FB.NP\_NULL\_ON\_SOME\_PATH

Детектор выявляет потенциальные NullPointerException, которые могут возникнуть при разыменовании переменной, если выполнится определенная ветвь кода, где эта переменная гарантированно будет null. Хотя в некоторых случаях такая ветвь может быть недостижимой (например, из-за логики программы), SpotBugs не может это подтвердить, поэтому рекомендуется явно обработать такие сценарии.

Таблица №1 – Основные характеристики детектора FB.NP\_NULL\_ON\_SOME\_PATH

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Null pointer dereference |
| CWE | CWE-476 |
| CVE | --- |
| Возможные последствия | Возможное возникновение NullPointerException при выполнении определенных ветвей кода |
| Комментарий (опционально) | Проверьте логику ветвления и добавьте обработку null. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public static void incorrectExample(String input) {  if (input != null) {  System.out.println("Длина строки: " + input.length());  } else {  // Потенциальный NPE, если выполнится эта ветвь  System.out.println("Длина строки: " + input.length());  }  } | public static void correctExample(String input) {  if (input != null) {  System.out.println("Длина строки: " + input.length());  } else {  // Безопасная обработка null  System.out.println("Ошибка: входная строка null");  }  } |

Методы устранения уязвимости при срабатывании детектора

1. Проверка на null: Всегда проверяйте переменные на null перед разыменованием.
2. Обработка null-значений: Замените потенциально опасные операции безопасной логикой (например, возврат default-значения или сообщения об ошибке).
3. Удаление недостижимого кода: Если ветвь с разыменованием null действительно недостижима, удалите ее для ясности.