FB.SF\_SWITCH\_FALLTHROUGH

Ошибка "statement found where one case falls through to the next case" указывает на то, что в вашем коде внутри конструкции switch для одного из случаев (case) отсутствует оператор break или return, который предотвратил бы "проброс" управления к следующему случаю.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | UAST |
| Тип ошибки | Ошибка разработчика |
| CWE | CWE484 |
| CVE | — |
| Возможные последствия | Отсутствие оператора break или return в конструкции switch может привести к неожиданному поведению программы, когда выполнение продолжается в следующий блок case, что может вызвать ошибочные результаты или неожиданные побочные эффекты. |
| Комментарий | Для каждого случая в конструкции switch необходимо либо явно завершить выполнение с помощью break или return, либо указать намеренную логику для пропуска выполнения через несколько блоков case с соответствующими комментариями. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest(int value) {  switch (value) {  case 1:  System.out.println("1");  break;  case 2:  System.out.println("2");  // Отсутствует break, что приводит к переходу к следующему case  case 3:  System.out.println("3");  break;  default:  System.out.println("default");  }  } | public static void correctTest(int value) {  switch (value) {  case 1:  System.out.println("q");  break;  case 2:  System.out.println("2");  break;  case 3:  System.out.println("3");  break;  default:  System.out.println("default");  }  } |

Методы устранения уязвимости при срабатывании детектора:

1. Проанализируйте логику каждого case блока, где отсутствует оператор завершения. Определите, является ли намеренным переход к следующему case.
2. Если переход к следующему case не является намеренным, добавьте оператор break; в конце текущего case блока. Оператор break прервет выполнение оператора switch и передаст управление следующей за ним инструкции.
3. Если в конце case блока должен происходить выход из метода, используйте оператор return;.
4. Если выполнение текущего case должно завершиться выбросом исключения, используйте оператор throw new Exception(...);.
5. В редких случаях, когда "проваливание" в следующий case является намеренным и логически обоснованным, добавьте явный комментарий (// fall through, /\* falls through \*/, @SuppressWarnings("fallthrough")) непосредственно перед началом следующего case блока. Это сигнализирует о вашем намерении и предотвратит появление предупреждений от статических анализаторов в будущем. Однако намеренное "проваливание" следует использовать с осторожностью, так как оно может сделать код менее читаемым и поддерживаемым.
6. Убедитесь, что логика обработки нескольких case блоков при намеренном "проваливании" корректна и не приводит к нежелательным побочным эффектам или ошибкам.