Лабораторная работа № 9

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.

Время выполнения лабораторной работы (аудиторные часы): 4 часа

Оборудование и программное обеспечение: ПК с установленной PostgreSQL, расширение pgcrypto, поддержка SSL-сертификатов. Текстовый редактор для написания SQL-запросов.

1. Теоретические сведения

1.1 Шифрование данных на уровне столбцов

Шифрование данных на уровне столбцов в PostgreSQL является важным инструментом для защиты конфиденциальной информации, такой как пароли, финансовые данные, личные идентификационные данные и другие чувствительные сведения, которые требуют повышенной безопасности. В PostgreSQL для этой задачи используется расширение pgcrypto, которое предоставляет функции симметричного и асимметричного шифрования.

Симметричное шифрование:

Описание - Один и тот же ключ используется для шифрования и дешифрования данных. Симметричное шифрование подходит для данных, к которым нужен быстрый доступ, и для которых нет необходимости в обмене ключами между сторонами.

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

Рисунок 1 – Симметричное шифрование отрытого текста

Асимметричное шифрование:

Описание **-** Использует пару ключей — открытый (для шифрования) и закрытый (для расшифрования). Только закрытый ключ может расшифровать данные, зашифрованные открытым ключом, и наоборот.

Алгоритмы **-** В pgcrypto так же реализованы алгоритмы асимметричного шифрования, такие как RSA.

Рисунок 2 – Асимметричное шифрование

Основные функции шифрования в pgcrypto

В pgcrypto реализованы функции для симметричного и асимметричного шифрования, которые можно применять непосредственно в SQL-запросах для защиты данных.

Симметричное шифрование:

pgp_sym_encrypt(data, key) - Зашифровывает текстовые данные с использованием симметричного ключа.

pgp_sym_decrypt(encrypted_data, key) - Расшифровывает зашифрованные симметричным ключом данные.

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): Зашифровывает данные с использованием открытого ключа.

pgp_pub_decrypt(encrypted_data, private_key): Расшифровывает данные с использованием закрытого ключа.

Пример:

```
INSERT INTO documents (title, content)

VALUES ('Confidential', pgp_pub_encrypt('Это секретный документ', public_key));
```

Шифрование паролей и других конфиденциальных данных требует надёжного метода шифрования и безопасного хранения ключей. При хранение паролей часто используется хеширование вместо шифрования, так как хеширование необратимо. В pgcrypto для создания хешей используется функция crypt(), которая применяет к данным соль для обеспечения уникальности хешей, даже если пароли одинаковые.

```
INSERT INTO users (username, password_hash) VALUES ('VasyA', crypt('password123', gen_salt('bf')));
```

Здесь функция crypt создаёт хеш пароля с использованием алгоритма Blowfish и случайной соли (gen_salt('bf')). При аутентификации пользовательский ввод будет хешироваться тем же способом, и результат сравнивается с хешем, хранящимся в базе данных.

Шифрование конфиденциальных данных - конфиденциальные данные, такие как номера карт, идентификационные номера и другие поля, могут быть зашифрованы с помощью pgp_sym_encrypt. При выборке данных их можно расшифровать функцией pgp_sym_decrypt.

Шифрование даных номера карты для выборки:

```
INSERT INTO payments (user_id, card_number)
VALUES (1, pgp_sym_encrypt('1234-5678-9012-3456', 'secret_key'));
```

Расшифровка номера карты при выборке:

```
SELECT user_id, pgp_sym_decrypt(card_number::bytea, 'secret_key') AS card_number FROM payments;
```

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';
```

1.2 Защита соединений с использованием SSL

SSL (Secure Sockets Layer) — это криптографический протокол, разработанный для защиты данных, передаваемых между клиентом и сервером. Основное назначение SSL — шифрование данных и аутентификация соединений. В PostgreSQL SSL позволяет защитить данные от перехвата, обеспечить их целостность и подтвердить подлинность сервера, а при необходимости — и клиента. SSL создаёт зашифрованный канал передачи данных, что предотвращает их перехват злоумышленниками. Данные между клиентом и сервером передаются в зашифрованном виде, что обеспечивает их конфиденциальность. Подлинность сервера проверяется через SSL-сертификат. Сервер может также запросить у клиента сертификат для подтверждения его личности, обеспечивая двухстороннюю аутентификацию. Сертификаты подтверждаются центром сертификации (CA), который выступает доверенным посредником и подтверждает подлинность ключей.

Сертификат сервера — это цифровой документ, содержащий публичный ключ сервера и подписанный CA. Клиент может доверять серверу, если сертификат валиден и выдан доверенным CA. Закрытый ключ сервера хранится в защищённой среде и используется для расшифровки данных, зашифрованных клиентом с помощью публичного ключа. Цепочка доверия устанавливается через корневой сертификат CA, подтверждая, что сертификат сервера был выдан доверенным источником. Таким образом, проверяется подлинность обеих сторон.

Рисунок 3 – Процесс взаимодействие Пользователь - Сервер

Настройка SSL в PostgreSQL

SSL можно включить в PostgreSQL путём создания или получения SSL-сертификатов и корректной настройки конфигурации сервера. Сертификат сервера и соответствующий ему закрытый ключ (**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 обеспечивает гарантии от доверенных сторон. Его можно использовать для внутренних соединений. Сертификат, подписанный CA, обеспечивает доверие между клиентом и сервером, позволяя избежать предупреждений о недоверенном сертификате. В PostgreSQL файлы сертификатов (**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. Так же клиенты PostgreSQL могут задавать параметры проверки **SSL** через **sslmode**, который контролирует требуемый уровень проверки безопасности:

disable — SSL не используется.

allow — SSL используется, если поддерживается сервером.

prefer — используется SSL, если сервер поддерживает SSL (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, указанного в сертификате. С помощью **OpenSSL** требуется создать сертификат и ключ. Полученные файлы **server.crt** и **server.key** необходимо поместить в каталог данных PostgreSQL. Для подключения клиента, например, с помощью psql, можно задать уровень проверки SSL:

```
openssl req -new -text -out server.csr
openssl rsa -in privkey.pem -out server.key
```

```
openssl x509 -in server.csr -out server.crt -req -signkey server.key -
days 365
psql "host=hostname dbname=mydb user=myuser sslmode=verify-full"
```

Этот параметр заставит клиент проверять сертификат сервера на подлинность и соответствие имени хоста, что обеспечит безопасность передачи данных. Использование сертификатов от доверенного CA. Сертификаты, подписанные CA, упрощают настройку SSL и повышают доверие клиента, так как в случае само подписанных сертификатов клиенту приходится вручную добавлять исключения. Регулярное обновление сертификатов и ключей. Сертификаты имеют срок действия, после которого требуется их обновление. Регулярная смена ключей снижает риск компрометации данных. Ограничение доступа 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. В некоторых системах рекомендуется клиентская аутентификация через сертификат. Она может быть настроена путём установки сертификата клиента и проверки его подлинности сервером, что добавляет ещё один уровень защиты.

1. **Настройка**: сгенерируйте SSL-сертификаты для сервера: сертификат и закрытый ключ (server.crt и server.key). Для этого можно использовать утилиту OpenSSL или создать сертификаты, подписанные доверенным центром сертификации (CA).

Переместите файлы server.crt и server.key в каталог данных PostgreSQL. Убедитесь, что закрытый ключ (server.key) защищён от несанкционированного доступа, например, через права доступа **chmod 600.**

Включите SSL в конфигурации PostgreSQL, открыв файл **postgresql.conf** и добавив следующие параметры:

```
ssl = on
ssl_cert_file = 'server.crt'
ssl_key_file = 'server.key'
ssl_ca_file = 'root.crt'
```

Перезапустите сервер PostgreSQL для применения настроек. Настройка доступа через SSL в **pg_hba.conf**. Откройте файл **pg_hba.conf** и добавьте правило для принудительного использования SSL при подключении к базе данных:

```
hostssl mydb myuser 0.0.0.0/0 md5
```

2. **Проверка доступа:** убедитесь, что доступ к базе данных **mydb** возможен только для пользователя **myuser** и только через SSL-соединение. Шифрование данных на уровне столбцов с использованием **pgcrypto**. Создайте таблицу **users** с полями **username** и **password**. Поле **password** должно храниться в

зашифрованном виде. Используйте функцию **pgp_sym_encrypt** для шифрования пароля при вставке записи.

Пример запроса:

```
INSERT INTO users (username, password) VALUES ('john_doe',
pgp_sym_encrypt('password123', 'encryption_key'));
```

При выборке данных используйте функцию **pgp_sym_decrypt**, чтобы расшифровать поле **password**:

```
SELECT username, pgp_sym_decrypt(password::bytea, 'encryption_key') AS
password FROM users;
```

3. **Проверка работоспособности**: проверка работы SSL-соединения подключитесь к базе данных PostgreSQL с помощью клиента **psql**, указав параметр **sslmode=verify-full**:

```
psql "host=hostname dbname=mydb user=myuser sslmode=verify-full"
```

Убедитесь, что при установлении соединения используется SSL. При подключении в режиме **verify-full** клиент проверит сертификат сервера и соответствие имени хоста. Попробуйте подключиться к серверу, используя параметры **sslmode=require** и **sslmode=disable**, чтобы протестировать настройку SSL и убедиться, что доступ возможен только через защищённое соединение.
Проверьте шифрование и дешифрование данных, выполняя вставку и выборку записей из таблицы **users**, и убедитесь, что данные надёжно защищены.

2. Задание

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

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'
```

2. Создайте таблицу согласно вашему варианту, добавив соответствующие столбцы и обеспечив зашифрованное хранение данных в указанных полях (например, с использованием **pgp_sym_encrypt**).

Создание таблицы:

```
CREATE TABLE client_data (client_id SERIAL PRIMARY KEY, name
VARCHAR(100),email BYTEA);
```

3. Вставляйте данные в зашифрованные столбцы, используя функции pgcrypto (например, pgp_sym_encrypt для симметричного шифрования):

```
INSERT INTO client_data (name, email) VALUES ('Ben Den',
pgp_sym_encrypt('bd@example.com','secret_key'));
```

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

Для тестирования подключения используйте psql или другой клиент PostgreSQL:

```
psql "host=your_host dbname=your_db user=your_user sslmode=verify-
full"
```

5. Выполните выборку данных из таблицы и расшифруйте их (если необходимо), используя pgp_sym_decrypt.

Выполните дополнительные требования по вашему варианту

Вариант	**Задание**
1.	Создайте таблицу client_data с полями client_id, name, и email. Поле email должно быть зашифровано при вставке с использованием симметричного шифрования. Настройте PostgreSQL так, чтобы подключение к базе данных было возможно только через SSL-соединение.
2.	Создайте таблицу payment_records с полями payment_id, account_number, и amount. Поле account_number зашифруйте с помощью симметричного ключа. Настройте SSL-соединение, при котором клиент должен использовать sslmode=verify-ca для подключения.
3.	Настройте базу данных так, чтобы доступ к таблице employee_info был возможен только по SSL-соединению с параметром sslmode=verify-full. Поле salary в таблице должно храниться в зашифрованном виде и требовать дешифровки при выборке.
4.	Создайте таблицу user_accounts с полями user_id, username, и password. Поле password должно быть зашифровано с использованием симметричного метода. Настройте сервер PostgreSQL так, чтобы требовалась проверка клиентского сертификата при подключении.
5.	Создайте таблицу medical_records с полями record_id, patient_name, и diagnosis. Поле diagnosis должно быть зашифровано симметричным шифрованием. Настройте PostgreSQL на использование SSL, принуждая клиента подключаться с sslmode=require.
6.	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.

7.	Создайте таблицу order_details с полями order_id, product_name, и credit_card. Поле credit_card должно быть зашифровано симметричным методом. Настройте PostgreSQL так, чтобы доступ к базе данных был возможен только по SSL-соединению с использованием параметра sslmode=verify-ca.
8.	Создайте таблицу confidential_reports с полями report_id, title, и content. Поле content должно быть зашифровано симметричным ключом при вставке. Проверьте настройку SSL-соединения, используя sslmode=verify-full для подтверждения подлинности сервера.
9.	Настройте PostgreSQL так, чтобы подключение к базе данных было возможно только при использовании клиентского SSL-сертификата. Создайте таблицу contract_data с полем contract_text, которое должно быть зашифровано с использованием симметричного шифрования.
10.	Создайте таблицу financial_transactions с полями transaction_id, sender_account, и receiver_account. Поля sender_account и receiver_account зашифруйте с использованием симметричного ключа. Настройте SSL-соединение так, чтобы клиент мог подключиться только с sslmode=verify-full.

.

Контрольные вопросы:

1. В чём разница между симметричным и асимметричным шифрованием, и в каких случаях предпочтительнее использовать каждый из них?
2. Какую роль выполняет расширение pgcrypto в PostgreSQL при работе с зашифрованными данными?
3. Какие функции pgcrypto используются для симметричного шифрования и дешифрования данных? Приведите примеры их использования.
4. Как создаётся SSL-сертификат для PostgreSQL, и зачем нужен закрытый ключ сервера?
5. Какие уровни проверки SSL-соединения предоставляет параметр sslmode? Объясните назначение каждого из них.
6. Какой файл конфигурации PostgreSQL необходимо изменить для включения SSL, и какие параметры следует указать?
7. Как параметр hostssl в файле pg_hba.conf влияет на доступ к базе данных PostgreSQL?
8. Почему для безопасного хранения данных часто используют шифрование на уровне столбцов? Как это защищает данные в случае утечки?
9. Какие действия необходимо выполнить для включения двусторонней аутентификации с клиентскими сертификатами в PostgreSQL?
10. Как можно проверить, что соединение с базой данных PostgreSQL действительно зашифровано SSL?

Литература

1. PostgreSQL Documentation. PostgreSQL 15 Documentation [Электронный ресурс]. — URL: https://www.postgresql.org/docs/.
2. PostgreSQL pgcrypto Module. PostgreSQL Encryption Module Documentation [Электронный ресурс]. — URL: https://www.postgresql.org/docs/current/pgcrypto.html.
3. Krawczyk, H., Bellare, M., & Canetti, R. HMAC: Keyed-Hashing for Message Authentication // RFC 2104. — Network Working Group, 1997. — P. 1–11.
4. Иванов, П. Н. Шифрование данных и защита соединений в PostgreSQL / П. Н. Иванов, В. К. Смирнов // Информационные технологии безопасности. — 2022. — Т. 13, № 3. — С. 43–49.
5. SSL and TLS Encryption. IBM Knowledge Center [Электронный ресурс]. — URL: https://www.ibm.com/docs/en/ssl-tls.