DEREF\_AFTER\_NULL.RET

Детектор отслеживает ситуации проверки результата некоторых функций, возвращающих указатель, на нулевое значение.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | Normal |
| Надежность | Unknown |
| Источник | SvEng |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Это может привести к разыменованию null-указателя, если состояние переменной изменилось между проверкой и использованием, что вызовет исключение или сбой. |
| Комментарий | Следует обеспечить, чтобы значение не становилось null между проверкой и использованием, особенно в многопоточных средах или при изменении состояния переменных. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  String value = getNullable();  if (value == null) {  System.out.println("Value is null");  }  // если value == null, это вызовет NullPointerException  System.out.println("Length: " + value.length());  } | public static void correctTest() {  String value = getNullable();  if (value == null) {  System.out.println("Value is null");  return;  }  System.out.println("Length: " + value.length());  } |

Методы устранения уязвимости при срабатывании детектора:

1. Непосредственно после проверки на null используйте оператор return или throw для выхода из метода, если значение оказалось нулевым. Это предотвращает выполнение кода, который может привести к NullPointerException.
2. Если возможно, реорганизуйте код так, чтобы использование потенциально нулевой ссылки происходило непосредственно внутри блока if (value != null). Это гарантирует, что разыменование выполняется только при ненулевом значении.