FB.NP\_NONNULL\_FIELD\_NOT\_INITIALIZED\_IN\_CONSTRUCTOR

Поле помечено как ненулевое, но конструктор не записал его. Поле может быть инициализировано в другом месте во время конструктора или всегда может быть инициализировано перед использованием.

Таблица №1 – Основные характеристики детектора FB.NP\_NONNULL\_FIELD\_NOT\_INITIALIZED\_IN\_CONSTRUCTOR

|  |  |
| --- | --- |
| Категория критичности | Minor |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Null Pointer Dereference |
| CWE | CWE476 |
| CVE | --- |
| Возможные последствия | Некорректное поведение программы, сбои при обращении к полю, которое может быть null |
| Комментарий (опционально) | Проверьте инициализацию всех @Nonnull полей в конструкторе |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Потенциально небезопасное | Корректная конструкция |
| public static class IncorrectExample {  @Nonnull private String data; // Не инициализировано в конструкторе  public IncorrectExample() {  // Поле data остается null  }  } | public static class CorrectExample {  @Nonnull private String data = ""; // Инициализация по умолчанию  public CorrectExample() {  // Дополнительная логика, если требуется  }  // Или инициализация в конструкторе:  public CorrectExample(String data) {  this.data = data;  }  } |

Методы устранения уязвимости при срабатывании детектора

1. Инициализация полей в конструкторе: Все поля, помеченные @Nonnull, должны быть явно инициализированы в каждом конструкторе класса.
2. Использование default-значений: Присваивайте полям значения по умолчанию (например, пустые строки или объекты).
3. Проверка аннотаций: Убедитесь, что аннотации @Nonnull соответствуют логике кода.