DEREF\_OF\_NULL.RET.ANNOT

Детектор предназначен для ситуаций проверки возвращаемого значения функций на ноль. Он срабатывает, когда функция, которая может вернуть null (как указано в аннотации @Nullable или аналогичной), возвращает значение, которое затем разыменовывается без проверки на null.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | Normal |
| Надежность | High |
| Источник | SvEng |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Это может привести к NullPointerException, если возвращаемое значение null разыменовывается без должной проверки. |
| Комментарий | Необходимо всегда проверять возвращаемые значения, помеченные как @Nullable, на null перед их использованием, чтобы избежать непредсказуемых ошибок во время выполнения. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  String result = getNullableString();  // результат не проверяется на null перед разыменованием  System.out.println(result.length());  }  @Nullable  public static String getNullableString() {  return Math.random() > 0.5 ? "Hello" : null;  } | public static void correctTest() {  String result = getNullableString();  // Проверка на null перед разыменованием  if (result != null) {  System.out.println(result.length());  } else {  System.out.println("Результат равен null");  }  }  @Nullable  public static String getNullableString() {  return Math.random() > 0.5 ? "Hello" : null;  } |

Методы устранения уязвимости при срабатывании детектора:

1. Непосредственно после вызова функции, возвращающей @Nullable значение, добавьте явную проверку на null. Используйте оператор if (result != null) перед любым обращением к результату.
2. Если возвращаемое значение оказалось null, предусмотрите обработку этого случая. Это может включать вывод сообщения об ошибке, использование значения по умолчанию, выполнение альтернативного кода или возврат из текущего метода.
3. Используйте Optional (в Java) или аналогичные типы для обработки потенциально отсутствующих возвращаемых значений. Это заставляет явно проверять наличие значения перед его извлечением с помощью методов isPresent() и get() (или orElse(), orElseGet()).