FB.JLM\_JSR166\_UTILCONCURRENT\_MONITORENTER

Детектор определяет ошибку, в ходе которой метод выполняет синхронизацию объекта, который является экземпляром класса из пакета java.util.concurrent (или его подклассов). Экземпляры этих классов имеют свои собственные механизмы управления параллелизмом, которые ортогональны синхронизации, обеспечиваемой ключевым словом Java Synchronized. Например, синхронизация по AtomicBoolean не помешает другим потокам изменять AtomicBoolean.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Синхронизация на объектах из пакета java.util.concurrent (например, AtomicBoolean, ReentrantLock и т. д.) может привести к некорректному использованию механизмов синхронизации. Эти классы уже имеют свои собственные встроенные механизмы управления параллелизмом, такие как атомарные операции или явные блокировки, которые работают независимо от механизма синхронизации synchronized. |
| Комментарий | При работе с классами из java.util.concurrent следует использовать их собственные методы синхронизации и блокировки, а не применять синхронизацию с помощью ключевого слова synchronized. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  class Test {  private final AtomicBoolean lock = new AtomicBoolean(false);  public void method() {  // синхронизация на AtomicBoolean  synchronized (lock) {  System.out.println("Executing critical section");  }  }  }  } | public static void correctTest() {  class Test {  private final AtomicBoolean lock = new AtomicBoolean(false);  public void method() {  // Не используем синхронизацию, так как AtomicBoolean уже управляет параллелизмом  if (lock.compareAndSet(false, true)) {  try {  // ...  System.out.println("Executing critical section");  } finally {  lock.set(false); // Обновление по завершении  }  }  }  }  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Удалите блок synchronized:** Синхронизация на объектах java.util.concurrent обычно не нужна и может быть удалена.
2. **Используйте специфичные методы класса:** Вместо synchronized, используйте методы, предоставляемые соответствующим классом из java.util.concurrent для управления параллелизмом (например, compareAndSet(), lock(), acquire(), await(), signal(), countDown() и т.д.).
3. **Пересмотрите логику синхронизации:** Убедитесь, что вы правильно понимаете, как работает класс из java.util.concurrent, который вы используете, и применяете его методы синхронизации в соответствии с его назначением.
4. **Обратитесь к документации:** Изучите документацию по классам из java.util.concurrent, чтобы понять их механизмы синхронизации и правильные способы их использования.