FB.MWN\_MISMATCHED\_WAIT

Детектор определяет ошибку, в ходе которой метод вызывает Object.wait() без явной блокировки объекта. Вызов wait() без удержания блокировки приведет к созданию исключения IllegalMonitorStateException.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Вызов Object.wait() без удержания соответствующей блокировки приведет к выбросу исключения IllegalMonitorStateException, что может вызвать некорректное поведение программы и нарушение логики многопоточного взаимодействия. |
| Комментарий | Для корректного использования метода wait() необходимо вызывать его внутри блока synchronized, удерживающего монитор объекта, чтобы избежать исключений и обеспечить безопасную синхронизацию потоков. |

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 method() throws InterruptedException {  // без захвата блокировки  lock.wait();  }  }  new Test().method();  } | public static void correctTest() throws InterruptedException {  class Test {  private final Object lock = new Object();  public void method() throws InterruptedException {  synchronized (lock) {  lock.wait();  }  }  }  new Test().method();  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Оберните вызов wait() в synchronized блок:** Убедитесь, что вызов wait() происходит внутри блока synchronized, который синхронизирован на том же объекте, для которого вызывается wait().
2. **Проверьте владение монитором:** Перед вызовом wait() убедитесь, что текущий поток владеет монитором соответствующего объекта.
3. **Используйте wait() внутри цикла:** Всегда вызывайте wait() внутри цикла, который проверяет условие, для которого поток ожидает. Это необходимо для обработки "ложных пробуждений", когда поток может быть разбужен без фактического выполнения условия.
4. **Используйте notify()/notifyAll() после изменения условия:** Поток, который изменяет условие, которого ожидают другие потоки, должен вызвать notify() или notifyAll() (также внутри синхронизированного блока на том же объекте).
5. **Рассмотрите более высокоуровневые средства синхронизации:** Пакет java.util.concurrent предоставляет более гибкие и безопасные средства синхронизации, такие как Lock, Condition, CountDownLatch, Semaphore и другие, которые часто предпочтительнее низкоуровневых wait()/notify()/notifyAll(). Рассмотрите возможность их использования вместо явной работы с мониторами объектов.