FB.FL\_MATH\_USING\_FLOAT\_PRECISION

Метод выполняет математические операции, используя точность с плавающей запятой. Точность с плавающей запятой очень неточна. Например, 16777216.0f + 1.0f = 16777216.0f. Вместо этого стоит рассмотреть возможность использования двойной математики.

Таблица №1 – Основные характеристики детектора FB.FL\_MATH\_USING\_FLOAT\_PRECISION

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Precision Loss |
| CWE | --- |
| CVE | --- |
| Возможные последствия | * Накопление ошибок округления * Неправильные результаты вычислений * Проблемы при сравнении чисел |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public static void incorrectTest() {  // Некорректно: использование float для точных вычислений  float accountBalance = 100.00f;  for (int i = 0; i < 100; i++) {  accountBalance += 0.01f; // Накопление ошибки округления  }  System.out.println("Incorrect balance: " + accountBalance);  } | public static void correctTest() {  // Корректно: использование BigDecimal для точных вычислений  java.math.BigDecimal accountBalance = new java.math.BigDecimal("100.00");  java.math.BigDecimal increment = new java.math.BigDecimal("0.01");    for (int i = 0; i < 100; i++) {  accountBalance = accountBalance.add(increment);  }  System.out.println("Precise balance: " + accountBalance);    // Альтернатива с double (для менее критичных вычислений)  double altBalance = 100.00;  for (int i = 0; i < 100; i++) {  altBalance += 0.01;  }  System.out.println("Double balance: " + altBalance);  } |