DEREF\_OF\_NULL.ANNOT

Этот детектор указывает на разыменование нулевого указателя, который был аннотирован как @Nullable или аналогично. Это означает, что разработчик явно отметил, что этот указатель может быть null, но в коде есть место, где он разыменовывается без предварительной проверки.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | Normal |
| Надежность | Unknown |
| Источник | SvEng |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Может привести к NullPointerException во время выполнения, несмотря на явно заданную возможность null в аннотации. |
| Комментарий | Необходимо строго соблюдать семантику аннотаций и всегда проверять @Nullable переменные перед разыменованием, чтобы обеспечить корректную и безопасную работу приложения. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  String text = getNullableText();  // возможен NPE  System.out.println(text.length());  }  @Nullable  public static String getNullableText() {  return Math.random() > 0.5 ? "Hello" : null;  } | public static void correctTest() {  String text = getNullableText();  if (text != null) {  System.out.println(text.length());  } else {  System.out.println("Текст отсутствует.");  }  }  @Nullable  public static String getNullableText() {  return Math.random() > 0.5 ? "Hello" : null;  } |

Методы устранения уязвимости при срабатывании детектора:

1. Добавьте явную проверку на null перед любым обращением к переменной, аннотированной как @Nullable (или аналогично). Используйте оператор if (variable != null) для безопасного доступа.
2. Если значение может быть null, предусмотрите альтернативный сценарий или обработку этого случая. Это может включать вывод сообщения, использование значения по умолчанию или пропуск операции.
3. Используйте Optional (в Java) или аналогичные типы для явного представления возможности отсутствия значения. Это заставляет явно обрабатывать случай null перед доступом к значению с помощью методов isPresent() и get() (или orElse(), orElseGet()).