FB.INT\_BAD\_REM\_BY\_1

Детектор определяет ошибку, в ходе которой любое выражение (exp % 1) гарантированно всегда возвращает ноль.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Использование exp % 1 бессмысленно, так как любой целочисленный остаток от деления на 1 всегда равен 0. |
| Комментарий | Следует удалить % 1, так как он не влияет на результат. Если проверяется делимость, возможно, имелось в виду exp % n == 0 для какого-то другого n. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  int x = 10;  int y = x % 1; // x % 1 всегда 0  System.out.println("Неправильный остаток от деления: " + y);  } | public static void correctTest() {  int x = 10;  int z = x % 2;  System.out.println("Правильный остаток от деления: " + z);  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Удалите операцию % 1:** Просто уберите % 1 из выражения. Это не повлияет на результат, но сделает код чище.
2. **Проверьте предполагаемую логику:** Подумайте, что вы на самом деле хотели сделать с помощью этой операции. Возможно, вы имели в виду проверку делимости на другое число.
3. **Замените на проверку делимости:** Если вашей целью была проверка, делится ли число на какое-либо значение n без остатка, замените exp % 1 на exp % n == 0.
4. **Проанализируйте контекст:** Рассмотрите, в каком контексте используется exp % 1. Возможно, это часть более сложного выражения, которое следует упростить.