифрования/Расшифрования. Также видно, что несмотря на шифрование траффика, среднее время операции не изменяется при SSL соединении.

Таблица 2 — Результаты исследований

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Режим шифрования | Объем данных | | Среднее время выполнения операции | | |
| До | После | Тип | До | После |
| Несколько столбцов | 1224 Kb | 2960 Kb | SELECT | 362 ms | 723 ms |
| Вся таблица | 32 Kb | 48Kb | INSERT | 71ms | 169ms |
| Данные при передаче | 1224 Kb | 1224 Kb | SELECT | 523ms | 590ms |
| INSERT | 815ms | 893ms |

# Вывод

В ходе выполнения данной лабораторной работы были изучены возможности PostgreSQL по реализации шифрования данных. Были рассмотрены и проанализированы различные режимы шифрования, также были получены навыки построения SSL соединения с сервером базы данных.

ПРИЛОЖЕНИЕ 1

CREATE DATABASE l2;

\c l2

CREATE TABLE document (

num\_doc BIGSERIAL NOT NULL PRIMARY KEY,

type\_doc VARCHAR(300) NOT NULL,

fio\_respons VARCHAR(300) NOT NULL,

department VARCHAR(300) NOT NULL,

date\_start DATE NOT NULL,

date\_end DATE,

secret INT,

content json);

CREATE TABLE change\_log (

num\_doc BIGSERIAL NOT NULL UNIQUE REFERENCES document (num\_doc),

date\_change DATE NOT NULL,

fio VARCHAR(100) NOT NULL,

operation VARCHAR(100),

PRIMARY KEY (num\_doc, date\_change, fio));

//////////////////////////////////////////////////////

INSERT INTO document (

type\_doc, fio\_respons, department, date\_start, date\_end, secret)

VALUES (‘docx’, ‘Ivanov’, ‘IT’, ‘2022-01-01’, ‘2025-01-01’, 2);

INSERT INTO document (

type\_doc, fio\_respons, department, date\_start, date\_end, secret)

VALUES (‘pdf’, ‘Smirnov’, ‘HR’, ‘2020-03-01’, ‘2024-01-01’, 1);

INSERT INTO document (

type\_doc, fio\_respons, department, date\_start, date\_end, secret)

VALUES (‘txt’, ‘Kuleeva’, ‘IC’, ‘2022-11-11’, ‘2022-02-02’, 2);

INSERT INTO document (

type\_doc, fio\_respons, department, date\_start, date\_end, secret)

VALUES (‘xml’, ‘test’, ‘admin’, ‘2022-12-12’, ‘2022-03-03’, 2);

INSERT INTO document (

type\_doc, fio\_respons, department, date\_start, date\_end, secret)

VALUES (‘docx’, ‘Petrov’, ‘IT’, ‘2020-01-01’, ‘2026-01-01’, 2);

INSERT INTO document (

type\_doc, fio\_respons, department, date\_start, date\_end, secret)

VALUES (‘pdf’, ‘Sidorov’, ‘HR’, ‘2020-05-01’, ‘2025-01-01’, 1);

-----------------------------

INSERT INTO change\_log (

date\_change, fio, operation)

VALUES (‘2022-05-01’, ‘Kuleeva’, ‘u’);

INSERT INTO change\_log (

date\_change, fio, operation)

VALUES (‘2021-05-01’, ‘Smirnov’, ‘r’);

INSERT INTO change\_log (

date\_change, fio, operation)

VALUES (‘2022-05-05’, ‘Ivanov’, ‘d’);

INSERT INTO change\_log (

date\_change, fio, operation)

VALUES (‘2022-06-06’, ‘test’, ‘d’);

///////////////////////////////////////////////////////