FB.FI\_PUBLIC\_SHOULD\_BE\_PROTECTED

Детектор определяет ошибку, в ходе которой Метод Finalize() класса должен иметь защищенный, а не общедоступный доступ.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | CWE583 |
| CVE | — |
| Возможные последствия | Если метод finalize() объявлен с модификатором public, это может привести к его непреднамеренному вызову извне класса, что нарушает инкапсуляцию и может вызвать некорректное завершение объекта, потенциально приводя к утечкам памяти или неожиданному поведению приложения. |
| Комментарий | Метод finalize() следует объявлять с модификатором protected, чтобы ограничить его вызов системой сборки мусора и предотвратить доступ из внешнего кода. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  class Test {  @Override  public void finalize() throws Throwable {  try {  System.out.println("Finalize method called");  } finally {  super.finalize();  }  }  }  } | public static void correctTest() {  class Test {  @Override  protected void finalize() throws Throwable {  try {  System.out.println("Finalize method called");  } finally {  super.finalize();  }  }  }  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Измените модификатор доступа:** Замените ключевое слово public на protected в объявлении метода finalize().
2. **Проверьте иерархию классов:** Убедитесь, что ни один из суперклассов не предоставляет public метод finalize(), который может быть унаследован.
3. **Рассмотрите альтернативы finalize():** Метод finalize() имеет ряд недостатков и его использование не рекомендуется. Рассмотрите более надежные способы управления ресурсами, такие как:
   * Использование блоков try-with-resources для автоматического закрытия ресурсов, реализующих интерфейс AutoCloseable.
   * Предоставление явного метода close() или dispose() для освобождения ресурсов, который должен вызываться клиентами объекта.
   * Использование объектов-наблюдателей (например, PhantomReference) в сочетании с очередью ссылок для отслеживания объектов, подлежащих удалению, и выполнения очистки в выделенном потоке.