FB.STI\_INTERRUPTED\_ON\_CURRENTTHREAD

Этот метод вызывает вызов Thread.currentThread() только для вызова метода прерывания(). Поскольку прерывание() является статическим методом, проще и понятнее использовать Thread.interrupted().

Таблица №1 – Основные характеристики детектора FB.STI\_INTERRUPTED\_ON\_CURRENTTHREAD

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Interrupt on Current Thread |
| CWE | --- |
| CVE | --- |
| Возможные последствия | Некорректная обработка прерывания потока, что может привести к: неожиданному завершению работы, потере данных или неконсистентному состоянию, Deadlock, если поток не обрабатывает InterruptedException. Нарушение логики работы, если поток не ожидает прерывания |
| Комментарий | Использовать interrupt() только для управления потоком извне; восстанавливать статус прерывания после обработки InterruptedException |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public static void incorrectTest() {  try {  Thread.sleep(1000);  // Плохо: прерываем текущий поток без причины  Thread.currentThread().interrupt();  } catch (InterruptedException e) {  // Неправильно: не восстанавливаем флаг прерывания  System.out.println("Sleep interrupted");  }  } | public static void correctTest() {  try {  Thread.sleep(1000);  } catch (InterruptedException e) {  // Правильно: восстанавливаем статус прерывания  Thread.currentThread().interrupt();  System.out.println("Sleep interrupted properly");  }  } |

Методы устранения уязвимости при срабатывании детектора.

1. Не вызывать interrupt() без необходимости: прерывание должно использоваться только для управления потоком из другого потока
2. Правильно обрабатывать InterruptedException: либо пробрасывать исключение дальше, либо восстанавливать статус прерывания через Thread.currentThread().interrupt()
3. Использовать проверку Thread.interrupted() только когда это ожидаемо