DEREF\_OF\_NULL.ANNOT.STRICT

Этот детектор подобен DEREF\_OF\_NULL.ANNOT, но он используется в более строгом режиме. Он будет срабатывать на любое разыменование указателя, который был аннотирован как @Nullable или аналогично, даже если перед этим была проверка на null.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | Normal |
| Надежность | VeryHigh |
| Источник | SvEng |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Может указывать на потенциально небезопасные конструкции, которые считаются корректными в обычном режиме, но в строгом режиме считаются рискованными из-за возможной некорректной интерпретации условий проверки. |
| Комментарий | Следует избегать неоднозначных логических путей и обеспечить явную безопасность разыменования, используя, при необходимости, дополнительные защитные конструкции или перепроверки. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest(@Nullable String input) {  if (input != null) {  // Строгий анализатор может всё равно посчитать это потенциально опасным разыменованием  System.out.println(input.length());  }  } | public static void correctTest(@Nullable String input) {  if (input == null) {  System.out.println("Строка равна null");  return;  }  // разыменование вне блока условия  System.out.println(input.length());  } |

Методы устранения уязвимости при срабатывании детектора:

1. Используйте ранний выход (return) сразу после проверки на null. Если значение оказалось null, немедленно завершите выполнение метода.
2. Разыменовывайте @Nullable переменную непосредственно после проверки на null и вне блока if. Это явно показывает, что разыменование происходит только в случае ненулевого значения.
3. Присвойте @Nullable переменную локальной final переменной сразу после проверки на null и используйте эту локальную переменную для разыменования. Это создает "снимок" значения, который строгий анализатор может счесть более безопасным.
4. Используйте Optional (в Java) или аналогичные типы. Работа с Optional явно требует проверки наличия значения перед его извлечением.