FB.SIC\_INNER\_SHOULD\_BE\_STATIC

Детектор определяет ошибку, в ходе которой класс является внутренним классом, но не использует встроенную ссылку на объект, который его создал. Эта ссылка увеличивает размер экземпляров класса и может сохранять ссылку на объект-создатель дольше, чем необходимо. Если возможно, класс следует сделать статическим.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Необоснованное использование нестатического (не static) внутреннего класса приводит к тому, что он неявно хранит ссылку на экземпляр внешнего класса. Это увеличивает потребление памяти, усложняет управление временем жизни объектов и может привести к утечкам памяти, если внутренний класс используется асинхронно или передаётся за пределы внешнего объекта. |
| Комментарий | Если внутренний класс не использует поля и методы внешнего класса, его следует объявить static. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  class Outer {  class InnerClass { // Нестатический внутренний класс  public void doSomething() {  System.out.println("...");  }  }  }  Outer outer = new Outer();  // Требуется ссылка на внешний объект  Outer.InnerClass inner = outer.new InnerClass();  inner.doSomething();  } | public static void correctTest() {  class Outer {  // Статический вложенный класс  static class InnerClass {  public void doSomething() {  System.out.println("...");  }  }  }  // Не требует ссылки на внешний объект  Outer.InnerClass inner = new Outer.InnerClass();  inner.doSomething();  } |

Методы устранения уязвимости при срабатывании детектора:

1. Проанализируйте код внутреннего класса. Убедитесь, что он действительно не обращается к каким-либо нестатическим полям или методам внешнего класса. Если обращение есть, класс не может быть статическим.
2. Если внутренний класс не использует члены внешнего класса, добавьте ключевое слово static в объявление внутреннего класса. Это превратит его в статический вложенный класс.
3. Пересмотрите способ создания экземпляров внутреннего класса. Для статических вложенных классов не требуется экземпляр внешнего класса для их создания. Экземпляры создаются как Outer.Inner inner = new Outer.Inner(); вместо Outer outer = new Outer(); Outer.Inner inner = outer.new Inner();.
4. Убедитесь, что после преобразования внутреннего класса в статический, код, использующий этот класс, продолжает работать корректно. Поскольку статическим вложенным классам не передается неявная ссылка на внешний объект, любой код внутри внутреннего класса, который ранее мог предполагать наличие этой ссылки, необходимо будет пересмотреть.
5. Если внутренний класс должен иметь доступ к состоянию внешнего класса, рассмотрите другие способы передачи необходимой информации (например, через параметры конструктора внутреннего класса). Это может позволить сделать внутренний класс статическим и избежать проблем с управлением памятью.
6. В случаях, когда внутренний класс тесно связан с конкретным экземпляром внешнего класса и использует его состояние, оставьте его нестатическим, но убедитесь, что время жизни экземпляров внутреннего класса контролируется должным образом, чтобы избежать утечек памяти.