FB.IP\_PARAMETER\_IS\_DEAD\_BUT\_OVERWRITTEN

Детектор определяет ошибку, в ходе которой начальное значение этого параметра игнорируется, и здесь параметр перезаписывается. Это часто указывает на ошибочное мнение, что запись в параметр будет передана обратно вызывающей стороне.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | CWE-563 |
| CVE | — |
| Возможные последствия | Логическая ошибка в коде, неиспользуемый параметр, бесполезная операция, возможное неправильное понимание механизма передачи параметров в Java |
| Комментарий | Начальное значение параметра игнорируется, так как оно сразу перезаписывается. Это может быть индикатором ошибки, если разработчик ожидал, что изменение параметра отразится на вызывающем коде (что невозможно для примитивных типов в Java). Рекомендуется либо использовать значение параметра, либо возвращать новый результат вместо его перезаписи. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest(int value) {  value = 42;  // ...  // В Java примитивные типы передаются по значению,  // поэтому изменение никак не повлияет на  // вызывающий код  } | public static void correctTest(int value) {  // ... преобразования с value  // или же передать AtomicInteger вместо int  // и изменять его:  AtomicInteger x = new AtomicInteger(value);  x.set(42);  } |

Методы устранения уязвимости при срабатывании детектора:

1. **Используйте параметр:** Если начальное значение параметра необходимо для логики метода, убедитесь, что оно используется до того, как будет перезаписано.
2. **Возвращайте новое значение:** Если цель метода состоит в том, чтобы преобразовать входное значение и вернуть результат, используйте оператор return для возвращения нового значения.
3. **Изменяйте состояние объекта (для ссылочных типов):** Если параметр является ссылкой на объект, и вы хотите, чтобы изменения были видны вызывающей стороне, изменяйте состояние самого объекта (его поля или содержимое коллекции), а не присваивайте параметру новую ссылку.
4. **Пересмотрите сигнатуру метода:** Если параметр никогда не используется до перезаписи, подумайте, нужен ли он вообще в сигнатуре метода. Возможно, значение, которое присваивается параметру, может быть определено внутри метода без передачи извне.
5. **Используйте изменяемые обертки (при необходимости):** Для имитации передачи примитивных типов "по ссылке" можно использовать классы-обертки, такие как AtomicInteger или AtomicReference, состояние которых можно изменять внутри метода и эти изменения будут видны вызывающей стороне.