INT\_OVERFLOW.LOOP.STRICT

Детектор указывает на ситуацию, когда происходит переполнение целочисленного типа данных в цикле. "STRICT" в данном контексте указывает на то, что анализатор кода строго проверяет все возможные пути выполнения кода на предмет таких ошибок.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | Normal |
| Надежность | Unknown |
| Источник | SvEng |
| Тип ошибки | Ошибка разработчика |
| CWE | CWE190;CWE191;CWE194;CWE195;CWE196;CWE197 |
| CVE | — |
| Возможные последствия | Переполнение целочисленного типа в цикле, даже при строгой проверке, может привести к сложным ошибкам, особенно если код имеет несколько ветвей или зависит от динамических значений, что может привести к некорректной работе программы или сбоям. |
| Комментарий | Этот тип ошибки требует особого внимания, так как даже при наличии проверок переполнение может быть сложно обнаружить, особенно в сложных логических конструкциях или при изменении входных данных. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  int result = 0;  for (int i = 0; i < Integer.MAX\_VALUE; i++) {  result += i;  if (result < 0) {  System.out.println("Overflow detected!");  break;  }  }  } | public static void correctTest() {  long result = 0L;  for (int i = 0; i < Integer.MAX\_VALUE; i++) {  // Проверка на переполнение до выполнения операции  if (result + i > Integer.MAX\_VALUE) {  System.out.println("Overflow detected!");  break;  }  result += i;  }  } |

Методы устранения уязвимости при срабатывании детектора:

1. Используйте тип данных с большим диапазоном (например, long) для переменных, которые могут переполниться в цикле. Это наиболее надежный способ предотвратить переполнение.
2. Перед выполнением арифметической операции внутри цикла, которая может привести к переполнению, явно проверьте, не выйдет ли результат за пределы допустимого диапазона int.
3. Если переполнение является частью логики программы (например, при работе с кольцевыми буферами или хеш-функциями), убедитесь, что это поведение явно контролируется и обрабатывается корректно.
4. В условиях выхода из цикла используйте проверки, которые не зависят от потенциального переполнения счетчика или других переменных.