DEREF\_OF\_NULL.STRICT

Детектор сигнализирует о возможном разыменовании нулевой ссылки. Детектор указывает, что ошибка может возникать даже с относительно малой вероятностью.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | Normal |
| Надежность | Unknown |
| Источник | SvEng |
| Тип ошибки | Ошибка разработчика |
| CWE | CWE476 |
| CVE | — |
| Возможные последствия | Разыменование null может привести к выбросу исключения NullPointerException, что приведет к сбою приложения в процессе его работы. |
| Комментарий | Необходимо внимательно проверять и обрабатывать значения, которые могут быть null, особенно в случае с параметрами или возвращаемыми значениями, чтобы избежать ошибок времени выполнения. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest(String maybeNull) {  // Разыменование без проверки на null  System.out.println(maybeNull.length());  } | public static void correctTest(String maybeNull) {  if (maybeNull != null) {  // Безопасное разыменование  System.out.println(maybeNull.length());  } else {  System.out.println("Значение отсутствует");  }  } |

Методы устранения уязвимости при срабатывании детектора:

1. Добавьте явную проверку на null перед любым обращением к переменной, которая может быть нулевой. 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, пропуск операции или возврат из метода.
3. Используйте Optional (в Java) или аналогичные типы для явного представления возможности отсутствия значения. Это заставляет явно проверять наличие значения перед его извлечением с помощью методов isPresent() и get() (или orElse(), orElseGet()).