FB.DL\_SYNCHRONIZATION\_ON\_UNSHARED\_BOXED\_PRIMITIVE

Детектор определяет ошибку, в ходе которой код синхронизируется с очевидно необщим упакованным примитивом, таким как Integer.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Синхронизация на объекте-обертке примитивного типа, даже если он не является общим, может быть неэффективной и избыточной. В случае, если объект не разделяется между потоками, синхронизация на нем может привести к ненужным затратам на блокировки, ухудшая производительность программы без реальной необходимости в синхронизации. |
| Комментарий | Для синхронизации следует использовать объекты-мониторы, специально предназначенные для этих целей (например, private final Object lock = new Object();), а не объекты-обертки примитивных типов, чтобы избежать лишней нагрузки и обеспечить более эффективную работу многопоточного приложения. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  class Test {  // значение переменной может быть уникальным для каждого экземпляра  private Integer lock = 42;  public void method() {  // синхронизация на переменной Integer  synchronized (lock) {  System.out.println("Executing critical section");  }  }  }  } | public static void correctTest() {  class Test {  // независимый объект  private final Object lock = new Object();  public void method() {  // Гарантированно корректная синхронизация  synchronized (lock) {  System.out.println("Executing critical section");  }  }  }  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Создайте приватный объект-монитор:** Вместо использования объектов-оберток примитивных типов для синхронизации, создайте приватный экземпляр Object специально для этой цели: private final Object lock = new Object();.
2. **Используйте созданный объект в synchronized блоках:** Замените синхронизацию на объект-обертку на синхронизацию по созданному приватному объекту-монитору.
3. **Избегайте использования объектов-оберток для синхронизации:** Даже если вам кажется, что объект-обертка не является общим, использование Object для синхронизации является более стандартной и понятной практикой.
4. **Проверьте необходимость синхронизации:** Если объект действительно не разделяется между потоками, возможно, синхронизация вообще не нужна. Удалите избыточные synchronized блоки или методы.
5. **Рассмотрите альтернативы синхронизации:** В некоторых случаях для обеспечения потокобезопасности могут быть использованы более эффективные механизмы, такие как атомарные переменные (java.util.concurrent.atomic) или блокировки из пакета java.util.concurrent.locks.