FB.TQ\_MAYBE\_SOURCE\_VALUE\_REACHES\_NEVER\_SINK

Значение, которое помечено как возможное экземпляр значений, обозначенных квалификатором типа, и это значение гарантированно будет использоваться таким образом, который запрещает значения, обозначаемые этим квалификатором типа.

Значение аннотировано потенциально являющимся экземпляром значений определяемых квалификатором типов, хотя оно гарантированно будет использоваться значением, которое запрещает использование значений, определяемых этим квалификатором.

Таблица №1 – Основные характеристики детектора FB.TQ\_MAYBE\_SOURCE\_VALUE\_REACHES\_NEVER\_SINK

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Dead Data Flow |
| CWE | --- |
| CVE | --- |
| Возможные последствия | Бессмысленные вычисления - ресурсы тратятся на обработку данных, которые никогда не используются; усложнение кода - наличие "мертвого" кода снижает читаемость; потенциальные уязвимости - неиспользуемые данные могут содержать чувствительную информацию; затруднение рефакторинга - ложные зависимости между компонентами |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public static void incorrectTest() {  // 1. Результат метода игнорируется  String data = fetchData(); // @MaybeSource  processData(data); // @NeverSink (ничего не делает с данными)  // 2. Бессмысленное присваивание  int computedValue = computeExpensiveValue(); // Тяжелые вычисления  // computedValue нигде не используется  // 3. Параметр метода не используется  logMessage("Important event", true); // Второй параметр игнорируется  } | public static void correctTest() {  // 1. Используем возвращаемое значение  String data = fetchData();  saveToDatabase(data); // Данные действительно используются  // 2. Убираем ненужные вычисления  if (needsComputation()) {  int value = computeExpensiveValue();  useValue(value);  }  // 3. Упрощаем метод  logMessage("Important event"); // Убрали неиспользуемый параметр  } |

Методы устранения уязвимости при срабатывании детектора.

1. Удаление неиспользуемого кода;
2. Рефакторинг методов: удалите неиспользуемые параметры и измените void-методы, возвращающие данные, на возвращающие значение.