FB.NP\_NONNULL\_PARAM\_VIOLATION

Детектор определяет ошибку, в ходе которой метод передает нулевое значение в качестве параметра метода, который не должен быть нулевым. Либо этот параметр явно помечен как @Nonnull, либо анализ определил, что этот параметр всегда разыменовывается.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | CWE-476 |
| CVE | — |
| Возможные последствия | NullPointerException, сбой программы, непредсказуемое поведение |
| Комментарий | Ошибка возникает, когда метод передает null в параметр, который явно **не должен** быть null. Чаще всего это параметр, аннотированный @Nonnull, или анализ кода определил, что он всегда разыменовывается. Чтобы избежать ошибки, следует либо **проверять аргумент перед вызовом**, либо использовать Objects.requireNonNull() внутри метода для явного выброса исключения с пояснением. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest(String str) {  class Example {  public static void printStr(@Nonnull String str) {  System.out.println(str);  }  }  // Возможно str = null  Example.printStr(str);  } | public static void correctTest(String str) {  class Example {  public static void printStr(@Nonnull String str) {  System.out.println(str);  }  }  if (Objects.nonNull(str)) {  Example.printStr(str);  }  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Проверяйте аргументы на null перед вызовом метода, который не должен принимать null.**
2. **Используйте java.util.Objects.requireNonNull() внутри метода, который не должен принимать null аргументы, чтобы явно выбросить исключение при получении null.**
3. **Пересмотрите логику, приводящую к передаче null.** Возможно, в коде есть ошибка, из-за которой null значение попадает туда, где его быть не должно.
4. **Рассмотрите возможность использования Optional (начиная с Java 8) для явного обозначения возможности отсутствия значения.** Если параметр может быть необязательным, использование Optional<String> вместо @Nonnull String может сделать код более выразительным и безопасным.