FB.DB\_DUPLICATE\_BRANCHES

Детектор определяет ошибку, в ходе которой метод использует один и тот же код для реализации двух ветвей условного перехода. Убедитесь, что это не ошибка кодирования.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Наличие одинакового кода в разных ветвях условного оператора (if-else, switch) может указывать на логическую ошибку или избыточность. |
| Комментарий | Проверьте, действительно ли обе ветви должны выполнять одинаковые действия. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  class MyClass {  String checkValue(int value) {  if (value > 10) {  return "Yes";  } else {  return "Yes"; // Дублирующийся код  }  }  }  MyClass myClass = new MyClass();  System.out.println(myClass.checkValue(15));  System.out.println(myClass.checkValue(5));  } | public static void correctTest() {  class MyClass {  String checkValue(int value) {  // Убрана ненужная ветка  return "Yes";  }  }  MyClass myClass = new MyClass();  System.out.println(myClass.checkValue(15));  System.out.println(myClass.checkValue(5));  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Удалите избыточную ветвь:** Если обе ветви выполняют один и тот же код, удалите одну из ветвей и оставьте код, который должен выполняться в любом случае. Пересмотрите условие, чтобы оно охватывало все необходимые сценарии.
2. **Переместите общий код за пределы условного оператора:** Если часть кода является общей для обеих ветвей, вынесите этот код за пределы if-else или switch конструкции, чтобы он выполнялся только один раз.
3. **Проверьте логику:** Тщательно проанализируйте логику условного оператора. Возможно, дублирование кода указывает на ошибку в понимании требований или неправильно написанное условие.
4. **Используйте логические операторы:** Иногда дублирование можно устранить, объединив условия с помощью логических операторов (||, &&) для более компактного выражения нужной логики.
5. **Рефакторинг:** Если дублирующийся код более сложный, рассмотрите возможность выделения его в отдельный метод, который затем вызывается из обеих ветвей (если логика действительно должна быть одинаковой). Однако чаще всего дублирование указывает на то, что одна из ветвей является лишней.