**FB.NM\_METHOD\_CONSTRUCTOR\_CONFUSION**

Этот обычный метод имеет то же имя, что и класс, в котором он определен. Вероятно, он задумывался как конструктор. Если он должен был быть конструктором, следует удалить объявление возвращаемого значения void. Если этот метод был определен случайно, была найден ошибка, был определен правильный конструктор, но нет возможностти избавиться от этого метода из-за обратной совместимости, следует объявить этот метод устаревшим.

Таблица №1 – Основные характеристики детектора **FB.NM\_METHOD\_CONSTRUCTOR\_CONFUSION**

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Naming Convention |
| CWE | --- |
| CVE | --- |
| Возможные последствия | Путаница в понимании кода |
| Комментарий (опционально) | Нарушение соглашений об именовании |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public static void incorrectExample() {  Document doc = new Document();  Document result = doc.Document(); // Выглядит как конструктор, но это метод  System.out.println("Incorrect method call: " + result.getTitle());  } | public static void correctExample() {  Document doc = Document.createDefault();  Document updated = doc.withTitle("New Title");  System.out.println("Correct method call: " + updated.getTitle());  }  } |

Методы устранения уязвимости при срабатывании детектора

1. Переименование метода:

java

// Было:

public ClassName methodName() {...}

// Стало:

public ClassName createOrGet() {...}

1. Преобразование в статический фабричный метод:

java

public static Configuration loadConfiguration() {

Configuration config = new Configuration();

config.settings = readFromFile();

return config;

}

1. Изменение сигнатуры на void (если метод должен быть void):

java

public void initialize() {

this.settings = readFromFile();

}