FB.LI\_LAZY\_INIT\_STATIC

Детектор определяет ошибку, в ходе которой метод содержит несинхронизированную ленивую инициализацию энергонезависимого статического поля. Поскольку компилятор или процессор может изменить порядок инструкций, потоки не гарантируют, что увидят полностью инициализированный объект, если метод может быть вызван несколькими потоками. Чтобы исправить проблему, вы можете сделать поле нестабильным.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | CWE543 |
| CVE | — |
| Возможные последствия | Несинхронизированная ленивую инициализация статического поля может привести к состояниям гонки, где несколько потоков одновременно попытаются инициализировать поле, что приведет к неполной или некорректной инициализации. |
| Комментарий | Для безопасной ленивой инициализации статических полей следует использовать синхронизацию (например, через synchronized или volatile), чтобы гарантировать, что инициализация происходит атомарно и потоки видят корректно инициализированный объект. Альтернативно, можно использовать конструкции вроде Double-Checked Locking или Enum Singleton для обеспечения потокобезопасности. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  class Example {  // ...  }  class Test {  private static Example instance;  public static Example getInstance() {  // ленивая инициализация без синхронизации  if (instance == null) {  instance = new Example();  }  return instance;  }  }  } | public static void correctTest() {  class Example {  // ...  }  class Test {  // гарантируем правильное обновление состояния  private static volatile Example instance;  public static Example getInstance() {  // двойная проверка внутри и снаружи блока  // для минимизации количества блокировок  if (instance == null) {  synchronized (Test.class) {  if (instance == null) {  instance = new Example();  }  }  }  return instance;  }  }  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Сделайте поле volatile:** Для простых объектов, где инициализация состоит из одной атомарной операции присваивания, объявление поля как volatile может решить проблему видимости. Однако для более сложных объектов, где конструктор выполняет несколько шагов, этого может быть недостаточно.
2. **Синхронизируйте метод доступа:** Объявите метод, предоставляющий доступ к экземпляру, как synchronized. Это гарантирует, что только один поток сможет выполнить инициализацию.
3. **Используйте Double-Checked Locking:** Реализуйте паттерн двойной проверки с использованием блока synchronized и проверки instance == null как до, так и внутри синхронизированного блока. Поле должно быть объявлено как volatile.
4. **Используйте статический инициализатор:** Инициализируйте статическое поле непосредственно при объявлении класса. Статическая инициализация выполняется потокобезопасным способом.
5. **Используйте Enum Singleton:** Если вам нужен Singleton, использование enum с одним экземпляром является простым и потокобезопасным решением.