МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
 «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)

Факультет «Информатика и вычислительная техника»

Кафедра «Кибербезопасность информационных систем»

Лабораторная работа №9

по дисциплине «Основы построения защищённых баз данных»

Выполнил: обучающийся гр. ВКБ43

Ковалев Данил Петрович

Проверил:

Скляров Алексей Викторович

Тема: комплексная настройка шифрования и защиты соединений PostgreSQL.

Цель: изучить шифрование данных на уровне столбцов с использованием pgcrypto, а также настройку SSL для защиты соединений и передачу зашифрованных данных через защищённые соединения.

Вариант 6. Индивидуальное задание: Настройте SSL-соединение с PostgreSQL, используя самоподписанные сертификаты. Создайте таблицу secure\_messages с полем message, которое должно быть зашифровано при вставке и расшифровано при выборке.

Задание 1. Установить pgcrypto и настроить SSL. Убедитесь, что расширение pgcrypto установлено в PostgreSQL. Настройте SSL-сертификаты для PostgreSQL.

Для взаимодействия с базой данных использовалась система контейнеризации Docker. Конфигурация работы PostgreSQL для работы с Docker представлена на рисунке 1. Настройка pgcrypto будет показана в 2 задании.



Рисунок 1 – конфигурация Docker compose для работы с PostgreSQL

Теперь приступим к настройке SSL, в самом начале нам нужно создать сертификаты. В данном случае я использовал код, который представлен на рисунке 2.

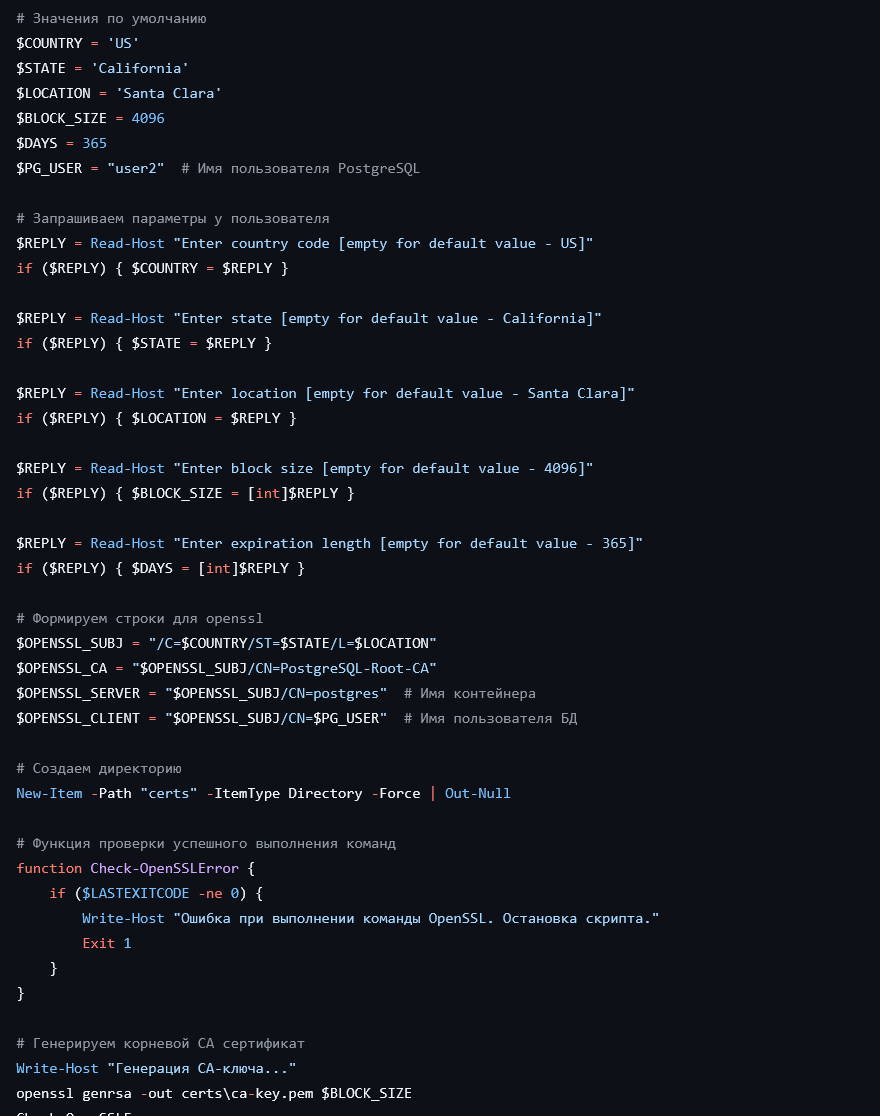


Рисунок 2 – скрипт по созданию SSL сертификатов

После запуска данного скрипта появляются файлы с расширением pem, которымы мы будем пользоваться для создания сертификатов. Данные файлы представлены на рисунке 3.

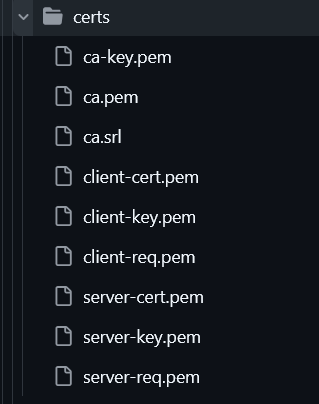


Рисунок 3 – сертификаты pem для подписи

После создания сертификатов был создан Dockerfile, который используется в docker compose. В нем была описана логика передачи сертификатов с настройкой доступов к определенным файлам, как рекомендовалось в теории к лабораторной работе. Код можете увидеть на рисунке 4.

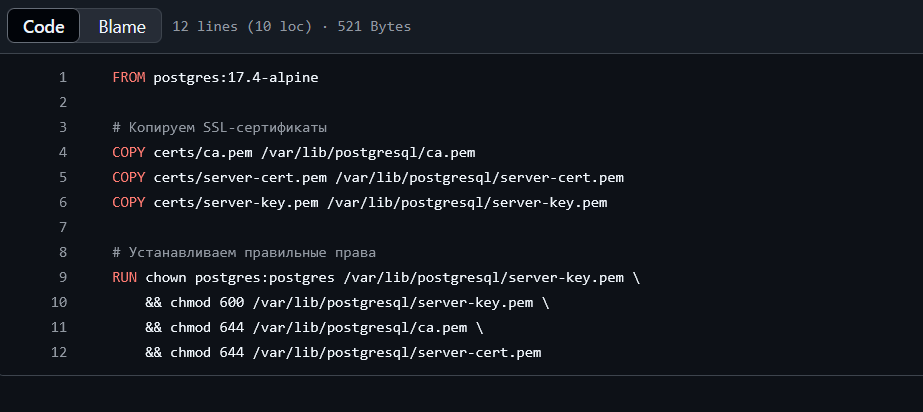


Рисунок 4 – Dockerfile

Задание 1.2. Включите SSL в файле конфигурации postgresql.conf, указав следующие параметры: ssl = on; ssl\_cert\_file = 'server.crt'; ssl\_key\_file = 'server.key'; ssl\_ca\_file = 'root.crt'

В моем случае все сертификаты были сгенерированы были с расширением pem, но это не имеет роли, потому что PostgreSQL умеет и с ними взаимодействовать. Укажем в postgresql.conf где находятся наши файлы для конфигурации SSL подключения. Конфигурация для данного варианта приведена ниже на рисунке 5.

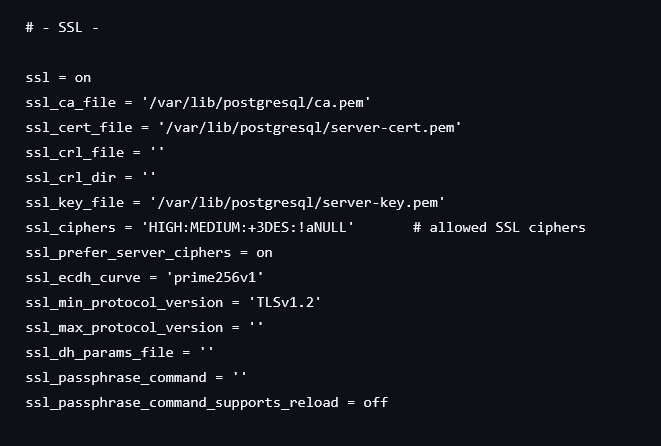


Рисунок 5 – настройки для SSL

Задание 2. Создайте таблицу согласно вашему варианту, добавив соответствующие столбцы и обеспечив зашифрованное хранение данных в указанных полях (например, с использованием pgp\_sym\_encrypt).

На рисунке 6 представлен код, который включает расширение pg\_crypto в PostgreSQL. Также был создан пользователь, с помощью которого будем тестировать SSL подключение.

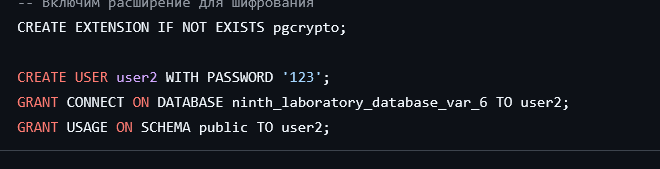


Рисунок 6 – включение расширения и настройка пользователя user2

Создадим таблицу по нашему варианту задания. Также выдадим доступы для пользователя user2, как показано на рисунке 7.

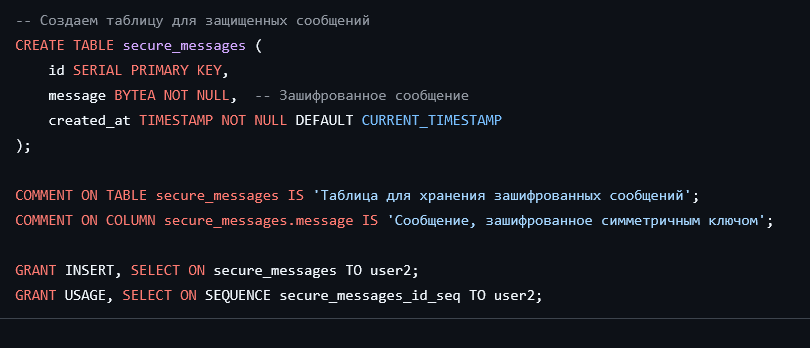


Рисунок 7 – создание таблицы и выдача прав для user2

Создадим функции для шифрования и дешифрования для удобства взаимодействия с системой. Результат представлен на рисунке 8.

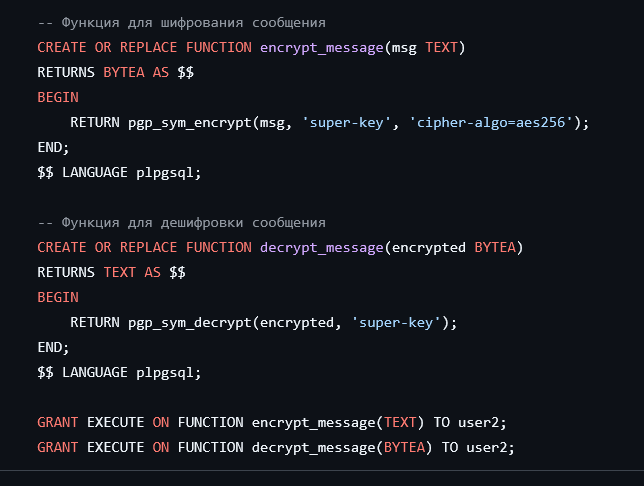


Рисунок 8 – функции для шифрования и дешифрования

Задание 3. Вставляйте данные в зашифрованные столбцы, используя функции pgcrypto (например, pgp\_sym\_encrypt для симметричного шифрования).

В моем случае я буду использовать для вставки просто свои функции, которые являются минимальной оберткой для pgcrypto.



Рисунок 9 – вставка данных в базу данных

Просмотрим теперь какие данные хранятся в таблице, если не дешифровать данные. Результат представлен на рисунке 10.

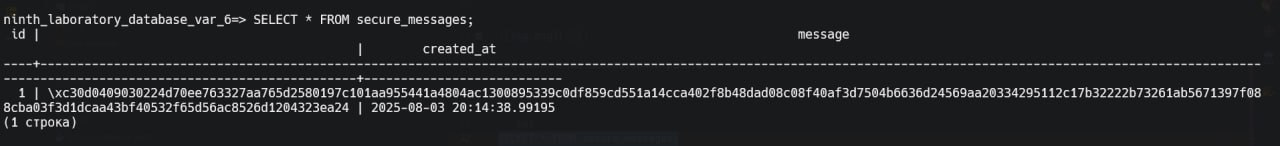


Рисунок 10 – зашифрованные сообщения в базе данных

Проведем теперь дешифровку сообщения, результат представлен на рисунке 11.

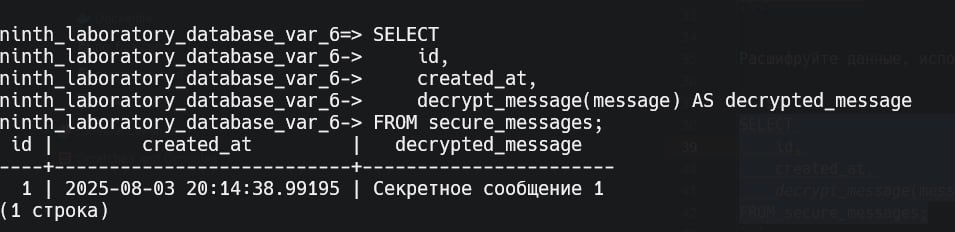


Рисунок 11 – результат дешифровки

Задание 4. Настройте соединение с использованием нужного уровня проверки sslmode (verify-ca, require, verify-full в зависимости от вашего варианта).

Для настройки параметров подключения используется файл pg\_hba.conf, его конфигурация представлена на рисунке 12. Trust используется для postgres из-за особенностей docker, в ином случае не получится просто инициализировать базу данных. Также был создан пользователь myuser, чтобы можно было проводить healthcheck контейнера, в следствии этого был выбран режим scram-sha-256. Для всех остальных есть требование подключение только через SSL.

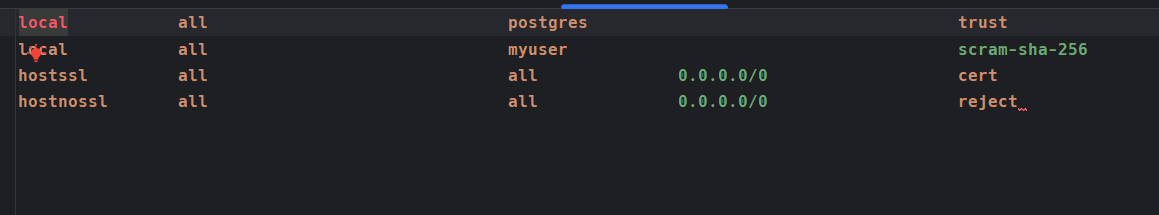


Рисунок 12 – параметры pg\_hba.conf

В 6 варианте не указано какой режим нужно использовать для параметров подключения SSL, в таком случае будет продемонстрирован режим “verify-full” для подключения к базе данных. К сожалению, Pycharm Professional имеет проблемы с подключением с помощью SSL, поэтому подключение будет происходить через консоль. Результат представлен на рисунке 13.

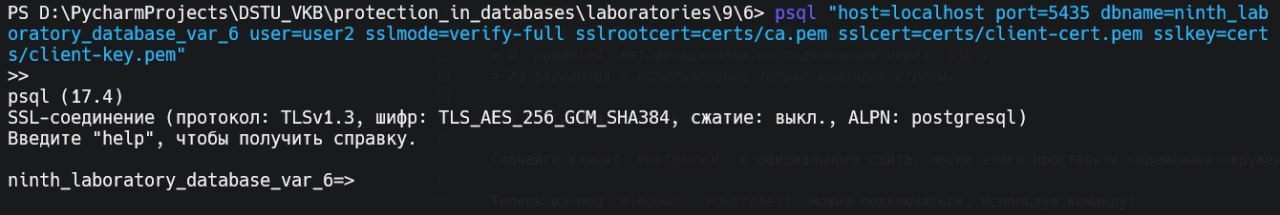


Рисунок 11 – подключение к базе данных через SSL

Вывод: в ходе лабораторной работы были изучены принципы работы SSL-соединений, освоены навыки работы с прозрачным шифрованием, а также создана таблица в базе данных, данные из которой нельзя получить без использования симметричного ключа.

Теория

1.1 Шифрование данных на уровне столбцов

Шифрование данных на уровне столбцов в PostgreSQL — это важный способ защиты личной информации. Это может быть информация, такая как пароли, финансовые данные, личные идентификационные данные и другие чувствительные сведения, которые требуют особой безопасности. В PostgreSQL для этого используется специальное расширение под названием pgcrypto. Оно предоставляет функции для двух типов шифрования: симметричного и асимметричного.

Симметричное шифрование

Описание: При симметричном шифровании используется один и тот же ключ для шифрования (преобразования данных в зашифрованный вид) и дешифрования (возврат к исходному виду). Это подходит для данных, к которым нужен быстрый доступ, и когда нет необходимости обмениваться ключами между сторонами.

Алгоритмы: В pgcrypto для симметричного шифрования используются алгоритмы, такие как AES, Triple DES и Blowfish.

Асимметричное шифрование

Описание: Асимметричное шифрование использует пару ключей: открытый ключ (для шифрования) и закрытый ключ (для дешифрования). Только закрытый ключ может расшифровать данные, зашифрованные открытым ключом, и наоборот.

Алгоритмы: В pgcrypto также реализованы алгоритмы асимметричного шифрования, такие как RSA.

Основные функции шифрования в pgcrypto

В pgcrypto есть функции для симметричного и асимметричного шифрования, которые можно использовать прямо в SQL-запросах для защиты данных.

Симметричное шифрование:

pgp\_sym\_encrypt(data, key): Эта функция шифрует текстовые данные с использованием симметричного ключа.

pgp\_sym\_decrypt(encrypted\_data, key): Эта функция расшифровывает данные, зашифрованные симметричным ключом.

Пример:

INSERT INTO users (username, password)

VALUES ('VasyA', pgp\_sym\_encrypt('password123', 'secret\_key'));

В этом примере пароль password123 для пользователя VasyA шифруется с использованием ключа secret\_key и хранится в зашифрованном виде.

Асимметричное шифрование:

pgp\_pub\_encrypt(data, public\_key): Эта функция шифрует данные с использованием открытого ключа.

pgp\_pub\_decrypt(encrypted\_data, private\_key): Эта функция расшифровывает данные с использованием закрытого ключа.

Пример:

INSERT INTO documents (title, content)

VALUES ('Confidential', pgp\_pub\_encrypt('Это секретный документ', public\_key));

В этом примере содержимое документа "Это секретный документ" шифруется с использованием открытого ключа и сохраняется в таблице документов.

1.2 Шифрование паролей и безопасное хранение ключей

При хранении паролей часто используется хеширование вместо шифрования, потому что хеширование — это необратимый процесс, что означает, что невозможно вернуть исходные данные из хеша. В PostgreSQL для создания хешей используется функция crypt(), которая применяет соль (случайные данные) для обеспечения уникальности хешей, даже если пароли одинаковые.

Пример:

INSERT INTO users (username, password\_hash) VALUES ('VasyA', crypt('password123', gen\_salt('bf')));

В этом примере функция crypt создаёт хеш пароля с использованием алгоритма Blowfish и случайной соли, сгенерированной функцией gen\_salt('bf'). При аутентификации (проверке пользователя) введённый пароль будет хешироваться тем же способом, и результат сравнивается с хешем, хранящимся в базе данных.

Шифрование конфиденциальных данных

Конфиденциальные данные, такие как номера кредитных карт, идентификационные номера и другие важные поля, могут быть зашифрованы с помощью функции pgp\_sym\_encrypt. При выборке данных их можно расшифровать с помощью функции pgp\_sym\_decrypt.

Пример шифрования номера карты:

INSERT INTO payments (user\_id, card\_number) VALUES (1, pgp\_sym\_encrypt('1234-5678-9012-3456', 'secret\_key'));

Пример расшифровки номера карты при выборке:

SELECT user\_id, pgp\_sym\_decrypt(card\_number::bytea, 'secret\_key') AS card\_number FROM payments;

Создание таблицы и вставка данных

Создание таблицы:

CREATE TABLE users (id serial PRIMARY KEY, username VARCHAR(50) UNIQUE NOT NULL, password BYTEA NOT NULL, email BYTEA);

Вставка данных:

INSERT INTO users (username, password, email) VALUES ('VasyA', pgp\_sym\_encrypt('password123', 'strong\_password\_key'), pgp\_sym\_encrypt('VasyA@example.com', 'strong\_password\_key'));

Дешифровка данных:

SELECT username, pgp\_sym\_decrypt(password::bytea, 'strong\_password\_key') AS password, pgp\_sym\_decrypt(email::bytea, 'strong\_password\_key') AS email FROM users WHERE username = 'VasyA';

1.3 Защита соединений с использованием SSL

SSL (Secure Sockets Layer) — это протокол, который защищает данные, передаваемые между клиентом и сервером. Его основная задача — шифровать данные и подтверждать подлинность соединений. В PostgreSQL SSL помогает защитить данные от перехвата, обеспечивает их целостность и подтверждает подлинность сервера, а также клиента, если это необходимо.

SSL создаёт зашифрованный канал для передачи данных, что предотвращает их перехват злоумышленниками. Данные между клиентом и сервером передаются в зашифрованном виде, что обеспечивает их конфиденциальность. Подлинность сервера проверяется с помощью SSL-сертификата. Сервер может также запросить у клиента сертификат для подтверждения его личности, что обеспечивает двухстороннюю аутентификацию.

Настройка SSL в PostgreSQL

Чтобы включить SSL в PostgreSQL, нужно создать или получить SSL-сертификаты и правильно настроить конфигурацию сервера. Сертификат сервера и соответствующий ему закрытый ключ (например, server.crt и server.key) можно создать самостоятельно (самоподписанный сертификат) или запросить у сертификационного центра (например, Let’s Encrypt, DigiCert).

Самоподписанный сертификат подтверждает подлинность сервера, но не гарантирует доверие со стороны клиентов. Его можно использовать для внутренних соединений. Сертификат, подписанный CA (центром сертификации), обеспечивает доверие между клиентом и сервером и позволяет избежать предупреждений о недоверенном сертификате.

Конфигурация SSL

Файлы сертификатов (например, server.crt, server.key, root.crt) помещаются в каталог данных PostgreSQL. После этого в файле конфигурации postgresql.conf SSL включается путём указания необходимых параметров:

ssl = on

ssl\_cert\_file = 'server.crt'

ssl\_key\_file = 'server.key'

ssl\_ca\_file = 'root.crt'

После настройки и перезапуска PostgreSQL все соединения могут быть переведены на защищённые каналы. Клиенты PostgreSQL могут задавать параметры проверки SSL через sslmode, который контролирует уровень проверки безопасности:

disable — SSL не используется.

allow — SSL используется, если поддерживается сервером.

prefer — используется SSL, если сервер поддерживает его (по умолчанию).

require — соединение установится только при наличии SSL.

verify-ca — требуется наличие SSL-сертификата сервера, проверка на доверие к CA.

verify-full — строгая проверка на совпадение имени хоста сервера и имени в сертификате.

Для подключения с параметром sslmode=verify-full клиент требует, чтобы сертификат сервера был подписан доверенным CA и совпадал с именем хоста, указанным в сертификате.

Рекомендации по безопасности

Используйте сертификаты от доверенного CA: это упрощает настройку SSL и повышает доверие клиента.

Регулярно обновляйте сертификаты и ключи: Сертификаты имеют срок действия, и их необходимо обновлять, чтобы снизить риск компрометации данных.

Ограничьте доступ к закрытым ключам: Закрытый ключ должен быть защищён правами доступа, например, с помощью команды:

chmod 600 server.key

Защита ключа критична, так как его утечка позволяет злоумышленникам расшифровать данные.

Клиентская аутентификация через сертификат: это добавляет ещё один уровень защиты, проверяя подлинность клиента на стороне сервера.