DEREF\_OF\_NULL.RET

Детектор предназначен для ситуаций проверки возвращаемого значения функций на ноль.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | Normal |
| Надежность | Unknown |
| Источник | SvEng |
| Тип ошибки | Ошибка разработчика |
| CWE | CWE476;CWE690 |
| CVE | — |
| Возможные последствия | Может привести к сбою выполнения или исключению (NullPointerException) при попытке обращения к неинициализированному ресурсу. |
| Комментарий | Рекомендуется всегда проверять возвращаемое функцией значение на null, особенно если документация не гарантирует ненулевой результат. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  String result = mayReturnNull();  // Попытка разыменования без проверки  System.out.println("Length: " + result.length());  } | public static void correctTest() {  String result = mayReturnNull();  if (result != null) {  System.out.println("Length: " + result.length());  } else {  System.out.println("Result is null");  }  } |

Методы устранения уязвимости при срабатывании детектора:

1. Непосредственно после вызова функции, которая может вернуть null, добавьте явную проверку на null. Используйте оператор if (returnValue != null) перед любым обращением к returnValue.
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 явного представления возможности отсутствия возвращаемого значения. Это заставляет явно проверять наличие значения перед его извлечением с помощью методов isPresent() и get() (или orElse(), orElseGet()).