FB.WA\_AWAIT\_NOT\_IN\_LOOP

Детектор определяет ошибку, в ходе которой метод содержит вызов java.util.concurrent.await() (или его вариантов), который не находится в цикле. Если объект используется для нескольких условий, то условие, которого вызывающий объект намеревался ожидать, может оказаться не тем, которое произошло на самом деле.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Ожидание (await()) вне цикла может привести к пропуску уведомлений, если условие изменится до вызова метода. |
| Комментарий | Рекомендуется использовать await() внутри цикла, проверяя условие, на которое ожидается изменение. Это гарантирует, что поток возобновит выполнение только после наступления нужного состояния, исключая пропуск уведомлений. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  class Test {  private final Lock lock = new ReentrantLock();  private final Condition condition = lock.newCondition();  public void waitForCondition() throws InterruptedException {  lock.lock();  try {  condition.await(); // await() вне цикла  System.out.println("Condition met, proceeding...");  } finally {  lock.unlock();  }  }  }  } | public static void correctTest() {  class Test {  private final Lock lock = new ReentrantLock();  private final Condition condition = lock.newCondition();  private boolean ready = false;  public void waitForCondition() throws InterruptedException {  lock.lock();  try {  while (!ready) { // проверка условия в цикле  condition.await();  }  System.out.println("Condition met, proceeding...");  } finally {  lock.unlock();  }  }  }  } |