FB.WA\_NOT\_IN\_LOOP

Ошибка "Wait not in loop" или "Wait not in loop in" указывает на то, что операция wait() используется внутри кода, но без обертывания её в цикл. Обычно, использование wait() должно быть в цикле, чтобы предотвратить "ложные пробуждения" (spurious wake-ups) и правильно обрабатывать условия, по которым поток должен продолжить выполнение.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Использование wait() вне цикла может привести к некорректному возобновлению потока из-за ложных пробуждений (spurious wake-ups) или уведомлений по неподходящему условию. |
| Комментарий | Рекомендуется всегда вызывать wait() внутри цикла с проверкой условия ожидания. Это гарантирует, что поток возобновит выполнение только тогда, когда действительно наступит требуемое состояние. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| class SharedResourceIncorrect {  private boolean ready = false;  public synchronized void waitForReady() throws InterruptedException {  wait(); // вызов wait() без цикла  System.out.println("Condition met, proceeding...");  }  public synchronized void setReady() {  ready = true;  notifyAll();  }  } | class SharedResourceCorrect {  private boolean ready = false;  public synchronized void waitForReady() throws InterruptedException {  while (!ready) { // проверка условия в цикле  wait();  }  System.out.println("Condition met, proceeding...");  }  public synchronized void setReady() {  ready = true;  notifyAll();  }  } |

Методы устранения уязвимости при срабатывании детектора:

1. Оберните вызов метода wait() в цикл while, который проверяет условие, необходимое для продолжения работы потока. Это ключевой шаг для обработки ложных пробуждений.
2. Создайте булеву переменную (или другое состояние), представляющее условие, наступления которого ожидает поток. Доступ к этой переменной должен быть синхронизирован с использованием того же объекта блокировки, на котором вызывается wait().
3. Вызывайте wait() только тогда, когда условие не выполнено (обычно while (!condition)). Цикл будет продолжаться до тех пор, пока условие не станет истинным.
4. В потоке, который должен сигнализировать об изменении условия, после изменения состояния переменной условия вызовите методы notify() или notifyAll() для того же объекта блокировки. Эти вызовы также должны находиться внутри синхронизированного блока, использующего ту же блокировку.
5. Используйте notifyAll() вместо notify(), если на одном и том же мониторе могут ожидать несколько потоков, и вы хотите разбудить их все для повторной проверки условия. Это помогает избежать "потерянных уведомлений", когда нужный поток может не получить notify().
6. Помните, что поток может проснуться от wait() без вызова notify() или notifyAll() (ложное пробуждение). Именно поэтому проверка условия в цикле while необходима.
7. Убедитесь, что доступ к переменной условия и вызовы wait() и notify()/notifyAll() защищены одним и тем же монитором (через synchronized блок или метод).