FB.SP\_SPIN\_ON\_FIELD

Детектор определяет ошибку, в ходе которой метод вращается в цикле, который считывает поле. Компилятор может законно вывести чтение из цикла, превратив код в бесконечный цикл. Класс следует изменить, чтобы он использовал правильную синхронизацию (включая вызовы ожидания и уведомления).

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Вращение в цикле без синхронизации может привести к тому, что компилятор оптимизирует код, кэшируя значение поля и создавая бесконечный цикл, который никогда не завершится. |
| Комментарий | Для корректной работы многопоточного кода следует использовать механизмы синхронизации, такие как volatile для переменной, synchronized блоки или методы wait() и notify(), чтобы гарантировать актуальность значения при каждом чтении. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  class Test {  private boolean flag = false;  public void method() {  // Спин-лок: цикл, который проверяет поле без синхронизации  while (!flag) {  // Ожидаем изменения флага  }  }  }  } | public static void correctTest() throws InterruptedException {  class Test {  private boolean flag = false;  public void method() throws InterruptedException {  synchronized (this) {  while (!flag) {  // Ожидаем, пока флаг не станет true  wait();  }  }  }  public void setFlagTrue() {  synchronized (this) {  flag = true;  // Уведомляем, что флаг изменился  notify();  }  }  }  } |