FB.SIC\_INNER\_SHOULD\_BE\_STATIC\_ANON

Этот 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 превратить в статический внутренний класс. Поскольку анонимные внутренние классы нельзя пометить как статические, для этого потребуется рефакторинг внутреннего класса, чтобы он стал именованным внутренним классом.

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

|  |  |
| --- | --- |
| 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 использовании анонимного 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 = "SensitiveData";  void createAnonymous() {  // Некорректно: нестатический анонимный класс  Runnable r = new Runnable() {  @Override  public void run() {  System.out.println(data); // Неявная ссылка на внешний класс  }  };  r.run();  }  }  new OuterClass().createAnonymous();  } | public static void correctTest() {  class OuterClass {  private String data = "SensitiveData";  void createAnonymous() {  // Корректно: использование лямбды (неявно static)  Runnable r = () -> System.out.println(data);  r.run();    // Или вынос в статический вложенный класс  class StaticNested implements Runnable {  final String message;    StaticNested(String msg) {  this.message = msg;  }    @Override  public void run() {  System.out.println(message);  }  }  new StaticNested(data).run();  }  }  new OuterClass().createAnonymous();  } |