FB.ESYNC\_EMPTY\_SYNC

Детектор определяет ошибку, в ходе которой код содержит пустой синхронизированный блок.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | CWE585 |
| CVE | — |
| Возможные последствия | Пустой синхронизированный блок не выполняет никакой полезной работы, но все равно вызывает накладные расходы, связанные с управлением блокировками. Это может привести к ухудшению производительности, увеличению сложности кода и путанице, так как неясно, почему синхронизация была использована. |
| Комментарий | Если синхронизация не требуется, следует удалить пустой синхронизированный блок, чтобы избежать лишних затрат. В противном случае синхронизацию следует применять только там, где она необходима для обеспечения корректности многопоточной работы. |

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) {  // Пустой синхронизированный блок  }  }  }  } | public static void correctTest() {  class Test {  private final Object lock = new Object();  public void method() {  // Синхронизация с реальной работой внутри  synchronized (lock) {  System.out.println("Executing critical section");  }  }  }  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Удалите пустой блок synchronized:** Если внутри блока нет никакого кода, который требует синхронизации для обеспечения потокобезопасности, просто удалите весь блок synchronized и окружающие фигурные скобки.
2. **Переместите защищаемый код внутрь блока:** Если пустой блок synchronized был создан по ошибке и предназначался для защиты определенного участка кода, переместите этот код внутрь синхронизированного блока.
3. **Проверьте логику многопоточности:** Убедитесь, что удаление или перемещение синхронизированного блока не нарушит потокобезопасность приложения. Проанализируйте, какие ресурсы являются общими и требуют защиты от одновременного доступа.
4. **Используйте более высокоуровневые средства синхронизации (при необходимости):** В некоторых случаях вместо низкоуровневой синхронизации с помощью synchronized могут быть более подходящими высокоуровневые средства из пакета java.util.concurrent, такие как Lock, Semaphore, CountDownLatch и другие.