INT\_OVERFLOW.COND

Указывает на ситуацию, когда переполнение целочисленного типа может произойти в условном выражении (например, в if-else структуре).

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | Normal |
| Надежность | VeryLow |
| Источник | SvEng |
| Тип ошибки | Ошибка разработчика |
| CWE | CWE190;CWE191;CWE194;CWE195;CWE196;CWE197 |
| CVE | — |
| Возможные последствия | Переполнение может привести к неожиданным результатам при выполнении условий, что может повлиять на логику программы. |
| Комментарий | При написании условий важно учитывать все возможные сценарии, включая переполнение целочисленных типов, и использовать проверки на переполнение или типы данных, которые обеспечивают безопасную работу с большими значениями. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest(int a, int b) {  if (a + b > a) { // Здесь может произойти переполнение  System.out.println("Condition passed, but overflow occurred.");  } else {  System.out.println("Condition failed, no overflow.");  }  } | public static void correctTest(int a, int b) {  if (a <= Integer.MAX\_VALUE - b) {  if (a + b > a) {  System.out.println("Condition passed, no overflow.");  }  } else {  System.out.println("Overflow detected, condition avoided.");  }  } |

Методы устранения уязвимости при срабатывании детектора:

1. Измените условное выражение так, чтобы избежать потенциального переполнения при вычислении условия. Перенесите части выражения или используйте эквивалентные проверки, которые не приводят к переполнению.
2. Используйте более широкий тип данных (например, long) для выполнения арифметических операций в условном выражении, если это возможно. Затем сравните результат с границами исходного типа.
3. При сложении a + b > a, безопасной проверкой на переполнение будет a <= Integer.MAX\_VALUE - b. Аналогично, для вычитания, умножения и других операций существуют безопасные способы проверки.
4. Разбейте сложное условное выражение на несколько более простых проверок, чтобы изолировать потенциально переполняющиеся операции и обработать их отдельно.