FB.SQL\_NONCONSTANT\_STRING\_PASSED\_TO\_EXECUTE

Этот метод вызывает метод выполнения или addBatch для оператора SQL со строкой, которая создается динамически. Вместо этого следует рассмотреть возможность использования подготовленного оператора. Он более эффективен и менее уязвим для атак с использованием SQL-инъекций.

Таблица №1 – Основные характеристики детектора FB.SQL\_NONCONSTANT\_STRING\_PASSED\_TO\_EXECUTE

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Security Vulnerability (SQL Injection) |
| CWE | CWE89 |
| CVE | --- |
| Возможные последствия | SQL-инъекции; несанкционированный доступ к БД; утечка/модификация данных; выполнение произвольных SQL-команд |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public static void incorrectTest() {  // Некорректно: конкатенация SQL-запроса  String username = "admin' OR '1'='1"; // Может прийти из пользовательского ввода  try (Connection conn = getConnection()) {  String sql = "SELECT \* FROM users WHERE username = '" + username + "'";  PreparedStatement stmt = conn.prepareStatement(sql); // Уязвимость!  var rs = stmt.executeQuery();  System.out.println("SQL executed (INSECURE)");  } catch (SQLException e) {  e.printStackTrace();  }  } | public static void correctTest() {  // Корректно: параметризованные запросы  String username = "admin' OR '1'='1";  try (Connection conn = getConnection()) {  String sql = "SELECT \* FROM users WHERE username = ?";  PreparedStatement stmt = conn.prepareStatement(sql);  stmt.setString(1, username); // Безопасная подстановка  var rs = stmt.executeQuery();  System.out.println("SQL executed securely");  } catch (SQLException e) {  e.printStackTrace();  }  } |