FB.NP\_NULL\_PARAM\_DEREF\_ALL\_TARGETS\_DANGEROUS

Значение, возможно, нулевое, передается на сайте вызова, где все известные целевые методы требуют, чтобы параметр был ненулевым. Либо параметр помечен как параметр, который всегда должен быть ненулевым, либо анализ показал, что он всегда будет разыменован.

Таблица №1 – Основные характеристики детектора FB.NP\_NULL\_PARAM\_DEREF\_ALL\_TARGETS\_DANGEROUS

|  |  |
| --- | --- |
| 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 incorrectUsage() {  String potentiallyNullValue = getPotentiallyNullValue();  // Метод требует ненулевого значения, но передается возможный null  dangerousMethod(potentiallyNullValue);  } | public static void correctUsage() {  String potentiallyNullValue = getPotentiallyNullValue();  // Проверка на null перед вызовом метода  if (potentiallyNullValue != null) {  dangerousMethod(potentiallyNullValue);  } else {  System.out.println("Ошибка: параметр равен 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.