FB.DM\_NEXTINT\_VIA\_NEXTDOUBLE

Детектор определяет ошибку, в ходе которой если r является java.util.Random, вы можете сгенерировать случайное число от 0 до n-1, используя r.nextInt(n), а не используя (int)(r.nextDouble() \* n).

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Использование nextDouble() для генерации случайного целого числа через (int)(r.nextDouble() \* n) может привести к сниженной равномерности распределения и потенциальным погрешностям из-за преобразования типов. |
| Комментарий | Рекомендуется использовать r.nextInt(n), так как этот метод специально предназначен для генерации равномерно распределённых целых чисел в заданном диапазоне и работает более эффективно, исключая возможные ошибки округления. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  Random random = new Random();  int randomNumber = (int) (random.nextDouble() \* 100);  System.out.println("Random number: " + randomNumber);  } | public static void correctTest() {  Random random = new Random();  int randomNumber = random.nextInt(100);  System.out.println("Random number: " + randomNumber);  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Замените на random.nextInt(n):** Везде, где вы видите код, генерирующий случайное целое число с использованием (int) (random.nextDouble() \* n), замените его на вызов метода random.nextInt(n), где n представляет собой верхнюю границу (невключительно) диапазона случайных чисел (т.е., для диапазона 0 до n-1).
2. **Убедитесь в правильности диапазона:** Метод nextInt(n) возвращает значение в диапазоне [0, n). Если вам нужен другой диапазон (например, [min, max]), используйте random.nextInt(max - min + 1) + min.
3. **Проверьте весь код:** Найдите все места, где используется nextDouble() для генерации целых чисел и примените замену.
4. **Используйте ThreadLocalRandom в многопоточной среде (при необходимости):** В многопоточных приложениях рассмотрите использование java.util.concurrent.ThreadLocalRandom вместо java.util.Random для потенциального повышения производительности и снижения contention. Его использование аналогично: ThreadLocalRandom.current().nextInt(n).