FB.IMA\_INEFFICIENT\_MEMBER\_ACCESS

Этот метод внутреннего класса считывает или записывает в частную переменную-член владеющего класса или вызывает закрытый метод владеющего класса. Компилятор должен создать специальный метод для доступа к этому частному члену, что делает его менее эффективным. Ослабление защиты переменной-члена или метода позволит компилятору рассматривать это как обычный доступ.

Таблица №1 – Основные характеристики детектора FB.IMA\_INEFFICIENT\_MEMBER\_ACCESS

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Performance Issue |
| CWE | --- |
| CVE | --- |
| Возможные последствия | * Избыточные обращения к полям/методам * Снижение производительности * Увеличение времени выполнения |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public static void incorrectTest() {  class DataHolder {  int value = 0;  int getValue() { return value; }  }    DataHolder holder = new DataHolder();    // Некорректно: многократный доступ к полю/методу  for (int i = 0; i < 100; i++) {  System.out.println(holder.getValue()); // Доступ в каждой итерации  }  } | public static void correctTest() {  class DataHolder {  int value = 0;  int getValue() { return value; }  }    DataHolder holder = new DataHolder();    // Корректно: кэширование значения  int value = holder.getValue();  for (int i = 0; i < 100; i++) {  System.out.println(value); // Использование кэшированного значения  }    // Альтернатива для изменяемых полей  for (int i = 0; i < 100; i++) {  int currentValue = holder.getValue(); // Если значение может меняться  System.out.println(currentValue);  }  } |