ЛР7. Шифрование. Мониторинг

Теоретическая (тестовая) часть

Лекция 9. Аудит

Лекция 10. Мониторинг

Лекция 11. Шифрование

Практическая часть

Шифрование

В базе данных необходимо реализовать хранение персональных данных сотрудников таким образом, чтобы:

1) Чувствительные данные (зарплата, медицинская информация) не хранились в открытом виде и были зашифрованы на уровне базы данных.
2) Разные категории пользователей имели разный уровень доступа:
   - одна роль – только просмотр открытых данных;
   - другая роль – полный доступ с возможностью расшифровки и изменения данных.
3) Ключи шифрования не должны храниться в таблицах базы данных.

1. Подготовка среды:
   1.1. Создайте новую схему для выполнения лабораторной работы.
   1.2. Подключите расширение `pgcrypto`.
2. Создание таблицы: в созданной схеме `lab_crypto` создайте таблицу `employees_secure` со следующими столбцами:
   - `id` – уникальный идентификатор сотрудника;
   - `full_name` – ФИО сотрудника;
   - `email` – электронная почта (уникальная);
   - `salary_encrypted` – зашифрованная зарплата;
   - `medical_info_encrypted` – зашифрованная медицинская информация;
   - `created_at` – дата добавления записи.
3. Реализация шифрования данных:
   3.1. Реализуйте функции шифрования и расшифровки зарплаты с использованием симметричного шифрования. Функции назовите `encrypt_salary/decrypt_salary`.

3.2. Реализуйте функции шифрования и расшифровки медицинской информации с использованием асимметричного шифрования (функции `encrypt_medical/decrypt_medical`).

Для реализации этих функций используйте расширение `pgcrypto`. Необходимо использовать разные ключи шифрования для указанных данных. Для передачи ключей используйте параметры текущей сессии, например:

```
SET app.salary_key = 'salary_key';
SELECT current_setting('app.salary_key', true);
```

4. Создание представлений:
   4.1. Создайте представление `v_employees_public`, которое возвращает следующие данные: `id, full_name, email, created_at`.
   4.2. Создайте представление `v_employees_full`, которое возвращает следующие данные: `id`, `full_name`, `email`, `salary_decrypted` (расшифрованная зарплата), `medical_info_decrypted` (расшифрованная медицинская информация), `created_at`.
5. Создание ролей и разграничение доступа. Создайте роли и установите необходимые права доступа к соответствующим объектам БД:
   - `hr_readonly`:
     - имеет право только читать открытые данные сотрудников с помощью представления `v_employees_public`;
     - не имеет доступа к зашифрованным данным (к исходной таблице `employees_secure`);
     - не может выполнять шифрование или дешифрование данных с помощью созданных функций.
   - `hr_full`:
     - может добавлять и изменять записи сотрудников;
     - может шифровать данные при вставке и обновлении;
     - может читать расшифрованные значения зарплаты и медицинских данных с помощью представления `v_employees_full`;
     - может читать открытые данные сотрудников с помощью представления `v_employees_public`;
6. Ограничение доступа к функциям: запретите вызов созданных крипто-функций для `public` и разрешите их выполнение только роли `hr_full`.
7. Проверка привилегий. Продемонстрируйте:

7.1. Возможность добавления записи сотрудника с помощью роли `hr_full` с использованием функций шифрования.
7.2. Возможность изменения зарплаты сотрудника с помощью роли `hr_full` и отсутствие такой возможности для роли `hr_readonly`.
7.3. Возможность чтения расшифрованных данных таблицы с помощью роли `hr_full`.
7.4. Возможность чтения открытых данных таблицы с помощью роли `hr_readonly`.
7.5. Отсутствие возможности чтения расшифрованных данных таблицы с помощью роли `hr_readonly`.
7.6. Отсутствие возможности шифрования данных с помощью созданной функции шифрования ролью `hr_readonly`.

Мониторинг

1. Используя систему просмотра статистики PostgreSQL:
   1.1. Получить информацию обо всех активных подключениях к СУБД.
   1.2. Получить информацию о 5 таблицах с наибольшим количеством операций добавления записей.
   1.3. Получить информацию о 5 индексах с наибольшим количеством произведённых сканирований.
   1.4. Используя модуль `pg_stat_statements`, вывести информацию о самых долгих запросах (с наибольшим общим временем выполнения запроса).
   1.5. Используя модуль `pg_stat_statements`, вывести информацию о запросах, вернувших наибольшее число строк.
2. Настроить систему мониторинга Prometheus + Grafana. Подключить как минимум один экспортер метрик из СУБД по выбору (MySQL, PostgreSQL, MongoDB). Настроить панель мониторинга (дашборд) для отображения показателей работы СУБД.