FB.UW\_UNCOND\_WAIT

Ошибка "unconditional wait" (безусловное ожидание) указывает на то, что в вашем коде присутствует операция ожидания (например, использование метода wait() в Java), которая выполняется без проверки какого-либо условия. Это может привести к блокировке (замораживанию) выполнения программы, если не выполнены определенные условия для продолжения работы.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Безусловное ожидание без проверки условия может привести к зависанию потока, если notify() или notifyAll() никогда не будут вызваны. |
| Комментарий | Рекомендуется использовать wait() внутри цикла с проверкой условия, чтобы избежать зависания потока. Например, применяя while (condition) { wait(); }, можно гарантировать, что поток продолжит выполнение только при выполнении необходимых условий. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() throws InterruptedException {  class Test {  private final Object lock = new Object();  public void work() throws InterruptedException {  synchronized (lock) {  // ожидание без условия  lock.wait();  }  }  }  Test test = new Test();  test.work(); // Ошибочный вызов (может зависнуть)  } | public static void correctTest() throws InterruptedException {  class Test {  private final Object lock = new Object();  private boolean ready = false;  public void doWork() throws InterruptedException {  synchronized (lock) {  // Проверка условия перед ожиданием  while (!ready) {  lock.wait();  }  // ...  }  }  public void markReady() {  synchronized (lock) {  ready = true;  lock.notifyAll();  }  }  }  Test test = new Test();  Thread worker = new Thread(() -> {  try {  test.doWork();  } catch (InterruptedException e) {  Thread.currentThread().interrupt();  }  });  worker.start();  Thread.sleep(1000);  test.markReady(); // wait() корректно завершится  worker.join();  } |

Методы устранения уязвимости при срабатывании детектора:

1. Оберните вызов wait() в цикл while, который проверяет условие, необходимое для продолжения работы. Это гарантирует, что поток проснется только при выполнении нужного условия.
2. Создайте булеву переменную (или другое состояние), представляющее условие ожидания. Этот флаг должен изменяться потоком, который вызывает notify() или notifyAll().
3. Вызывайте wait() только тогда, когда условие не выполнено. Цикл while должен продолжаться до тех пор, пока условие не станет истинным.
4. В потоке, который должен разблокировать ожидающий поток, после изменения условия вызовите notify() или notifyAll() для того же объекта блокировки. notifyAll() обычно предпочтительнее, чтобы избежать "потерянных уведомлений".
5. Убедитесь, что доступ к переменной условия и вызовы wait()/notify()/notifyAll() находятся в одном и том же синхронизированном блоке, использующем один и тот же объект блокировки.
6. Рассмотрите использование классов из пакета java.util.concurrent (например, CountDownLatch, CyclicBarrier, Condition) для более сложных сценариев синхронизации. Эти утилиты предоставляют более надежные и гибкие механизмы ожидания и оповещения.
7. Избегайте использования Thread.sleep() для ожидания наступления условия. wait() освобождает блокировку, позволяя другим потокам работать, в то время как sleep() блокирует текущий поток, удерживая блокировку.