FB.SIC\_INNER\_SHOULD\_BE\_STATIC\_NEEDS\_THIS

Этот класс является внутренним классом, но он не использует встроенную ссылку на объект, который его создал, за исключением случаев создания внутреннего объекта. Эта ссылка увеличивает размер экземпляров класса и может сохранять ссылку на объект-создатель дольше, чем необходимо. Если возможно, класс следует превратить в статический внутренний класс. Поскольку ссылка на внешний объект требуется во время создания внутреннего экземпляра, внутренний класс необходимо будет реорганизовать, чтобы передать ссылку на внешний экземпляр конструктору внутреннего класса.

Таблица №1 – Основные характеристики детектора FB.SIC\_INNER\_SHOULD\_BE\_STATIC\_NEEDS\_THIS

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Memory Leak Potential |
| CWE | --- |
| CVE | --- |
| Возможные последствия | Неявное удержание ссылки на внешний класс; утечка памяти при длительном существовании внутреннего класса; избыточное потребление памяти |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public static void incorrectTest() {  class OuterClass {  private String data = "Data";  void process() {  // Некорректно: нестатический внутренний класс  class InnerClass {  void print() {  System.out.println(OuterClass.this.data); // Явная ссылка на внешний класс  }  }  new InnerClass().print();  }  }  new OuterClass().process();  } | public static void correctTest() {  class OuterClass {  private String data = "Data";  void process() {  // Корректно: статический вложенный класс с явной передачей данных  class StaticNested {  final String message;  StaticNested(String msg) {  this.message = msg;  }  void print() {  System.out.println(message);  }  }  new StaticNested(data).print();  }  }  new OuterClass().process();  } |