FB.CO\_COMPARETO\_INCORRECT\_FLOATING

Этот метод сравнивает значения типа double или float, используя такой шаблон: val1 > val2 ? 1: значение1 <значение2? -1 : 0. Этот шаблон работает некорректно для значений -0.0 и NaN, что может привести к неправильному результату сортировки или нарушению сбора (если в качестве ключей используются сравниваемые значения). Стоит рассмотреть возможность использования статических методов Double.compare или Float.compare, которые правильно обрабатывают все особые случаи.

Таблица №1 – Основные характеристики детектора FB.CO\_COMPARETO\_INCORRECT\_FLOATING

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Algorithmic Error |
| CWE | --- |
| CVE | --- |
| Возможные последствия | * Некорректная сортировка объектов * Нарушение контракта compareTo() * Неожиданное поведение в коллекциях |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public static void incorrectTest() {  class FloatingPoint implements Comparable<FloatingPoint> {  double value;    public FloatingPoint(double value) {  this.value = value;  }    // Некорректно: прямое сравнение double через ==  @Override  public int compareTo(FloatingPoint other) {  if (this.value == other.value) { // Опасное сравнение  return 0;  }  return this.value < other.value ? -1 : 1;  }  }    FloatingPoint fp1 = new FloatingPoint(0.1 + 0.2);  FloatingPoint fp2 = new FloatingPoint(0.3);    // Может дать неожиданный результат из-за погрешности double  System.out.println("Incorrect comparison: " + fp1.compareTo(fp2));  } | public static void correctTest() {  class FloatingPoint implements Comparable<FloatingPoint> {  double value;  private static final double EPSILON = 1e-10;    public FloatingPoint(double value) {  this.value = value;  }    // Корректно: сравнение с учетом погрешности  @Override  public int compareTo(FloatingPoint other) {  if (Math.abs(this.value - other.value) < EPSILON) {  return 0;  }  return this.value < other.value ? -1 : 1;  }  }    FloatingPoint fp1 = new FloatingPoint(0.1 + 0.2);  FloatingPoint fp2 = new FloatingPoint(0.3);    System.out.println("Correct comparison: " + fp1.compareTo(fp2));  } |