FB.AT\_OPERATION\_SEQUENCE\_ON\_CONCURRENT\_ABSTRACTION

Детектор определяет ошибку, в ходе которой код содержит последовательность вызовов параллельной абстракции (например, параллельной хэш-карты). Эти вызовы не будут выполняться атомарно.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | — |
| CVE | — |
| Возможные последствия | Последовательные вызовы методов на параллельных структурах данных, таких как ConcurrentHashMap, могут привести к состоянию гонки, так как операции не выполняются атомарно. Это может вызвать неконсистентность данных, неожиданные результаты и ошибки в многопоточной среде. |
| Комментарий | При работе с конкурентными структурами данных следует использовать атомарные операции или синхронизацию, например, compute, putIfAbsent, merge или внешние механизмы блокировки, чтобы гарантировать корректность выполнения последовательных операций. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  ConcurrentHashMap<String, Integer> map = new ConcurrentHashMap<>();  String key = "example";  if (map.contains(key)) {  // другой поток может изменить  // карту после этой проверки  map.put(key, map.get(key) + 1);  } else {  map.put(key, 1);  }  } | public static void correctTest() {  ConcurrentHashMap<String, Integer> map = new ConcurrentHashMap<>();  String key = "example";  // Гарантированно атомарная операция  map.compute(key, (k, v) -> (v == null) ? 1 : v + 1);  } |