FB.JML\_JSR166\_CALLING\_WAIT\_RATHER\_THAN\_AWAIT

Этот метод вызывает wait(), notify() или notifyAll() для объекта, который также предоставляет методы await(), signal(), signalAll() (например, объекты util.concurrent Condition). Вероятно, это не то, что нужно, и даже если вы этого хотите, следует подумать об изменении вашего дизайна, поскольку другие разработчики найдут его исключительно запутанным.

Таблица №1 – Основные характеристики детектора FB.JML\_JSR166\_CALLING\_WAIT\_RATHER\_THAN\_AWAIT

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Concurrency API Misuse |
| CWE | --- |
| CVE | --- |
| Возможные последствия | Неправильная работа блокировок из пакета java.util.concurrent.locks; возможность deadlock-ов и livelock-ов; потеря уведомлений (missed signals); нарушение контрактов использования Condition |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public static void incorrectTest() {  TaskProcessorIncorrect processor = new TaskProcessorIncorrect();    new Thread(() -> {  try {  processor.waitForCondition();  System.out.println("Incorrect: Condition met (potentially wrong)");  } catch (InterruptedException e) {  Thread.currentThread().interrupt();  }  }).start();  } | public static void correctTest() {  TaskProcessorCorrect processor = new TaskProcessorCorrect();    new Thread(() -> {  try {  processor.waitForCondition();  System.out.println("Correct: Condition properly met");  } catch (InterruptedException e) {  Thread.currentThread().interrupt();  }  }).start();  new Thread(() -> {  processor.signalCondition();  }).start();  } |