FB.SQL\_PREPARED\_STATEMENT\_GENERATED\_FROM\_NONCONSTANT\_STRING

Код создает подготовленный оператор SQL из непостоянной строки. Если при построении этой строки используются непроверенные испорченные данные пользователя, то можно использовать SQL-инъекцию, чтобы заставить подготовленный оператор сделать что-то неожиданное и нежелательное.

Таблица №1 – Основные характеристики детектора FB.SQL\_PREPARED\_STATEMENT\_GENERATED\_FROM\_NONCONSTANT\_STRING

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Security Vulnerability (SQL Injection) |
| CWE | CWE89 |
| CVE | --- |
| Возможные последствия | SQL-инъекции через динамическое построение запросов; обход параметризованных запросов; компрометация базы данных |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public static void incorrectTest() {  // Некорректно: динамическое создание SQL из ненадежного источника  String tableName = "users; DROP TABLE users; --"; // Может прийти из пользовательского ввода  try (Connection conn = getConnection()) {  String sql = "SELECT \* FROM " + tableName; // Опасная конкатенация  PreparedStatement stmt = conn.prepareStatement(sql); // Ложное чувство безопасности  stmt.executeQuery();  System.out.println("Query executed (INSECURE)");  } catch (SQLException e) {  e.printStackTrace();  }  } | public static void correctTest() {  // Корректно: валидация имен таблиц/колонок  String tableName = "users";  try (Connection conn = getConnection()) {  if (!isValidTableName(tableName)) {  throw new IllegalArgumentException("Invalid table name");  }  String sql = "SELECT \* FROM " + tableName;  PreparedStatement stmt = conn.prepareStatement(sql);  stmt.executeQuery();  System.out.println("Query executed securely");  } catch (SQLException e) {  e.printStackTrace();  }  } |