FB.VO\_VOLATILE\_INCREMENT

Этот код увеличивает/уменьшает изменяемое поле. Приращения/уменьшения изменчивых полей не являются атомарными. Если несколько потоков увеличивают/уменьшают поле одновременно, приращения/уменьшения могут быть потеряны.

Таблица №1 – Основные характеристики детектора FB.VO\_VOLATILE\_INCREMENT

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Concurrency Vulnerability |
| CWE | --- |
| CVE | --- |
| Возможные последствия | Неатомарные операции с volatile переменными; потеря обновлений значения в многопоточной среде; непредсказуемые результаты инкремента |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public static void incorrectTest() {  class Counter {  private volatile int count = 0;    public void increment() {  count++; // Неатомарная операция с volatile  }  }    Counter counter = new Counter();  Runnable task = () -> {  for (int i = 0; i < 1000; i++) {  counter.increment();  }  };    Thread t1 = new Thread(task);  Thread t2 = new Thread(task);    t1.start();  t2.start();    try {  t1.join();  t2.join();  } catch (InterruptedException e) {  Thread.currentThread().interrupt();  }    System.out.println("Incorrect count: " + counter.count);  } | public static void correctTest() {  class Counter {  private final java.util.concurrent.atomic.AtomicInteger count =  new java.util.concurrent.atomic.AtomicInteger(0);    public void increment() {  count.incrementAndGet(); // Атомарный инкремент  }  }    Counter counter = new Counter();  Runnable task = () -> {  for (int i = 0; i < 1000; i++) {  counter.increment();  }  };    Thread t1 = new Thread(task);  Thread t2 = new Thread(task);    t1.start();  t2.start();    try {  t1.join();  t2.join();  } catch (InterruptedException e) {  Thread.currentThread().interrupt();  }    System.out.println("Correct count: " + counter.count.get());  } |