FB.NN\_NAKED\_NOTIFY

Детектор определяет ошибку, в ходе которой вызов notify() или notifyAll() был выполнен без какого-либо (кажущегося) сопутствующего изменения состояния изменяемого объекта. Обычно вызов метода уведомления на мониторе выполняется потому, что какое-то условие, которого ожидает другой поток, стало истинным. Однако чтобы условие было значимым, оно должно включать объект кучи, видимый обоим потокам.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Вызов notify() или notifyAll() без сопутствующего изменения состояния объекта может привести к неопределенному поведению программы, когда ожидающий поток пробуждается без изменения условий, необходимых для его корректного выполнения. Это может вызвать зависание, потерю уведомлений или некорректную обработку данных. |
| Комментарий | Перед вызовом notify() или notifyAll() следует убедиться, что изменено состояние объекта, определяющее условие пробуждения ожидающих потоков. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  class Test {  private final Object lock = new Object();  public void method() {  synchronized (lock) {  lock.notify();  }  }  public void waitForCondition() throws InterruptedException {  synchronized (lock) {  lock.wait();  }  }  }  Test test = new Test();  new Thread(() -> {  try {  test.waitForCondition();  } catch (InterruptedException e) {  Thread.currentThread().interrupt();  }  }).start();  try {  Thread.sleep(100);  } catch (InterruptedException ignored) {}  // Вызываем notify() без изменения состояния  test.method();  } | public static void correctTest() {  class Test {  private final Object lock = new Object();  private boolean condition = false;  public void method() {  synchronized (lock) {  // Изменяем состояние перед notify  condition = true;  lock.notify();  }  }  public void waitForCondition() throws InterruptedException {  synchronized (lock) {  while (!condition) {  lock.wait();  }  // ...  }  }  }  Test test = new Test();  new Thread(() -> {  try {  test.waitForCondition();  } catch (InterruptedException e) {  Thread.currentThread().interrupt();  }  }).start();  try {  Thread.sleep(100);  } catch (InterruptedException ignored) {}  // Вызываем notify() уже с изменённым состоянием  test.method();  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Перед вызовом notify() или notifyAll() изменяйте состояние разделяемых переменных:** Убедитесь, что вы устанавливаете в true условие, которого ожидают другие потоки.
2. **Используйте цикл while для проверки условия после пробуждения:** Ожидающие потоки должны проверять условие в цикле while после выхода из wait(), чтобы убедиться, что они не проснулись преждевременно (из-за "ложных пробуждений" или уведомления без изменения состояния).
3. **Следуйте стандартному паттерну wait()/notify():** Синхронизация, изменение состояния и вызов notify() должны быть тесно связаны.
4. **Рассмотрите использование java.util.concurrent:** Для более сложных сценариев синхронизации рассмотрите использование классов из пакета java.util.concurrent, таких как Condition (связанные с Lock), которые предоставляют более явное управление условиями ожидания и оповещения.