FB.NP\_NONNULL\_RETURN\_VIOLATION

Детектор определяет ошибку, в ходе которой метод может возвращать нулевое значение, но объявлен метод (или метод суперкласса, который он переопределяет) возвращающий @Nonnull.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | CWE-476 |
| CVE | — |
| Возможные последствия | NullPointerException, некорректная работа методов, поломка логики программы, нарушение контрактов аннотаций |
| Комментарий | Если метод аннотирован @Nonnull, но может вернуть null, это приведет к непредсказуемому поведению. Необходимо использовать Optional<T> или возвращать значение-заглушку. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  class Example {  @Nonnull  public static String getMessage() {  return null;  }  }  String message = Example.getMessage();  // NullPointerException:  System.out.println(message.length());  } | public static void correctTest() {  class Example {  @Nonnull  public static String getMessage() {  return "string";  }  }  String message = Example.getMessage();  System.out.println(message.length());  } |

Методы устранения уязвимости при срабатывании детектора:

1. Убедитесь, что метод, аннотированный @Nonnull, никогда не возвращает null ни при каких обстоятельствах.
2. Если метод может не иметь значения для возврата, используйте Optional<T> в качестве возвращаемого типа.
3. Рассмотрите возможность возвращения значения-заглушки, если это уместно и четко задокументировано.
4. Если отсутствие значения является исключительной ситуацией, выбросьте соответствующее исключение.
5. Пересмотрите логику метода, чтобы понять, почему может возникнуть ситуация, когда возвращается null, и найдите способ избежать этого или обработать ее более явно.