DEREF\_OF\_NULL.UNCLEAR

Если вызываемая функция, в которой может происходить разыменование, неизвестна анализу, то потенциальные предупреждения выделяются в этот детектор.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | Normal |
| Надежность | Unknown |
| Источник | SvEng |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Такое разыменование может привести к исключению NullPointerException во время выполнения, особенно если вызываемый метод возвращает null, но это не может быть точно определено на этапе анализа. |
| Комментарий | Рекомендуется явно проверять возвращаемые значения на null или использовать аннотации (@Nullable, @NotNull) для повышения надежности анализа. |

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 непосредственно после вызова неизвестной функции. Используйте оператор if (result != null) перед любым обращением к result.
2. Если null является недопустимым результатом, выбросьте исключение (IllegalArgumentException, NullPointerException) при обнаружении null.
3. Используйте Optional (в Java) или аналогичные типы для обработки потенциально отсутствующих возвращаемых значений. Это заставляет явно проверять наличие значения перед его извлечением.