FB.IS\_INCONSISTENT\_SYNC

Доступ к полям этого класса осуществляется непоследовательно с точки зрения синхронизации. Класс содержит смесь заблокированных и разблокированных доступов. По крайней мере один заблокированный доступ был выполнен одним из собственных методов класса. Количество несинхронизированных обращений к полям (чтение и запись) составило не более одной трети всех обращений, при этом операции записи имеют вдвое больший вес, чем операции чтения. Типичная ошибка, соответствующая этому шаблону ошибок, — отсутствие синхронизации. Стоит обратить внимание, что в этом детекторе существуют различные источники погрешностей; например, детектор не может статически обнаруживать все ситуации, в которых блокировка удерживается. Кроме того, даже если детектор точен, различая заблокированный и разблокированный доступ, рассматриваемый код все равно может быть прав.

Таблица №1 – Основные характеристики детектора FB.IS\_INCONSISTENT\_SYNC

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Concurrency Vulnerability |
| CWE | CWE366 |
| CVE | --- |
| Возможные последствия | * Состояние гонки (race conditions) * Несогласованное состояние объекта * Трудноуловимые ошибки в многопоточной среде |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public static void incorrectTest() {  class Counter {  private int count = 0;    public synchronized void increment() {  count++;  }    // Некорректно: несинхронизированный getter  public int getCount() {  return count;  }  }    Counter counter = new Counter();  System.out.println("Inconsistent counter: " + counter.getCount());  } | public static void correctTest() {  class SafeCounter {  private int count = 0;    public synchronized void increment() {  count++;  }    // Корректно: синхронизированный getter  public synchronized int getCount() {  return count;  }  }    SafeCounter counter = new SafeCounter();  System.out.println("Consistent counter: " + counter.getCount());    // Альтернатива с volatile (если подходит для случая)  class VolatileCounter {  private volatile int count = 0;    public void increment() {  count++; // Неатомарно, но изменения видны сразу  }    public int getCount() {  return count;  }  }  } |