FB.UL\_UNRELEASED\_LOCK

Детектор определяет ошибку, в ходе которой метод получает блокировку JSR-166 (java.util.concurrent), но не снимает ее на всех путях выхода из метода.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Если блокировка не снимается на всех возможных путях выхода из метода, это может привести к взаимоблокировке (deadlock) или снижению производительности из-за блокирования других потоков, ожидающих освобождения ресурса. |
| Комментарий | Рекомендуется убедиться, что блокировка освобождается во всех возможных сценариях завершения метода, включая обработку исключений. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  class Test {  private final Lock lock = new ReentrantLock();  public void work(boolean fail) {  lock.lock();  try {  // ...  if (fail) {  throw new RuntimeException();  }  } finally {  // Блокировка освобождается не во всех случаях  if (fail) {  lock.unlock();  }  }  }  }  Test test = new Test();  test.work(false);  } | public static void correctTest() {  class Test {  private final Lock lock = new ReentrantLock();  public void work(boolean fail) {  lock.lock();  try {  // ...  if (fail) {  throw new RuntimeException();  }  } finally {  // Блокировка всегда освобождается  lock.unlock();  }  }  }  Test test = new Test();  test.work(false);  } |

Методы устранения уязвимости при срабатывании детектора:

1. Оберните код, который работает с защищенным ресурсом, в блок try.
2. Непосредственно после блока try добавьте блок finally.
3. Внутри блока finally поместите вызов метода unlock() для объекта блокировки.