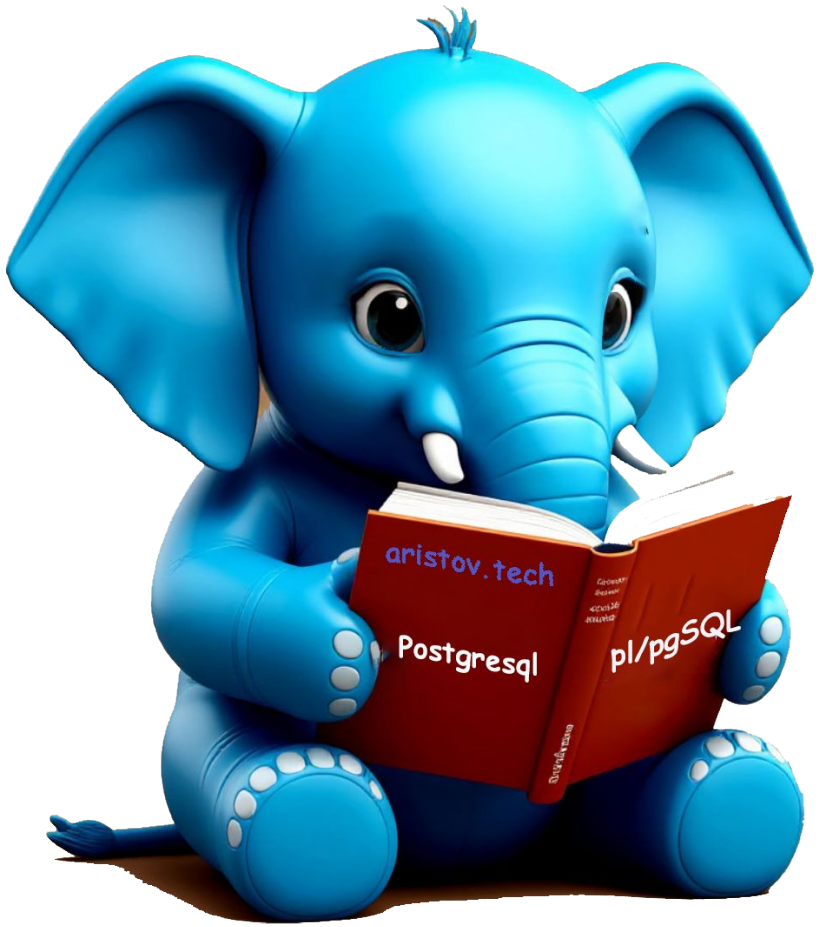
Аристов Евгений

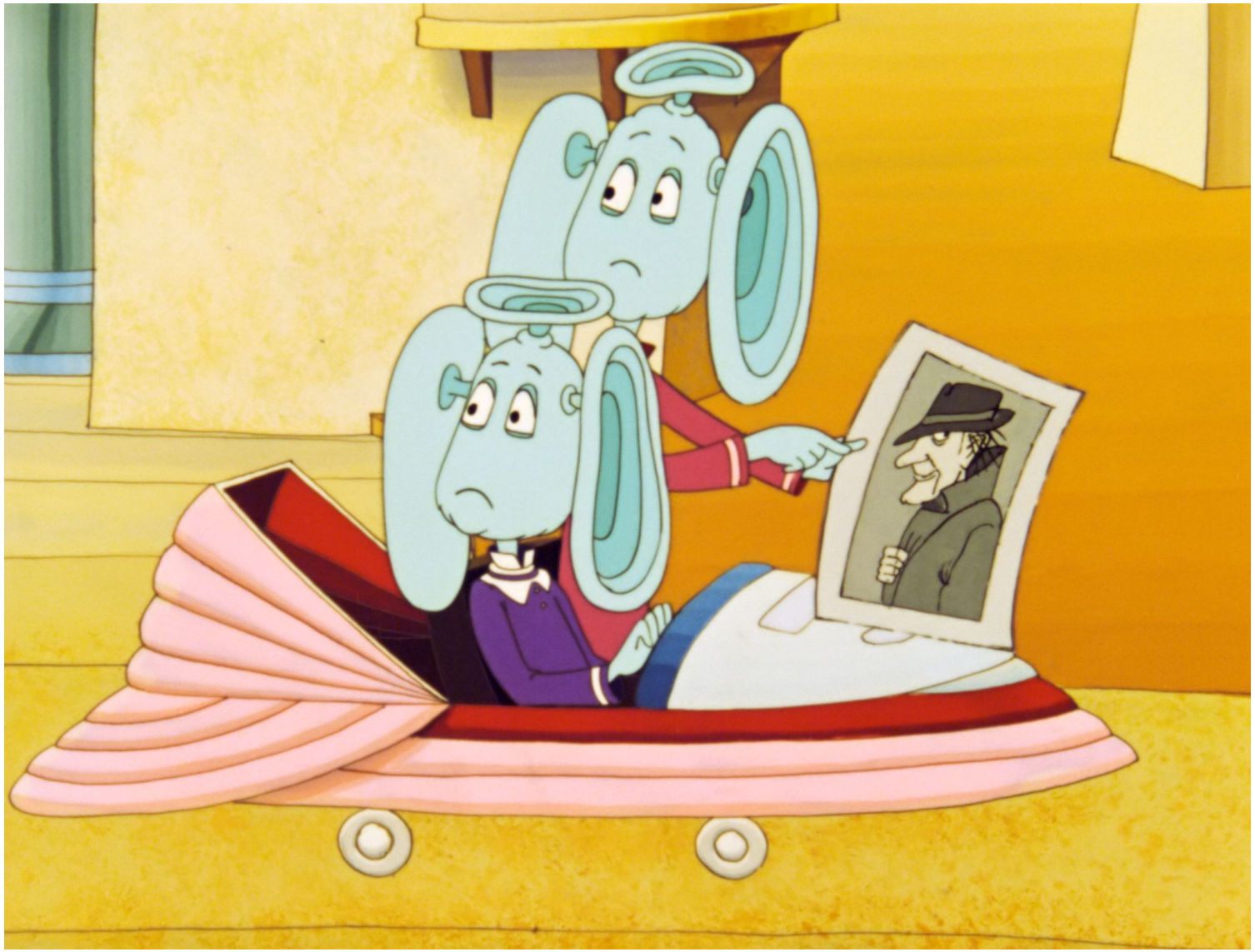
PL/pgSQL в PostgreSQL

за 31 занятие



SQL инъекции

https://aristov.tech



Аристов Евгений Николаевич

Founder & CEO aristov.tech
25 лет занимаюсь разработкой БД и ПО

Архитектор высоконагруженных баз данных и инфраструктуры

Спроектировал и разработал более ста проектов для финансового сектора, сетевых магазинов, фитнес-центров, отелей.

Сейчас решаю актуальные для бизнеса задачи: аудит и оптимизация БД и инфраструктуры, миграция на PostgreSQL, обучение сотрудников.

Автор более 10 практических курсов по PostgreSQL, MySQL, Mongo и др..

Автор книг по PostgreSQL. Новинка **PostgreSQL 16: лучшие практики оптимизации**

Правила вебинара

Задаем вопрос в чат

Вопросы вижу, отвечу в момент логической паузы

Если есть вопрос голосом - поставьте знак ? в чат

Если остались вопросы, можно их задать на следующем занятии или в комментариях к записи

https://aristov.tech

Маршрут вебинара

SQL инъекции - принцип атаки

Варианты защиты

Best practice

https://aristov.tech

SQL injection

SQL инъекции

SQL инъекции в PostgreSQL — это уязвимости безопасности, возникающие при некорректной обработке пользовательского ввода, позволяющие злоумышленнику выполнить произвольный SQL-код.

SQL инъекции

```
-- УЯЗВИМЫЙ КОД
CREATE OR REPLACE FUNCTION vulnerable_login(username TEXT, password TEXT)
RETURNS BOOLEAN AS $$
DECLARE
    query TEXT;
    result BOOLEAN;
BEGIN
    query := 'SELECT EXISTS(SELECT 1 FROM users WHERE username = ''' ||
            username || ''' AND password = ''' || password || ''')';
    EXECUTE query INTO result;
    RETURN result;
END;
$$ LANGUAGE plpgsql;
```

SQL инъекции

-- АТАКА

SELECT vulnerable_login('admin', 'anything" OR "1"="1');

-- Сгенерированный запрос:

-- SELECT EXISTS(SELECT 1 FROM users WHERE username = 'admin' AND password = 'anything' OR '1'='1')

https://aristov.tech

SQL инъекции

Защита:

Использование параметризованных запросов

Использование форматирования с экранированием

Валидация входных данных

Использование защищенных процедур с нужными правами и отзыв прямого доступа

Более подробно посмотрим на следующей теме - динамический SQL

SQL инъекции

!!!НИКОГДА НЕ ИСПОЛЬЗУЙТЕ!!!

```
-- Конкатенацию строк для запросов
query := 'SELECT ... WHERE name = '' || user_input || '';

-- Динамический SQL без экранирования
EXECUTE 'SELECT ... WHERE name = '' || user_input || '';

-- Прямое использование пользовательского ввода
EXECUTE user_input;
```

https://aristov.tech

SQL инъекции. Итоги

Ключевые принципы защиты:

1. **Всегда** используйте параметризованные запросы
2. **Валидируйте** и санируйте все пользовательские данные
3. **Принцип минимальных привилегий** для пользователей БД
4. **Экранируйте** идентификаторы с помощью format() и %I
5. **Логируйте** подозрительную активность
6. **Регулярно тестируйте** на уязвимости

https://aristov.tech

SQL инъекции. Доп.материальчик

[SQL injection для начинающих. Часть 1]

https://aristov.tech

Практика

Итоги

Итоги

Остались ли вопросы?

Увидимся на следующем занятии

Спасибо за внимание!

Когда дальше и куда?

Аристов Евгений