FB.NP\_UNWRITTEN\_PUBLIC\_OR\_PROTECTED\_FIELD

Детектор определяет ошибку, в ходе которой программа разыменовывает общедоступное или защищенное поле, в которое, похоже, никогда не было записано ненулевое значение. Если поле не инициализировано с помощью какого-либо механизма, не видимого при анализе, разыменование этого значения приведет к возникновению исключения нулевого указателя.

Таблица №1 – Основные характеристики детектора

|  |  |
| --- | --- |
| Категория критичности | MAJOR |
| Надежность | Unknown |
| Источник | SpotBugs |
| Тип ошибки | Ошибка разработчика |
| CWE | CWE476 |
| CVE | — |
| Возможные последствия | Попытка разыменовать поле, которое не инициализировано, может привести к возникновению исключения NullPointerException. |
| Комментарий | Если поле должно быть инициализировано, убедитесь, что оно получает значение до использования. Если это поле не требуется, рассмотрите возможность удаления или замены его на локальную переменную, чтобы избежать потенциальных ошибок. |

Таблица №2 – Примеры срабатывания детектора и устранения уязвимости

|  |  |
| --- | --- |
| Некорректное написание | Корректное написание |
| public static void incorrectTest() {  class Example {  // Поле не инициализируется  public String value;  public void printValue() {  // может быть NullPointerException  System.out.println(value.length());  }  }  Example example = new Example();  example.printValue();  } | public static void correctTest() {  class Example {  // Инициализация поля в конструкторе  public String value;  public Example(String value) {  this.value = value;  }  public void printValue() {  if (value != null) {  System.out.println(value.length());  }  }  }  Example example = new Example("Hello");  example.printValue();  } |

Методы устранения уязвимости при срабатывании детектора:

1. Инициализируйте все общедоступные и защищенные поля при их объявлении или в конструкторах класса.
2. Если поле может быть необязательным, рассмотрите использование java.util.Optional для явного представления этой возможности.
3. Если поле не предназначено для прямого доступа извне класса или его подклассов, сделайте его private и предоставьте методы доступа (геттеры и сеттеры).
4. Внимательно следите за жизненным циклом объекта и убедитесь, что общедоступные и защищенные поля получают ненулевое значение до того, как они могут быть использованы.
5. Рассмотрите возможность использования статического анализа (например, настроек SpotBugs) для более строгого контроля за потенциальными NullPointerException.